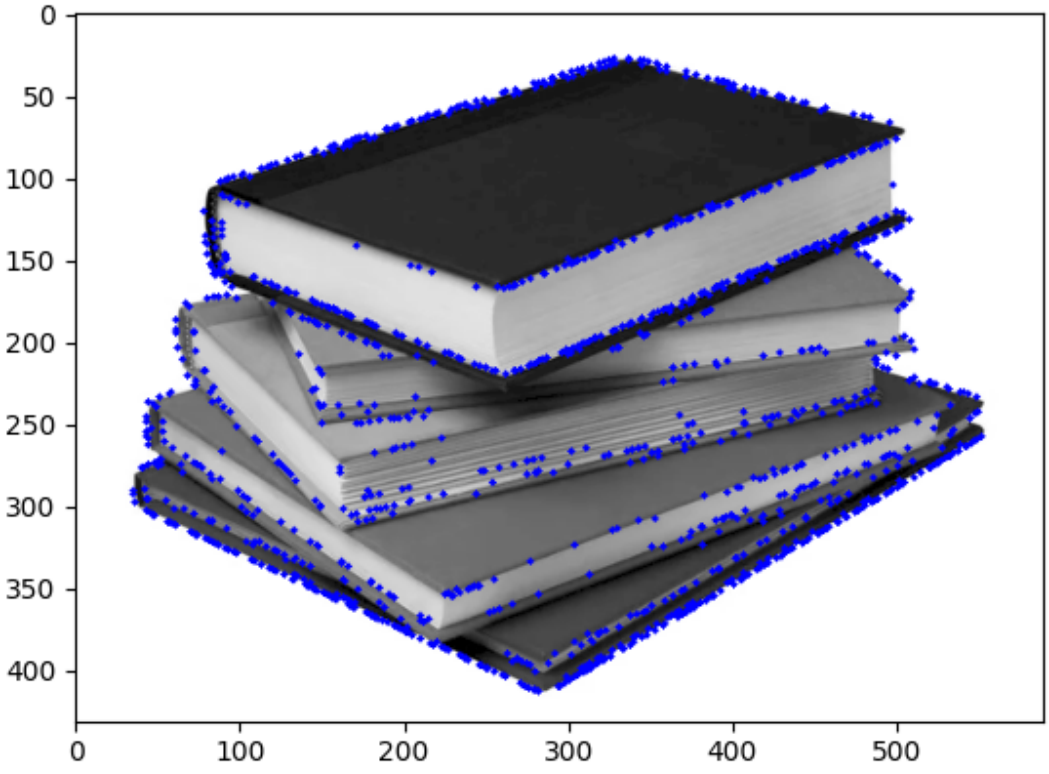
DIP Assignment - 1

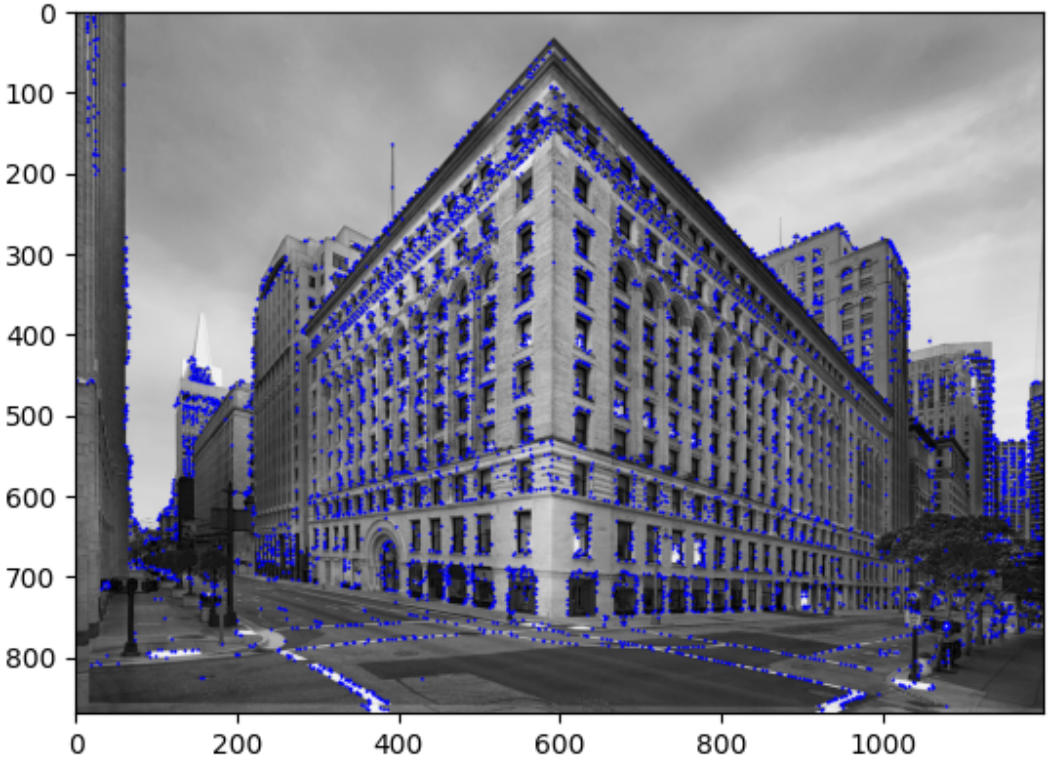
1. SIFT

Steps to detect key points:

- Construct the scale space extrema of size 5 for the given image starting with scale 0.707 and further scales multiplied by sqrt(2) with the previous scale.
- Construct DOG from scale-space extrema.
- Check each pixel value in a DOG with 26 neighbouring pixel values to decide whether it's an extremum.
- Thresholding was applied on the extremum points to get the keypoints.

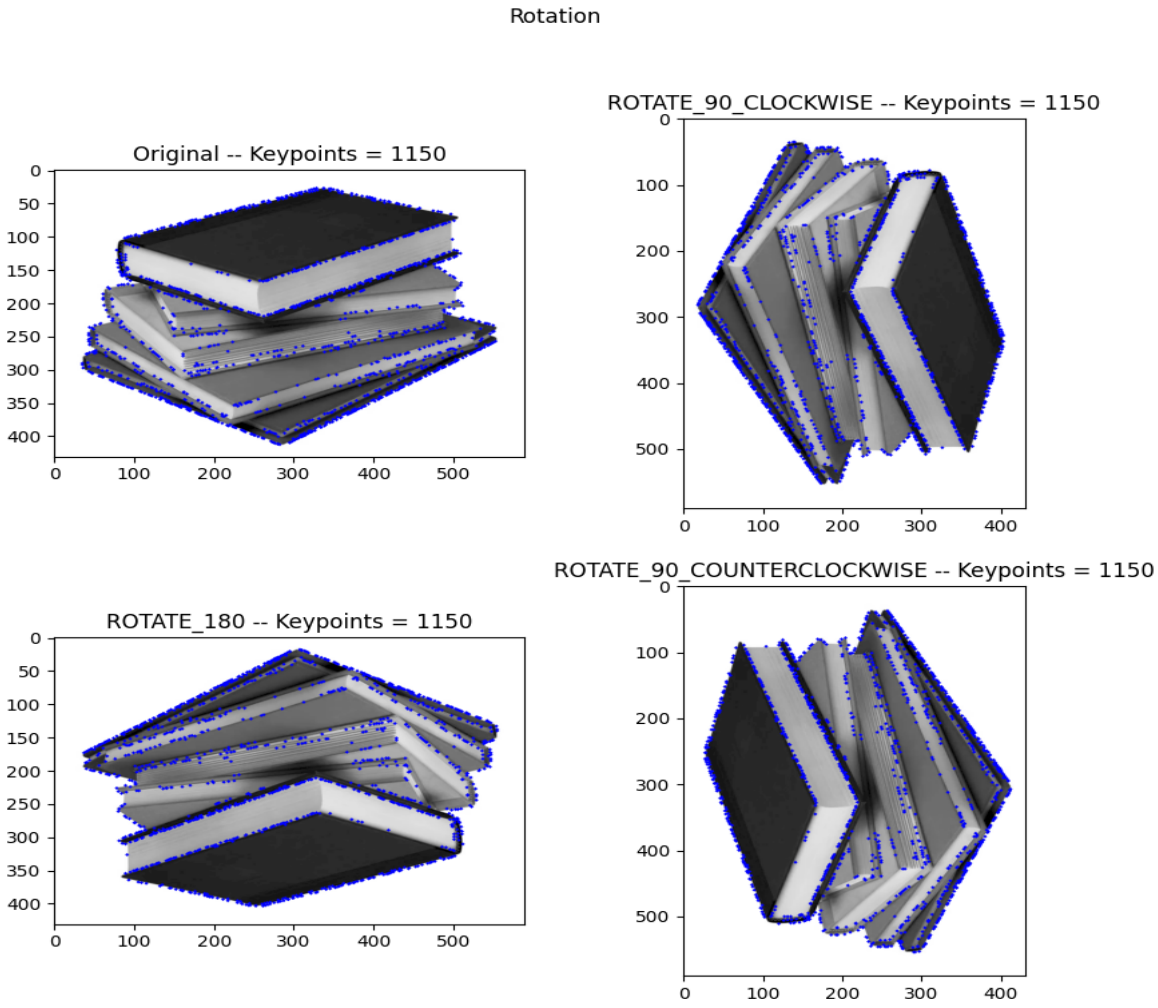
SIFT on original Images :



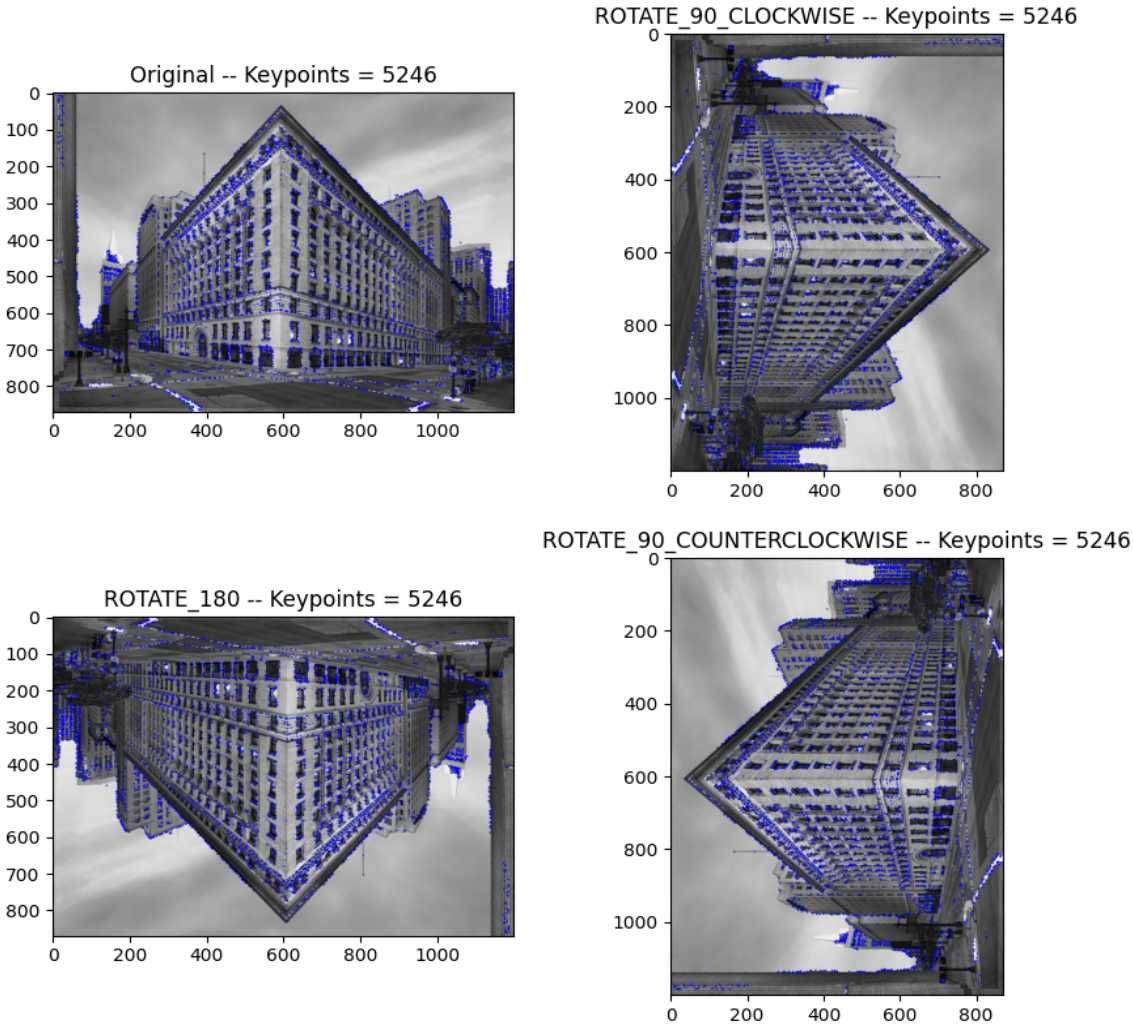


- We can observe that no key point was detected in the smooth regions of both images.
- It is evident that edges and corners of the book's image are detected as key points.
- In the image of the building, we can observe that the corners of windows in each room got detected as key points.
- Most detected key points correspond to edges and corners in an image, which are desirable features.

Rotation:

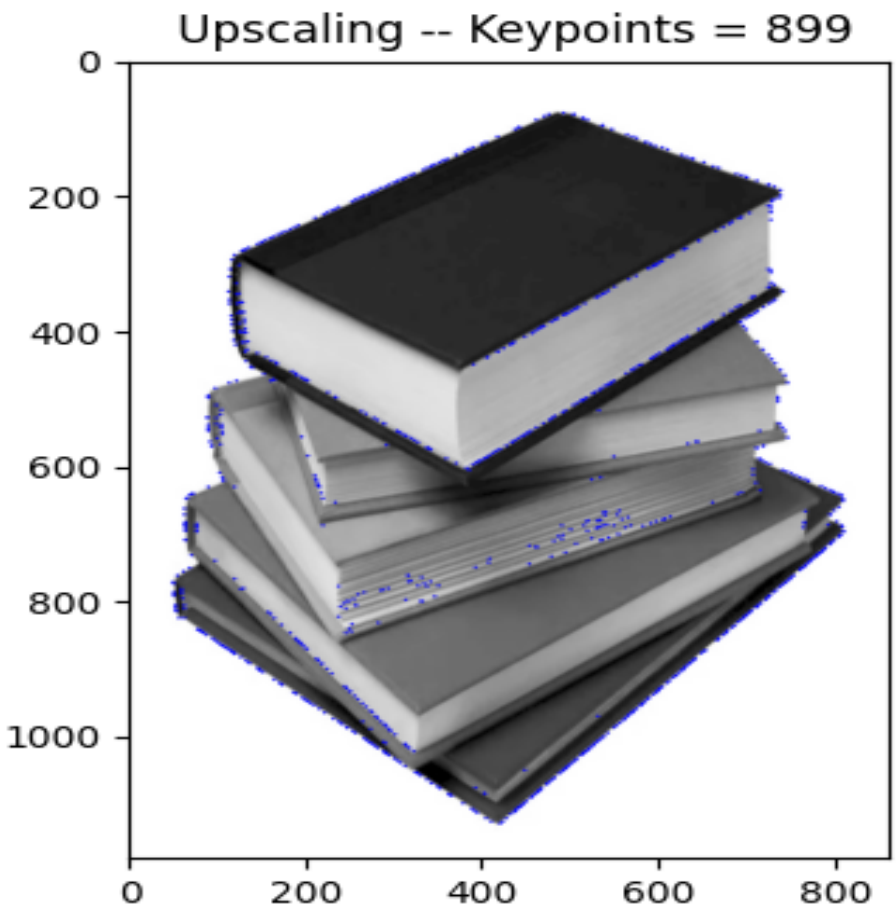

Rotation
Original -- Keypoints = 1150
ROTATE_90_CLOCKWISE -- Keypoints = 1150
ROTATE_180 -- Keypoints = 1150
ROTATE_90_COUNTERCLOCKWISE -- Keypoints = 1150

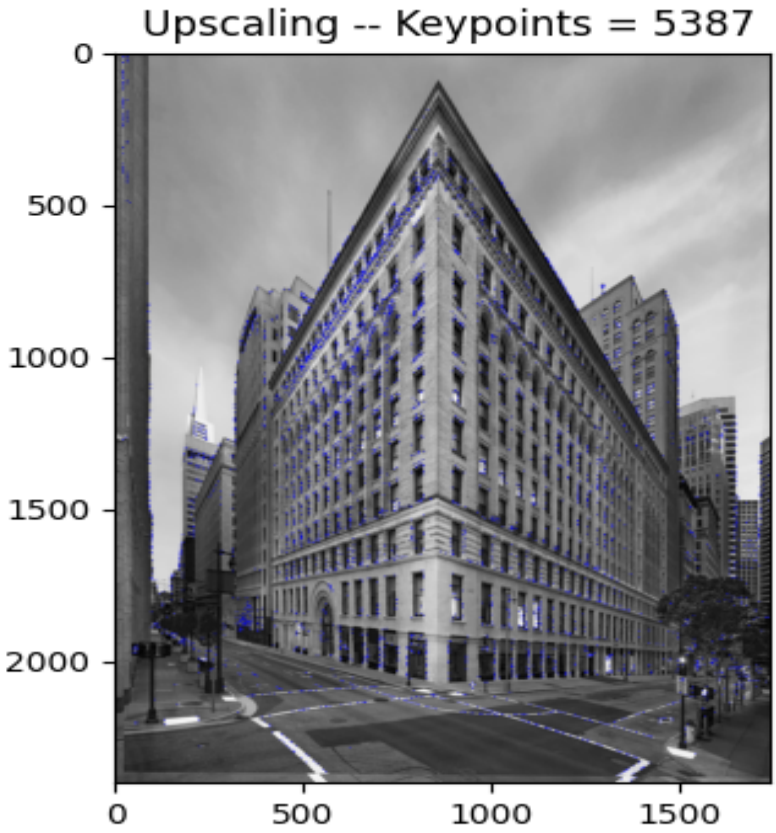
Rotation



- The number of key points detected is the same for the original and rotated images. Keypoints can also be matched between the original and rotated image.
- SIFT algorithm for finding key points is invariant to rotation.

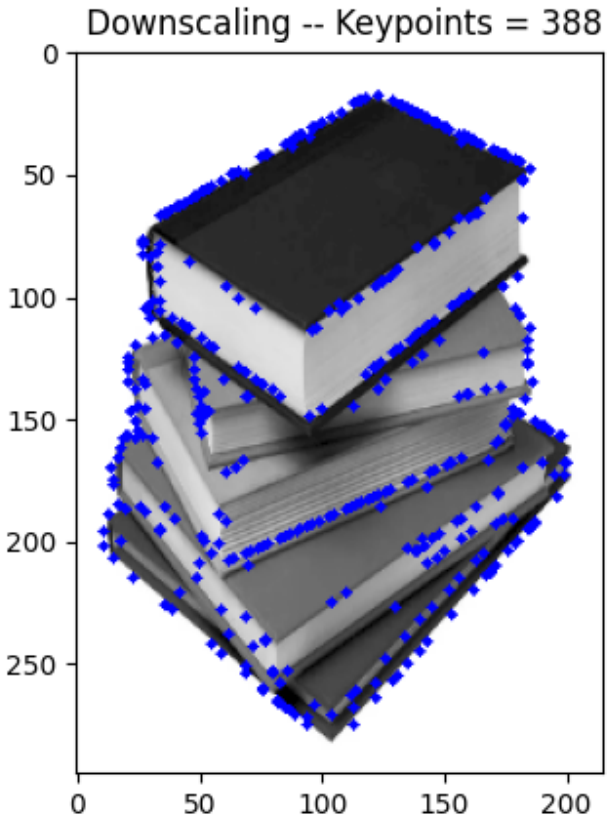
UpScaling :

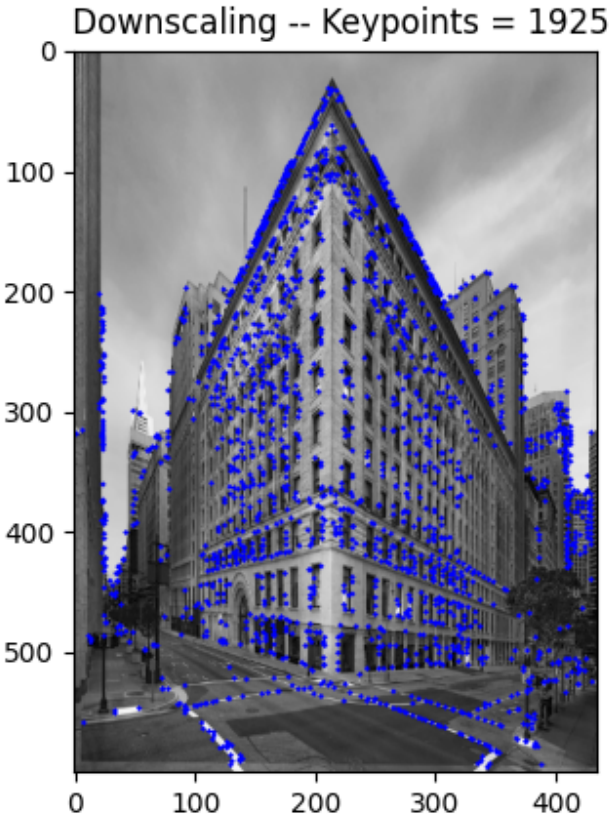

Upscaling -- Keypoints = 899
0
200
400
600
800
1000
0
200
400
600
800



- Upscaling doesn’t seem to affect the keypoints detected.

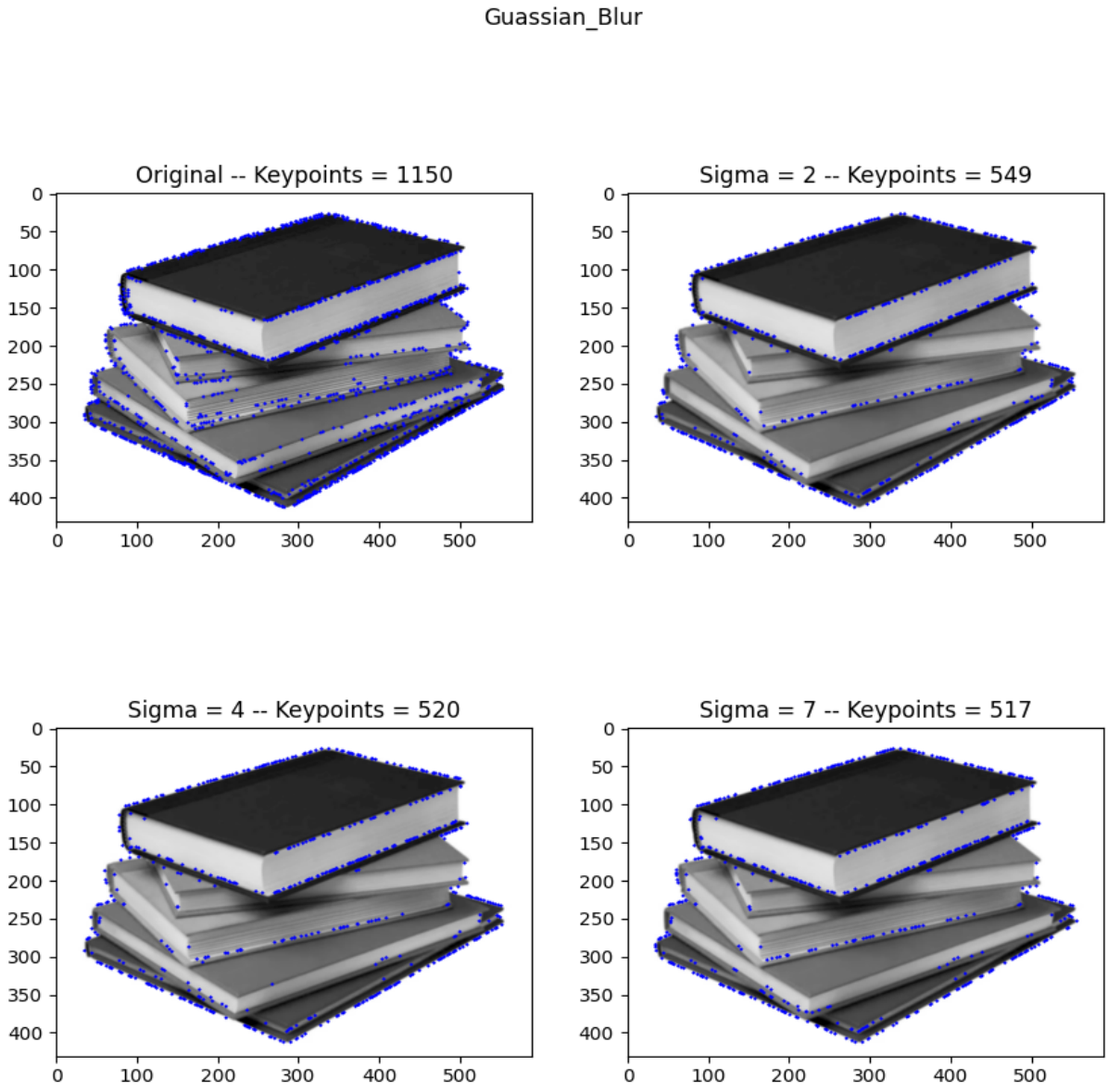
Downscaling :



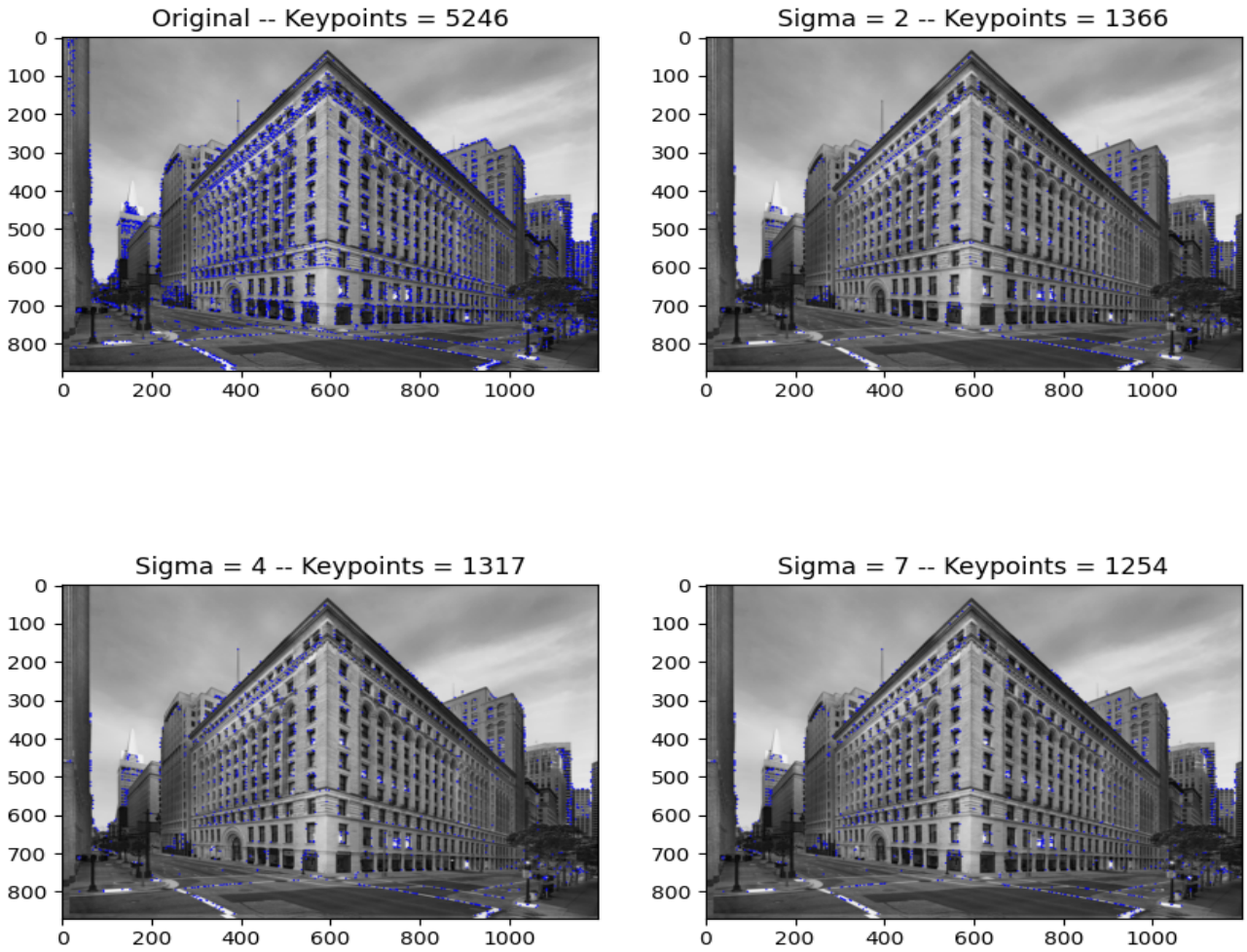


- Downsampling the image results in a decrease in the number of key points detected.
- Since the image size was reduced, potential key points detected in the original image might not have been included in the downsampled image.

Gaussian blur:

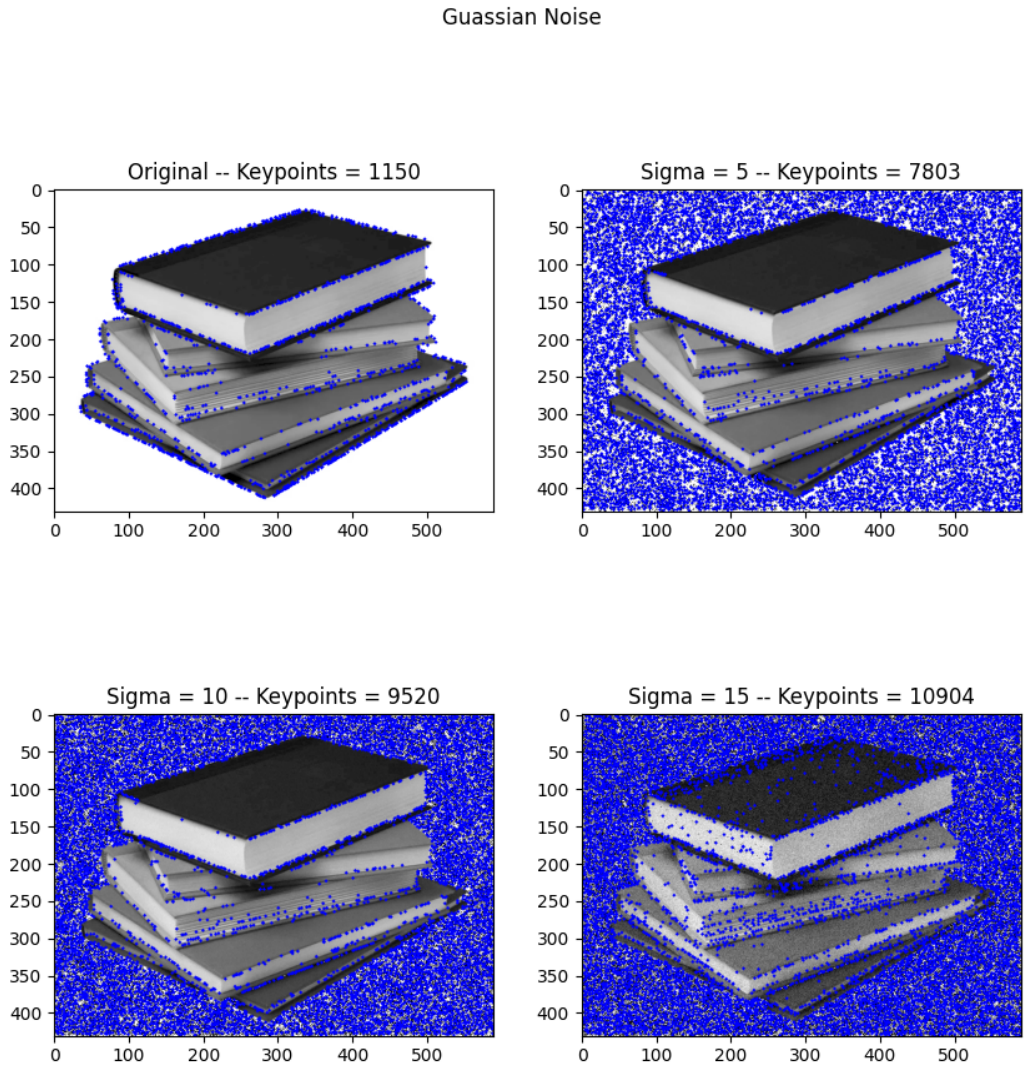


Guassian_Blur

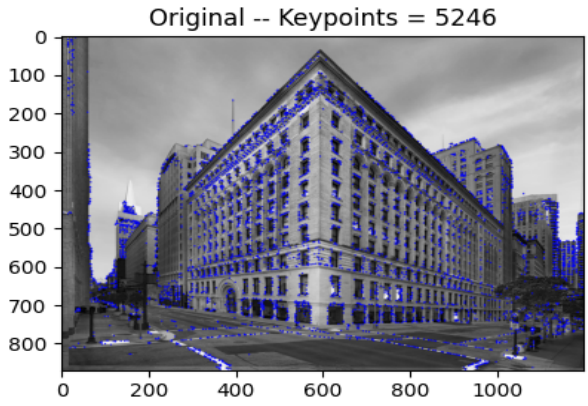


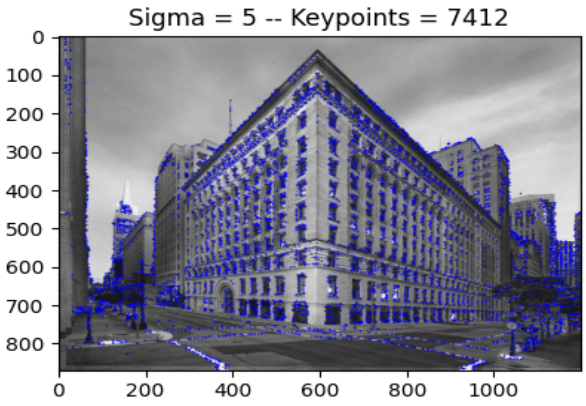
- Gaussian blur smoothens the image, which results in the loss of high-frequency details such as edges.
- As the sigma increases, blur content increases, which decreases the number of key points detected.

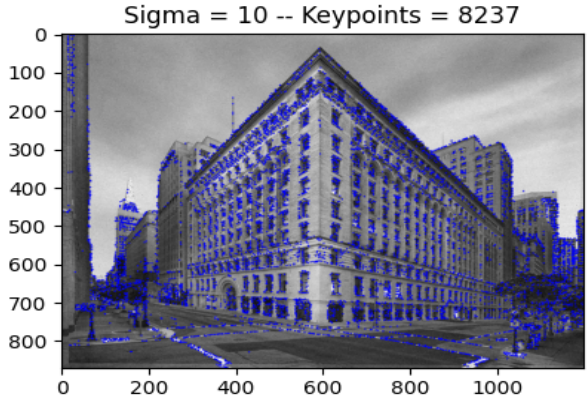
Gaussian noise:

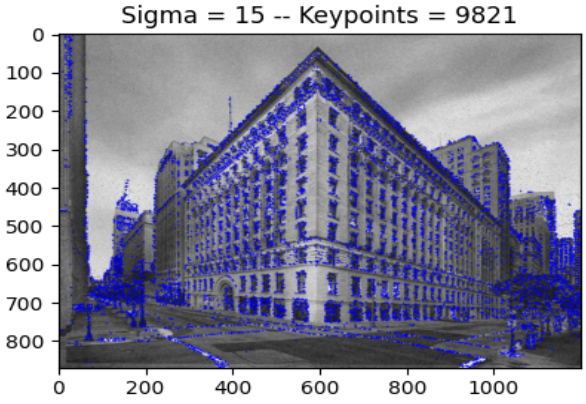


Guassian Noise









- Adding up Gaussian noise to the image, there will be a lot of fluctuations in the image pixel values, which account towards extremum in their neighbourhood of DOG.
- This will result in key points getting detected in the smooth region of the original image.
- As the amount of noise increases, the number of detected keypoints increases.
- It will result in several False Positives getting detected.

2. Classification of animals

i) Extracting features from Resnet18 and using k-NN model

- Extracted deep features for each image from the pre-trained Resnet18 model.
- Each feature vector of the image is of shape 512*1.
- k-NN algorithm was modelled to classify the feature vectors of images with num_neighbours = 3.
- Achieved an accuracy of 0.94.

	precision	recall	f1-score	support
bear	0.90	0.95	0.93	20
butterfly	0.95	1.00	0.98	20
camel	0.94	0.85	0.89	20
chimp	0.95	0.95	0.95	20
duck	1.00	0.90	0.95	20
elephant	0.91	1.00	0.95	20
accuracy			0.94	120
macro avg	0.94	0.94	0.94	120
weighted avg	0.94	0.94	0.94	120

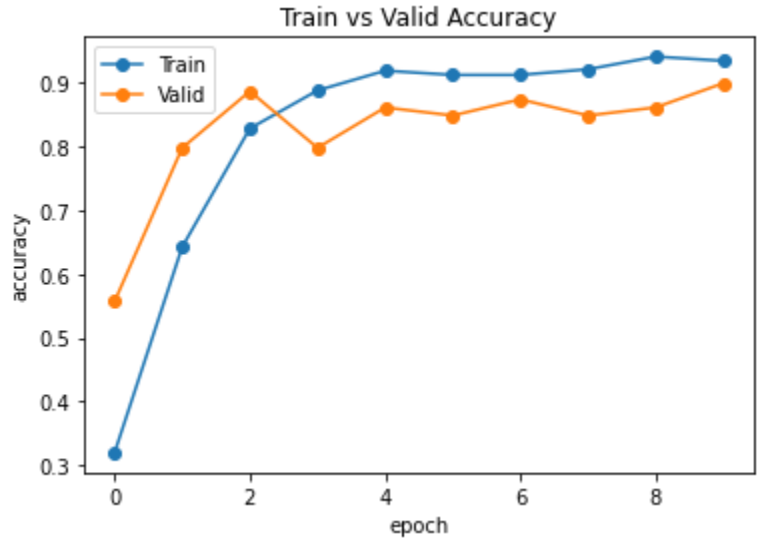
- Since Resnet18 was trained on a large dataset, we were able to extract the key features for our dataset.
- It enabled us to achieve 90+ accuracy using the kNN classifier on extracted features. It is evident that features extracted from Resnet18 are a better representation of the image.

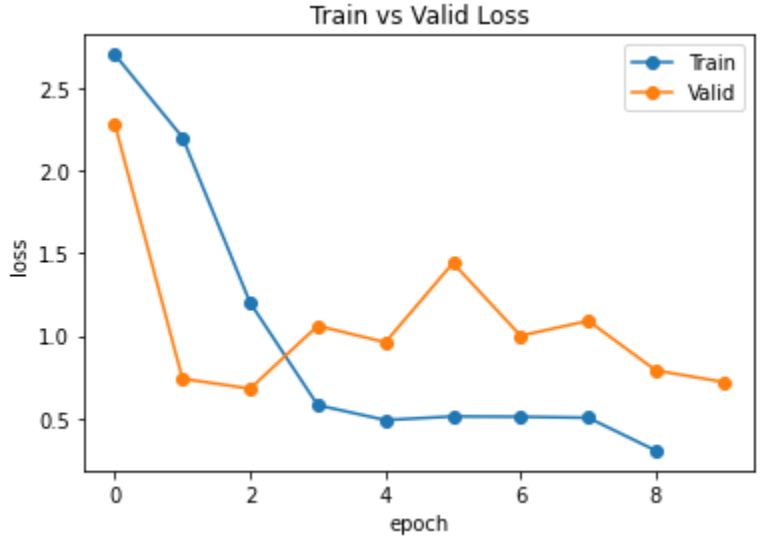
ii) Finetuning the resnet18 model

- The Resnet18 model was changed in such a way that the last layer of 1000 neurons was removed, and added a layer with six neurons.
- Froze the weights of the resnet18 model, and only the last layer weights are learnable in the new model.
- Divided the train data into train and validation sets to perform validation after each epoch and take the best model on the validation set.
- The final model is tested on the test set.

- Trained the model for ten epochs and achieved an accuracy of 100% on the test set.

Plots:





	precision	recall	f1-score	support
bear	1.00	1.00	1.00	20
butterfly	1.00	1.00	1.00	20
camel	1.00	1.00	1.00	20
chimp	1.00	1.00	1.00	20
duck	1.00	1.00	1.00	20
elephant	1.00	1.00	1.00	20
accuracy			1.00	120
macro avg	1.00	1.00	1.00	120
weighted avg	1.00	1.00	1.00	120

- Fine-tuning the model enabled the model to learn from extracted features to adapt to the application of classifying the animals.

iii) CNN architecture

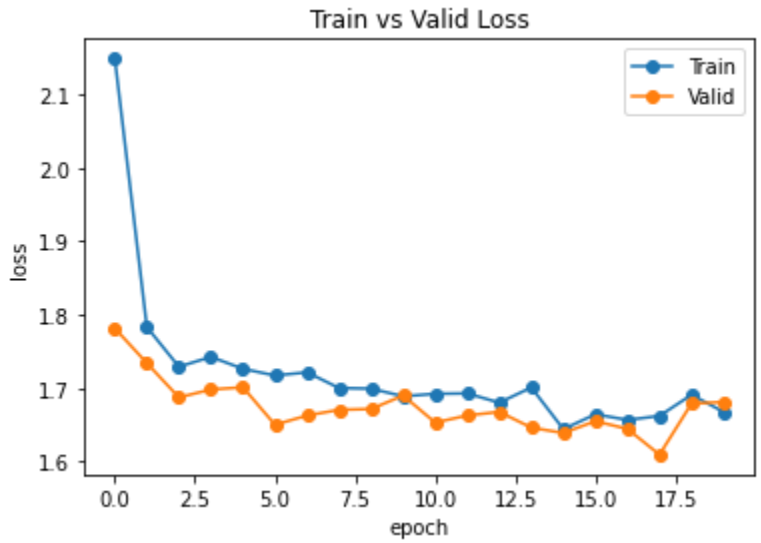
- Split the training data into train and validation sets.
- Performed different transformations on images, such as rotation, crop, and normalization.
- Created an architecture of 3 layers where each layer consists of a series of Convolution layers, a Batch Normalization layer, a Relu layer and a MaxPooling layer.
- CNN followed by Linear, Relu, Dropout and Linear layer.

CNN model information:

- **Loss function:** CrossEntropy Loss
- **Optimizer:** Adam optimiser
- **Learning rate:** 0.0001
- **Batch size:** 32

- Trained the model for ten epochs with the above parameters to the model.
- Achieved an accuracy of 40% on the test set.
- The reason for less accuracy is due to the small size of the training dataset. The model was not able to learn from the small dataset that was provided.

Loss on each epoch :



Acknowledgements :
https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html