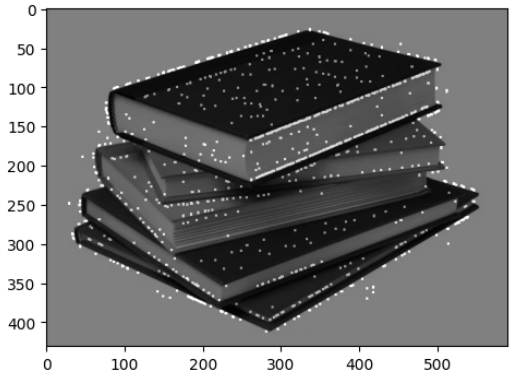
REPORT ON QUESTION 1

Number of features extracted by SIFT (with single octave) :

Type	Books	Building
Original	999	4978
Rotation(45)	920	4292
Rotation(180)	1022	4977
Upscale(x1.5)	1463	12996
Upscale(x2)	2610	25069
Downscale(x0.5)	389	1346
Downscale(x0.25)	140	251
Gaussian Blur(sigma = 2)	149	2649
Gaussian Blur(sigma = 3)	38	907
Gaussian Noise	884	2867

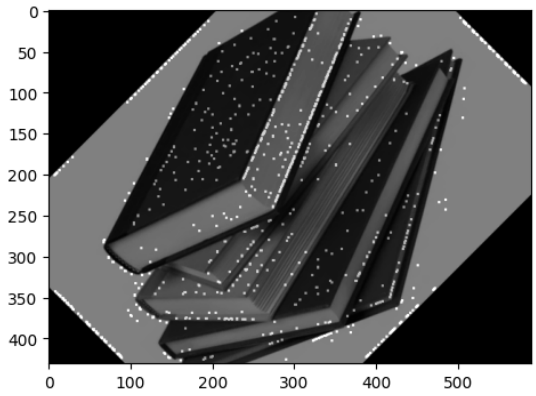
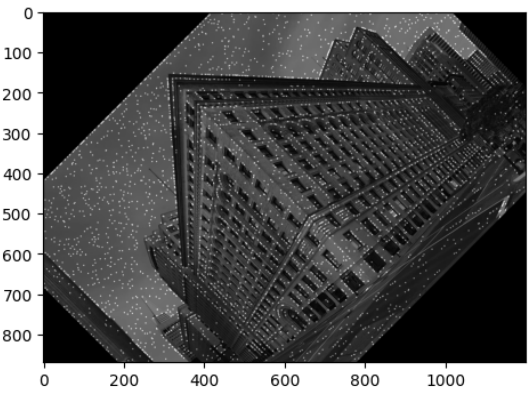
Analysis:

1. No of features extracted for building is very much higher than books as it can be seen from the image that in books the background of the books is very uniform and hence only a few features are from that uniform background. Where as building contains many edges, cornors and blobs, hence high number of features.

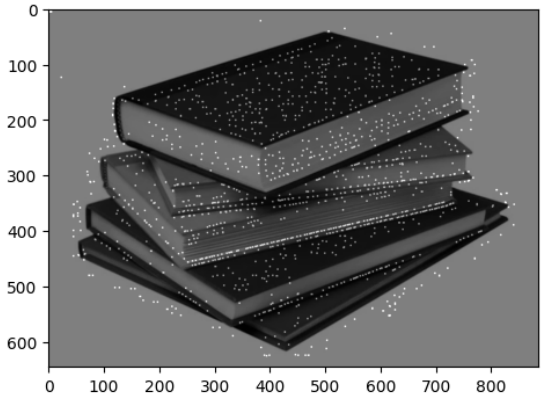
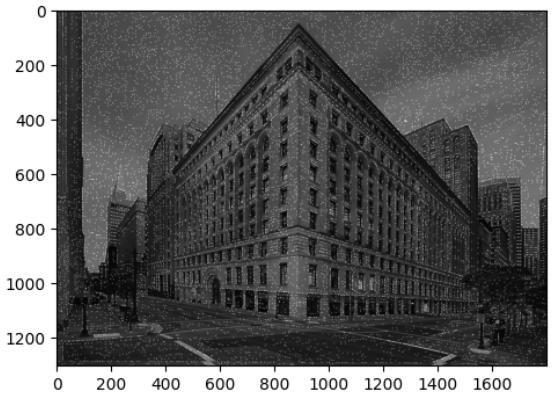


2. Rotation: (Computed for 2 angles : 45^{o} and 180^{o} couter clockwise) We can see that for 45^{o} there isn't much decrease in no of features for books as the region that got out of frame contained mostly uniform patches. But for the building as the features are uniformily distributed throughout the image , a major portion got out of frame after rotation hence

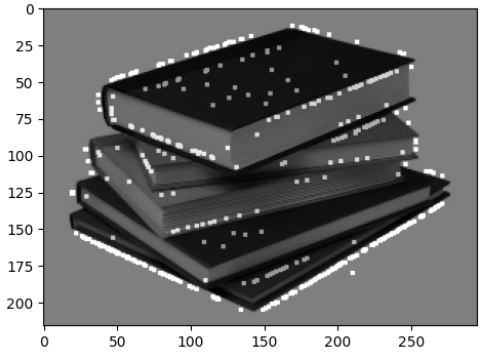
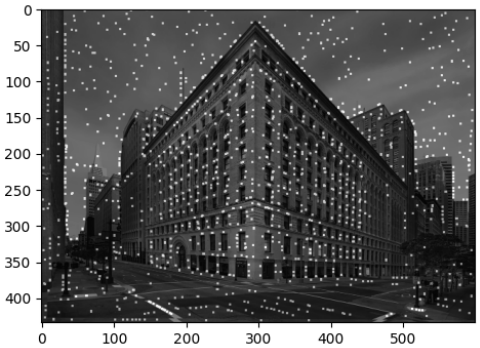
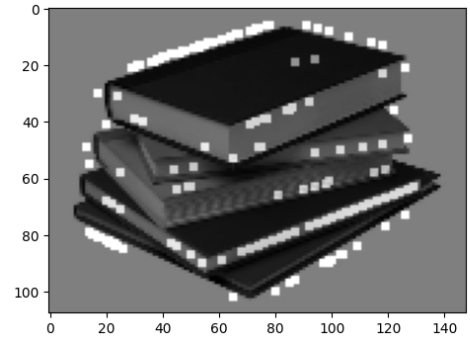
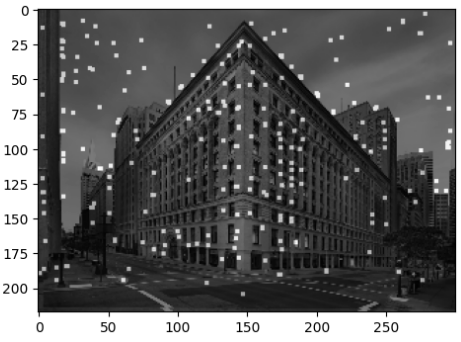
there a major decrease in the no of features as compared to original image. For 180° no of features are almost same as expected as it is just the image fliiped upside down.

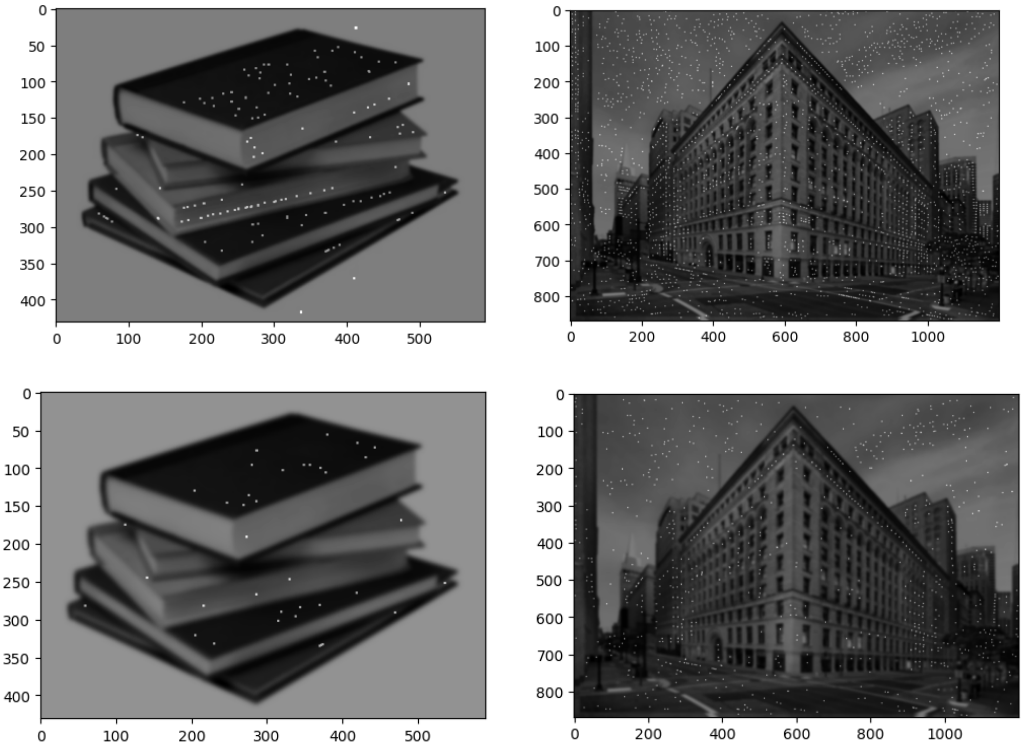
3. Upscale (x1.5 and x2): We can see that the no of features increased. We got extra features around the features from the original image. For x2 the features of books increased by 2.5 times and for building it increased by 5 times .

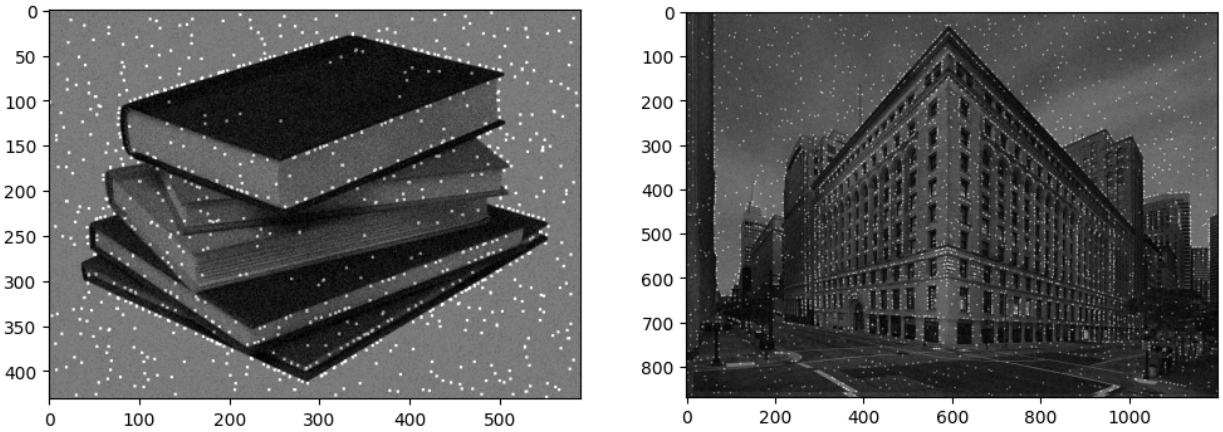
4. Downscale(x0.5 and x0.25): No of features significantly decreased

5. Gaussian Blur(sigma = 2 and 3): Smoothing decreased the number of features significantly. More the smoothing more the decrease in no of features as it can be seen for the table.



6. Gaussian Noise: Introduced many features in the background of books which were not present in the original image.



REPORT ON QUESTION 2

Library used : **PyTorch**

Part (a):

Objective :

To use an ImageNet Pre-Trained model to extract deep features for the given images and use these for classification using k-NN.

Details of Implementation:

Steps:

- Imported ResNe-18 pre-trained on ImageNet from Pytorch.
- Then removed the last fully conneted layer of ResNet-18 model to extract the features as desired.
- Some preprocessing done on all the images.
- Then used these pre-pocessed data to extract features of both train and test data.
- Feature vectors of train data and test data are assigned class labels and stored in numpy arrays.
- Then used k-nn (from sklearn library) with **k = 5** to fit the features from training data and then tested on test data.

Accuracy reported : **96.67 %**

Part (b):

Objective :

Replacing the classification layer (originally for 1000 classes) with new layer (for 6 classes) and only fine-tuning this layer.

Steps:

- Before changing the classification layer we set the requires_grad = 0 for all parameters to freeze all the layers.
- Then we change the classification layer so as to classify 6 objects and fine-tuned it with training data.
- Ran for 150 epochs (30 epochs – 5 times)
- Then tested on testing data

Accuracy reported : **100.00 %**

Part (c):

Simple 2 Linear layers: Ran for 60 epochs(30x2)

Accuracy reported **: 32.5 %**