DSE312 Assignment-3

Name : VK Susheel
Roll No. : 20297

<u>Code Description:</u>

The code is divided into 5 major cells. One each to perform classification using SIFT and HOG followed by SVM taking the raw RGB images of the dataset as train and test data, one each to do the same with the LBP images, and one cell to compute the optical flow.
For both SIFT and HOG, the process used is similar. First, descriptors are computed for all the images in the training dataset. These are then clustered using the KMeans clustering algorithm, to reduce the complexity of the calculation, and combine similar features. Here the number of clusters is set to 300. We then repeat the feature extraction process on the train dataset, and assign each obtained feature to the relevant cluster obtained from the KMeans result. This allows to create a histogram for each image, with the number of clusters as the number of bins of the histogram.
We then train the SVM classifier using this histogram as the input, with each image corresponding to a vector of dimension equal to the number of clusters. An identical process is followed for the test dataset, and the resulting histogram is used to predict the labels for the test images. Both the accuracy in terms of the percentage of correct predictions and the confusion matrix for the 10 classes is computed.
For the optical flow, the Lucas Kanede(LK) algorithm is applied. The frames of the input video are considered pairwise, and the key points to be tracked are obtained using the Shi-Tomasi corner detection algorithm. The LK implementation from OpenCV returns the tracked keypoints in the current frame when given the current frame, old frame and the detected key points from the old frame as input. It also returns a status vector corresponding to the old points which signify whether the flow has been detected for these features. Hence, only the points in the current and old frame where the status is 1 or 'True' are tracked. Colored lines are then drawn on these tracked points, which form a continuous track over multiple frames. The key points in the current frame are marked with circles to more easily identify them. The obtained frame is added to the output video. The process is repeated for all frames of the video, with the current frame and key points updated as the old frame and key points.

Q1:

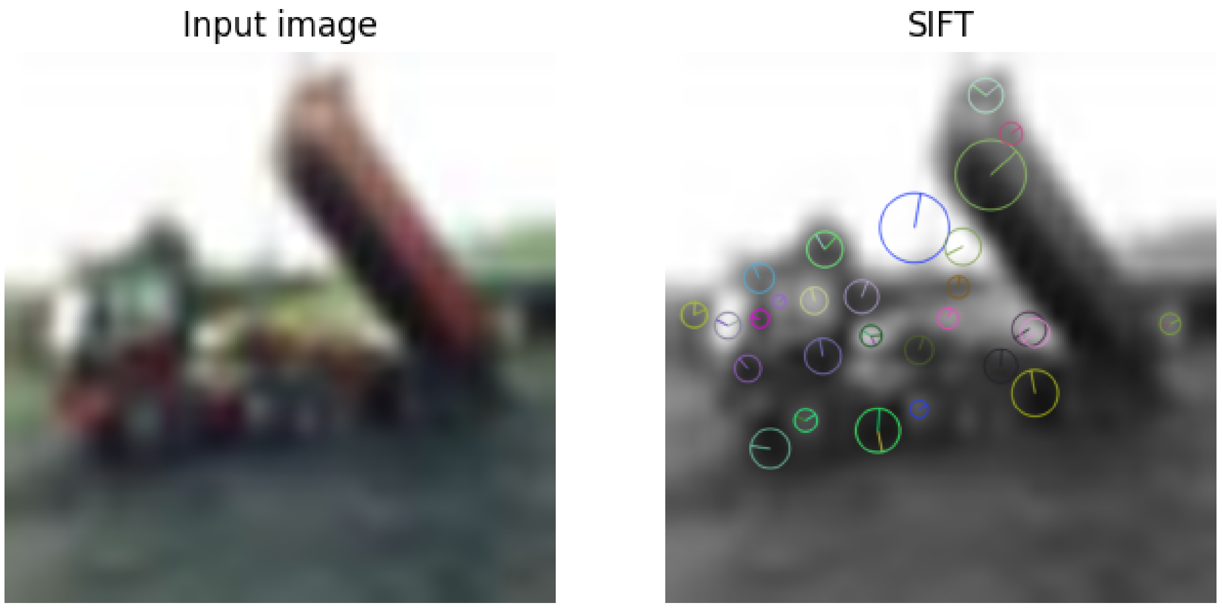
Briefly, the Scale Invariant Feature Transform(SIFT) algorithm as a detector when applied to an image, identifies the key points within that image. It is also used as a descriptor, where it computes a 128-dimensional vector around each keypoint. These functions are combined in the detectAndCompute function of the OpenCV library. The SIFT algorithm consists of the following major steps:

1. Scale space peak selection: The scale space of the image is formed, and extrema points are considered key point candidates.
2. Key point localization: The poorly localized candidates are removed using the Taylor series expansion of the Difference of Gaussian and the Hessian matrix based on the Principal Curvatures.
3. Orientation assignment: To achieve rotation invariance, a weighted direction histogram in the neighborhood of each key point is formed, consisting of 36 bins, and weighted by the gradient magnitudes.
4. Key point descriptor: Finally, the 128-dimensional descriptor around each key point is created by computing the relative orientation and magnitude in a neighborhood around the key point. Then a weighted histogram of the orientation is formed considering the magnitude as the weight.

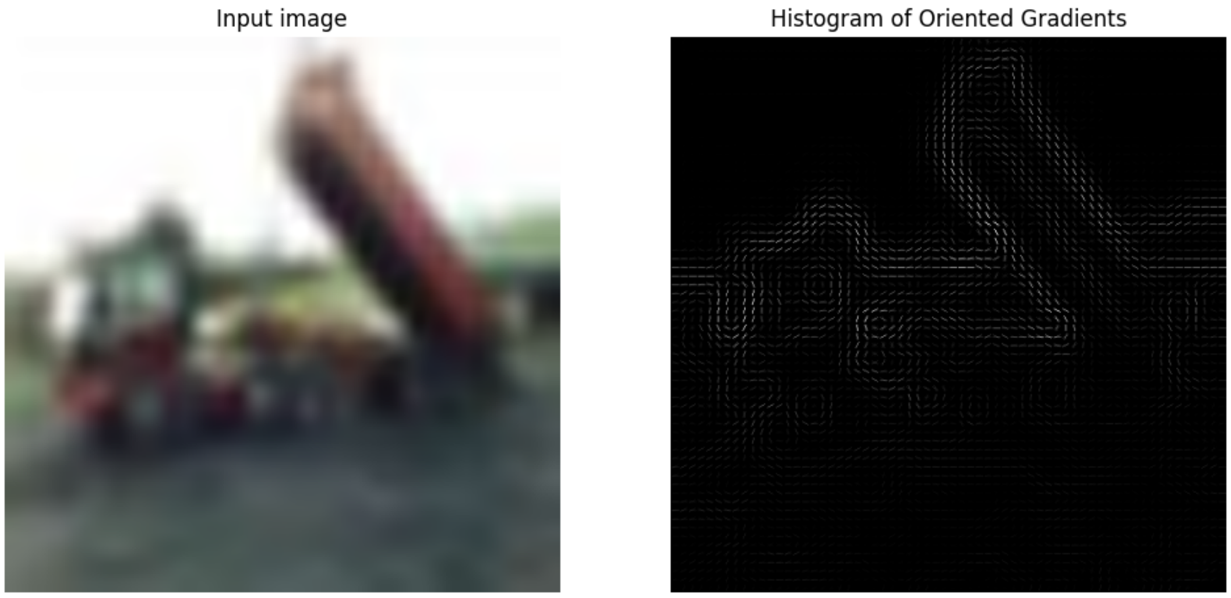
The Histogram of Oriented Gradients(HOG) feature extractor, unlike SIFT, cannot be used to compute the key points in an image. Given an image, it simply returns a feature vector corresponding to the image, considering the entire image, rather than the neighborhood of key points within the image. The original HOG algorithm divided the image into 64×128 windows, and considers 16×16 blocks with 50% overlap, where each block consists of 2×2 cells each of size 8×8. This forms a 3780-dimensional vector considering a histogram with 9 bins. Due to the smaller size of the images used(32×32), we consider each block to have 2×2 pixels, and 1×1 cells. This gives a vector of size $9\times256 = 2304$, where 9 is the number of bins in the histogram and 256 is the total number of blocks formed considering each block has 4 pixels.
The feature vector is derived by first computing the gradient in the X and Y direction for each pixel in the image. Then the gradient magnitude and orientation are computed. These computed matrices are then used to compute the histogram, with the bins of the histogram corresponding to the orientation and the magnitude acting as the vote for each orientation value, the same process used in SIFT. Votes are interpolated linearly between neighboring bin centers.

The features extracted for two sample images of the training set are given below. Both images were resized to 512×512 for better visualization.

SIFT:

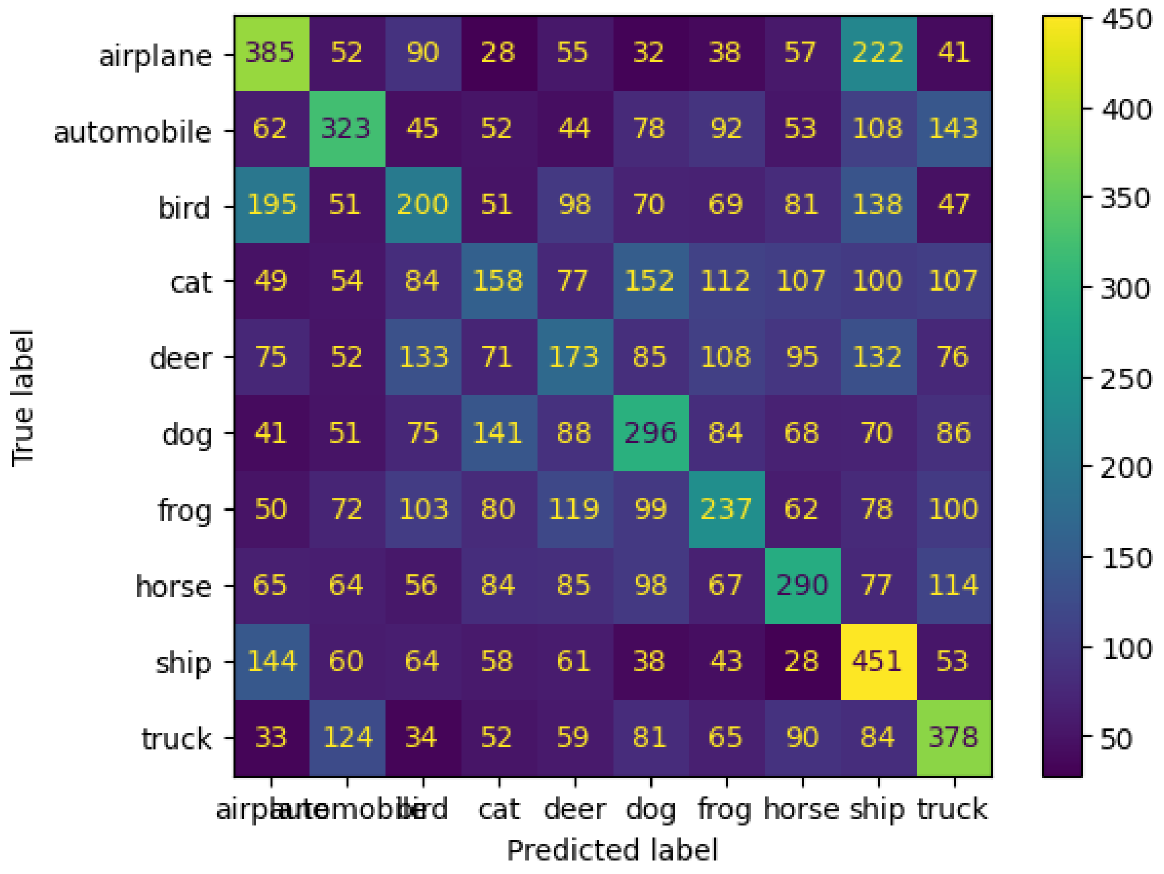

Input image
SIFT

HOG:


Input image
Histogram of Oriented Gradients

The results obtained from the application of **SIFT** followed by classification using SVM are as follows:

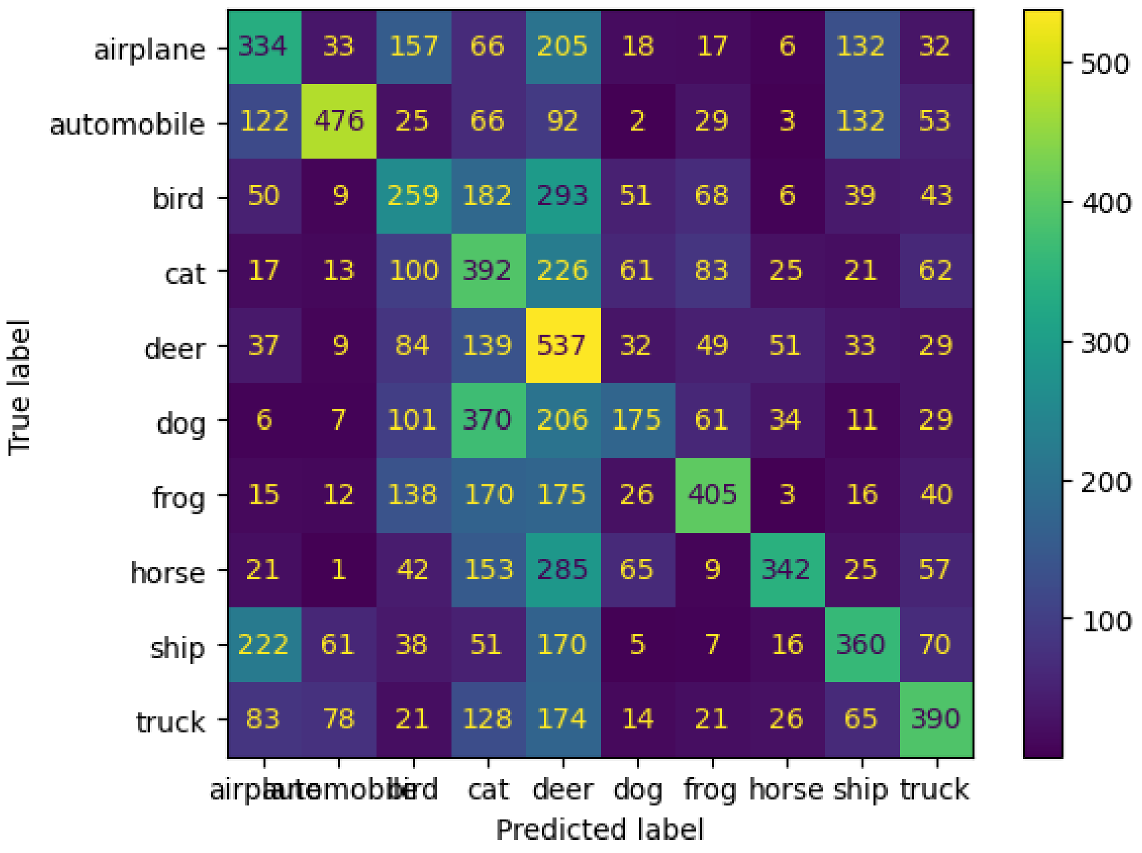
Accuracy: 28.91%
Confusion matrix:



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	385	52	90	28	55	32	38	57	222	41
automobile	62	323	45	52	44	78	92	53	108	143
bird	195	51	200	51	98	70	69	81	138	47
cat	49	54	84	158	77	152	112	107	100	107
deer	75	52	133	71	173	85	108	95	132	76
dog	41	51	75	141	88	296	84	68	70	86
frog	50	72	103	80	119	99	237	62	78	100
horse	65	64	56	84	85	98	67	290	77	114
ship	144	60	64	58	61	38	43	28	451	53
truck	33	124	34	52	59	81	65	90	84	378

The results obtained from the application of **HOG** followed by classification using SVM are as follows:

Accuracy: 36.70%
Confusion matrix:



True label \ Predicted label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	334	33	157	66	205	18	17	6	132	32
automobile	122	476	25	66	92	2	29	3	132	53
bird	50	9	259	182	293	51	68	6	39	43
cat	17	13	100	392	226	61	83	25	21	62
deer	37	9	84	139	537	32	49	51	33	29
dog	6	7	101	370	206	175	61	34	11	29
frog	15	12	138	170	175	26	405	3	16	40
horse	21	1	42	153	285	65	9	342	25	57
ship	222	61	38	51	170	5	7	16	360	70
truck	83	78	21	128	174	14	21	26	65	390

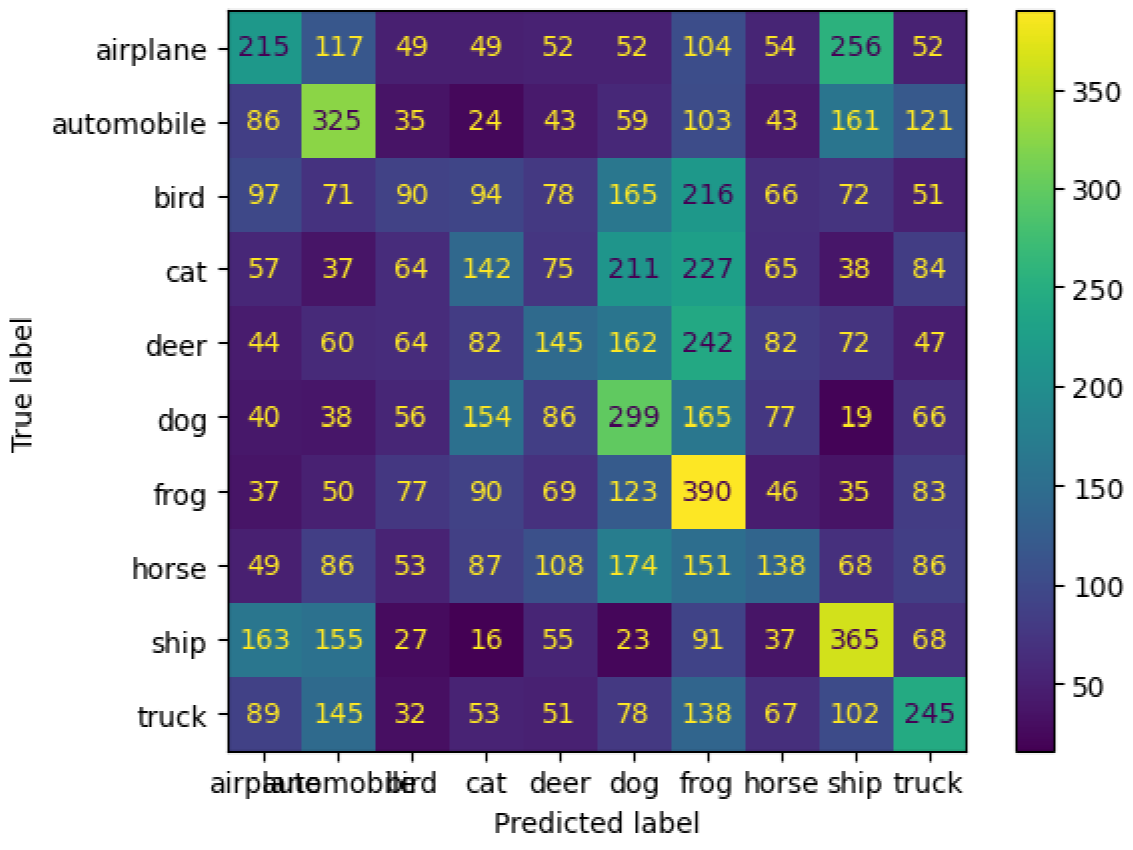
<u>Inference:</u>

In both cases the number of correctly classified samples for each class is greater than the incorrectly classified for another class in the majority of cases. However, there are large variations in the difference between these values. For example, using HOG 537 images belonging to the deer class are correctly classified, with the second most common prediction being cat at 139 samples, whereas for the dog class only 175 images are correctly classified, with 370 images incorrectly classified to the cat class, and 206 to the deer class.
There is also a correlation between the true class and the incorrectly predicted class. In general, vehicles are misclassified as other vehicles more often than as animals and vice versa for animals. This is expected. Another interesting aspect is the degree of this correlation. Using SIFT, the four vehicle classes(airplane, automobile, ship and truck) are correctly classified on average more often than the animal images. This could be attributed to the Gaussian smoothing applied to these images. With the images already at a relatively low resolution, smoothing may blur the boundaries between animal classes that are more likely to share similarities than the vehicle classes. Further, the size of the object within the image is likely more for the vehicle classes than the animal classes, which may contribute to higher accuracy. The absence of smoothing during HOG could explain the higher accuracy, and more even distribution of correctly classified images. However, unlike in SIFT, both the dog and bird classes are misclassified more often than they are correctly classified. Another interesting result is the dominance of one class. Images were classified to the deer class more often than to any other class, with 537 images correctly classified and at least 170 images classified as deer for 8 of the 9 other classes. Only automobiles are classified as deer less than 170 times, namely 92. This may be a consequence of the break_ties argument passed to the SVM, since a similar phenomenon is observed when using SIFT for the ship class.

Q2:

In this question, we use the Local Binary Patterns(LBP) of the images during training and testing instead of the raw RGB images. The LBP algorithm used is the original algorithm which is straightforward. For each pixel in the image,we consider the 3✕3 neighborhood around it. We then compare each of the 8 neighbors of the pixel to the pixel of interest and assign 1 if it is greater and zero otherwise. We then obtain the decimal value of the resulting 8 bit binary vector, considering the most significant bit(MSB) to be the value to the upper left of the central pixel, and the least significant bit(LSB) to be the value to the immediate left of the central pixel, i.e., we start from the upper left corner and move in a clockwise direction till all 8 neighbors are covered. The conversion is done by multiplying the bit value with 2 raised to the bit location in the binary string, with the bit locations starting at 0 at the LSB and increasing by 1 till 7 at the MSB. This value is stored at the coordinates of the pixel of interest in the output image. Since we are considering a 3✕3 neighborhood, the image formed will be of reduced size. Hence, we pad the image by one row or column in all directions to ensure the image size is preserved.
For the comparison to make sense, we convert the image to grayscale before applying the algorithm.

The results obtained from the application of **SIFT** followed by classification using SVM are as follows:
Accuracy: 23.54%
Confusion matrix:

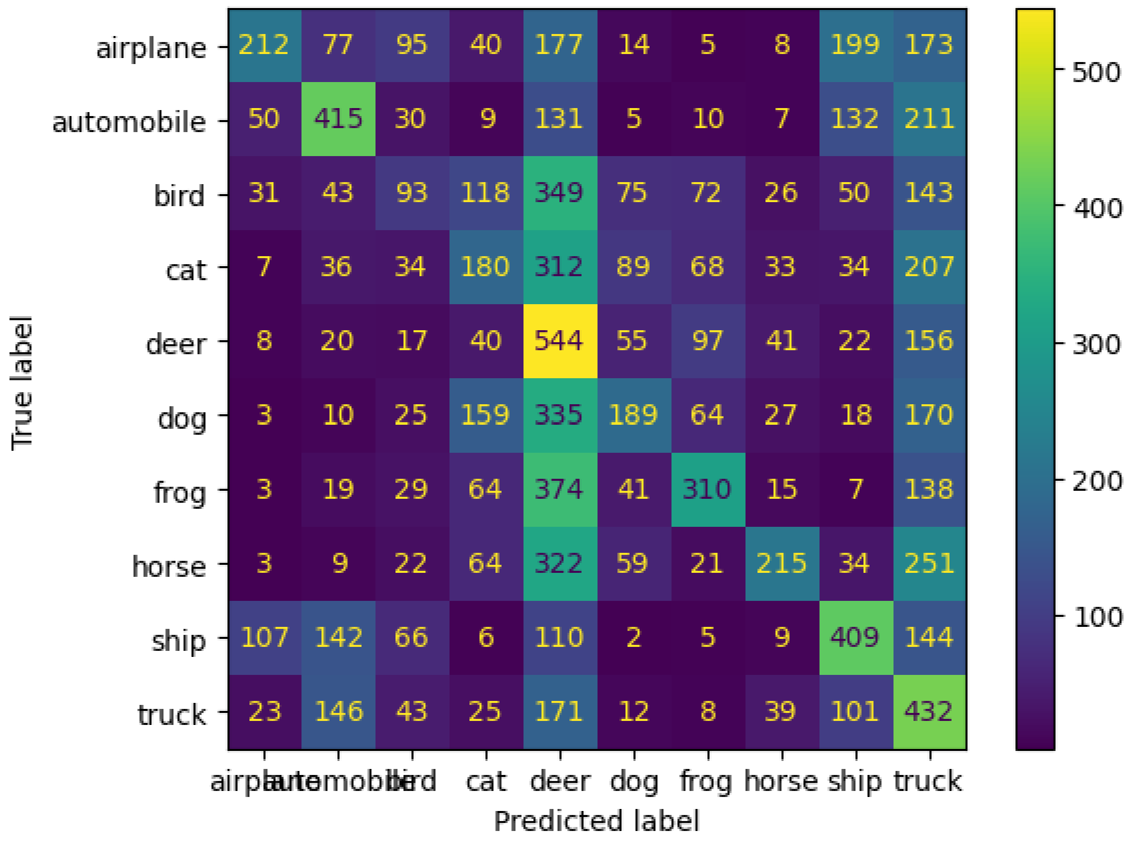


	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	215	117	49	49	52	52	104	54	256	52
automobile	86	325	35	24	43	59	103	43	161	121
bird	97	71	90	94	78	165	216	66	72	51
cat	57	37	64	142	75	211	227	65	38	84
deer	44	60	64	82	145	162	242	82	72	47
dog	40	38	56	154	86	299	165	77	19	66
frog	37	50	77	90	69	123	390	46	35	83
horse	49	86	53	87	108	174	151	138	68	86
ship	163	155	27	16	55	23	91	37	365	68
truck	89	145	32	53	51	78	138	67	102	245

The results obtained from the application of **SIFT** followed by classification using SVM are as follows:

Accuracy: 29.99%

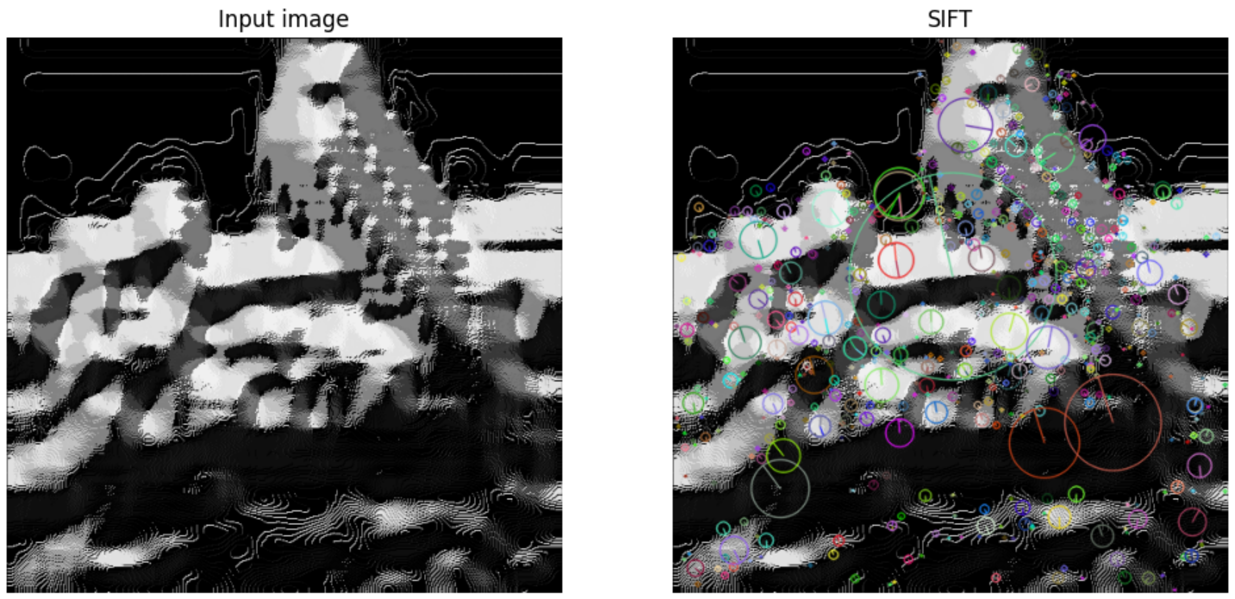
Confusion matrix:



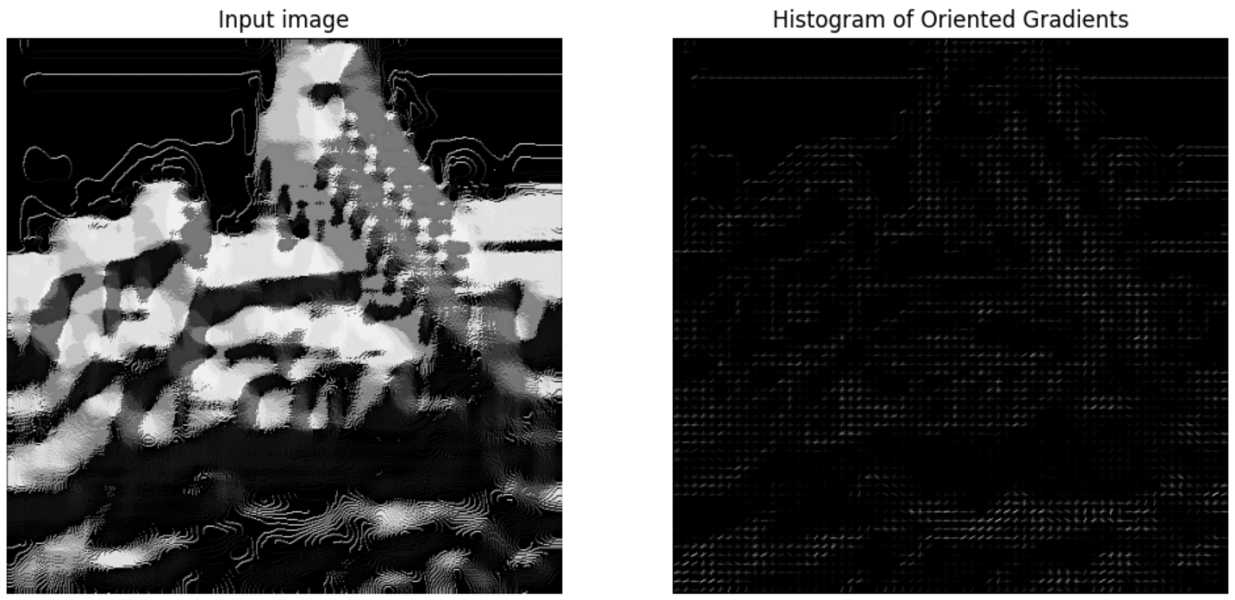
True label \ Predicted label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	212	77	95	40	177	14	5	8	199	173
automobile	50	415	30	9	131	5	10	7	132	211
bird	31	43	93	118	349	75	72	26	50	143
cat	7	36	34	180	312	89	68	33	34	207
deer	8	20	17	40	544	55	97	41	22	156
dog	3	10	25	159	335	189	64	27	18	170
frog	3	19	29	64	374	41	310	15	7	138
horse	3	9	22	64	322	59	21	215	34	251
ship	107	142	66	6	110	2	5	9	409	144
truck	23	146	43	25	171	12	8	39	101	432

Although the results are similar to those obtained in Q1, there is a decrease in the accuracy for both SIFT and HOG. The likely reason for this deviation is the variation in quality of the descriptors obtained during the SIFT and HOG feature extractions. This can be visualized by displaying the key points and descriptors on the LBP images as given below:

SIFT:



HOG:



Clearly, the object in the image is more poorly localized, and the key points obtained by SIFT and the descriptors extracted by both reflect the more noisy image.
Similar observations can be made with regard to the confusion matrix obtained. Again, the vehicle classes are more easily classified than the animal classes, and for HOG, the deer class is predicted more than any other class.

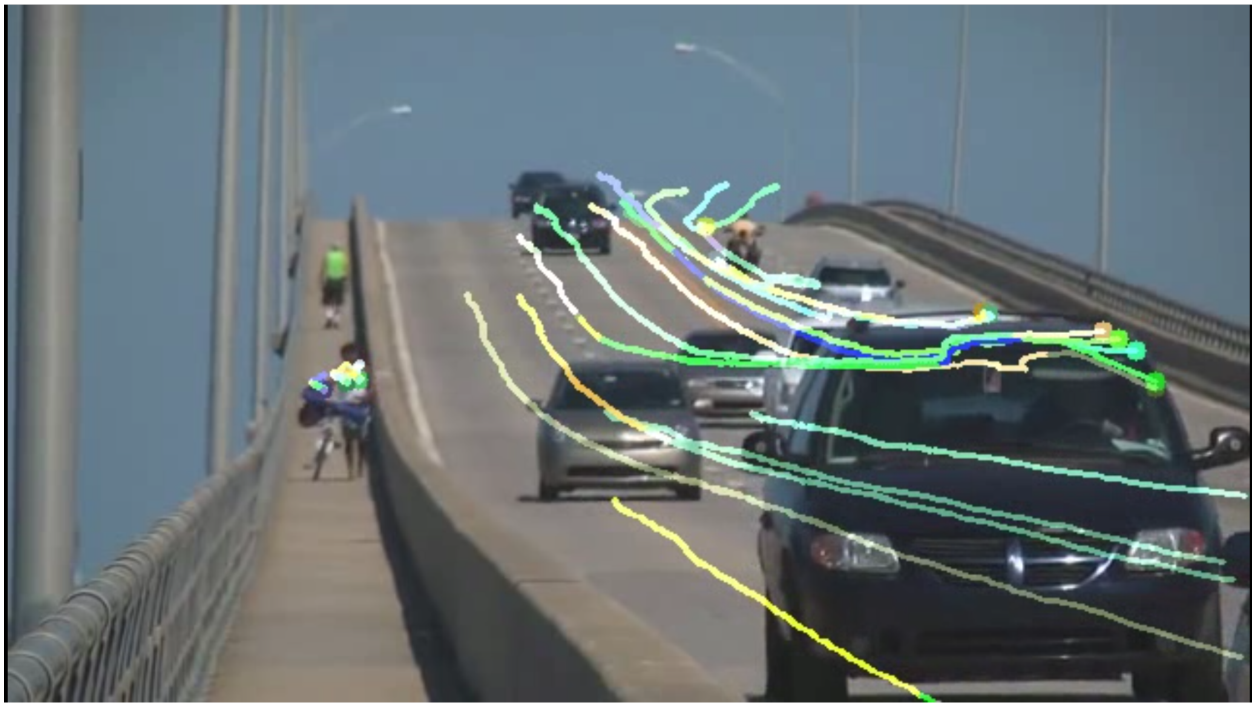
Q3:

Optical flow is a technique used to describe image motion. We can only work on the individual frames that make up a video. Given a frame, optical flow algorithms aim to predict the “velocity” of certain key points within the image, i.e. predict their location in the subsequent frame. The Lucas-Kanade algorithm implemented by OpenCV takes two frames as input, the old frame and current frame, the key points detected in the old frame, the size of the window considered, and the levels of resolution. For all points within the window, the algorithm considers the variation in intensity between the old and current frames to be zero. This allows us to use the gradients in the x, y and t axes to compute the velocity in the x and y directions as follows:

$$I_x(k,l)u + I_y(k,l)v + I_t = 0$$

, where I_x, I_y and I_t are the intensity gradients for each point (k,l) in the window, u and v are the velocities to be predicted in the x and y directions. This system of equations is solved using the least squares method. One possible drawback is the possibility that the assumption of fixed intensity within the window does not hold. In that case, the equation above cannot be used. To combat this, the image is downscaled, with the idea that at lower resolutions any object motion will be negligible, and the assumption will again be valid.

The results obtained from this algorithm were mapped back to the original video and uploaded along with this report and the code used. A screenshot of the video showing the obtained optical flow is given below:



The key points are marked with circles, and the tracks show the movement of these key points between frames.

References:
Optical Flow: https://docs.opencv.org/3.4/d4/dee/tutorial_optical_flow.html
SIFT and HOG: https://github.com/whoisraibolt/BoVF-with-SVM-Classifier