6. A RETRAINED CLASSIFIER – *Te₹₹ency*

This chapter starts with an introduction to the convolutional neural network and TensroFlow. Next, it discusses a CNN and TensorFlow based, implemented and retrained model *Te₹₹ency*. Finally, it shows a detailed performance analysis of *Te₹₹ency*.

6.1 INTRODUCTION TO THE CONVOLUTIONAL NEURAL NETWORKS (CNN) AND TENSORFLOW

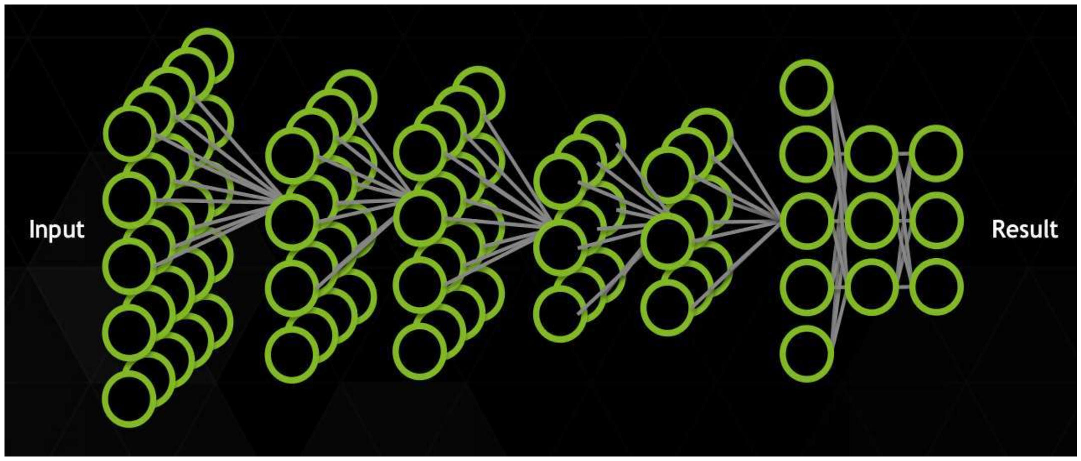


Figure 6.1. A deep neural network

Deep neural networks (DNNs) trained via back propagation have been proved promising in image classification with some millions of images and thousands of categories. These networks achieve the best performance in classification. It has capabilities to work with ample of various datasets. This potential makes DNNs more generic for learning. Convolutional Neural Networks are special types of deep neural networks used for image classification. These neural networks normally have millions of parameters with 10+ million images, and at least 10+ layers. This makes them deep with convolution layers, pooling layers, and fully connected layers. It learns from a higher level feature hierarchy and propagates it to the next layer. Here, the convolution is a technique to mix two functions or information, and pooling is a procedure to reduce multiple values from a region and convert it into a single value. Each CNN may have its architecture. The following figure shows the architecture of one of the CNN.

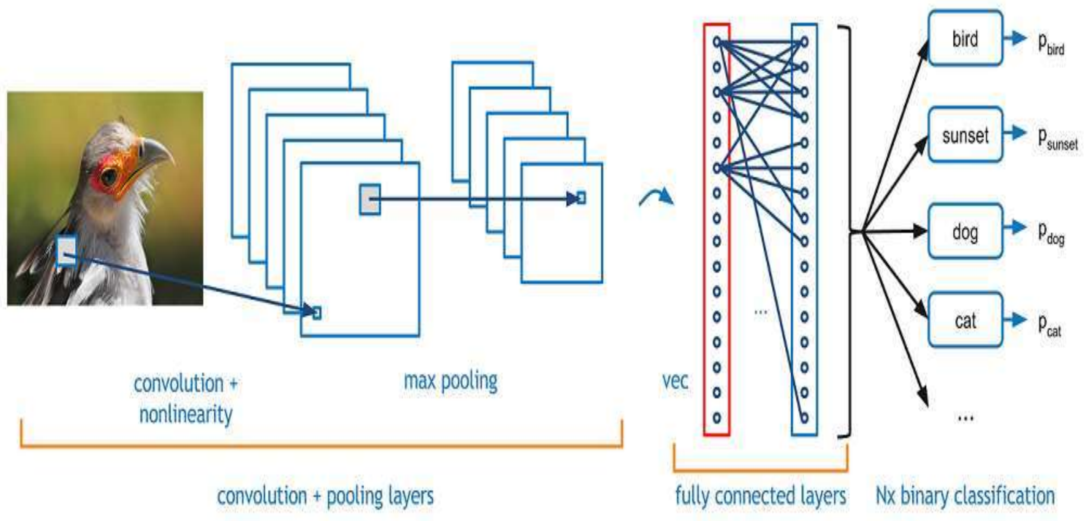


Figure 6.2. The architecture of a CNN

These CNN's are being used in a technology called TensorFlow for image classification. TensorFlow has been developed by the Google Brain team and written in Python, CUDA, and C++. It is an open-source machine learning library, for data flow programming. The motive behind the development of this framework was to support the high-performance numerical computing for CPUs, GPUs and Tensor Processing Units (TPUs). It is nowadays extensively used for the machine learning applications. Its initial release was launched in November 2015, but the stable version has been launched and available from April 2018.

TensorFlow uses a data flow graph to represent the computation in terms of the dependencies between individual operations which leads to a low-level programming model. Here, first, the data flow graph is defined and, later a TensorFlow session is created to execute the parts of the graph. The graph is described in terms of Nodes, Edges, and Operation. The nodes are responsible for performing the computations and may have zero or more input and outputs. The data values that moves between inputs and outputs are known as tensors. The tensor is nothing but a multidimensional array of floating point values. The edges are used to describe the flow of tensors in the graphs. The operation is an abstract computation which takes input and gives attributes as the output.

TensorFlow is known as a transfer learning technique. Transfer learning or inductive transfer is a machine learning technique which allows storing of knowledge obtained during one problem solving and using it as an initial point to solve the other relevant problem. For example, the knowledge gained while learning to classify the buses could be used to classify the cars. For Transfer learning, there are two approaches used: build the model or re-use the model. In the first approach, based on the source task, a source model is prepared and may be reused for a similar task in future. Many times the new model is also required to be tuned. In the second approach of pre-trained models, an existing model is selected as a model, and if it suits the task, then that model can be retrained and tuned, if needed. The following figure shows the basic difference between traditional machine learning and transfer learning.

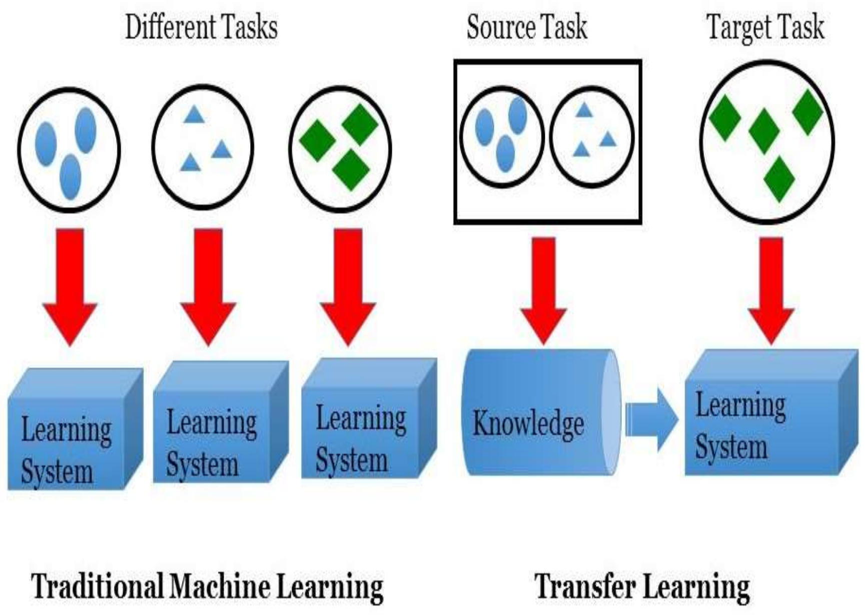
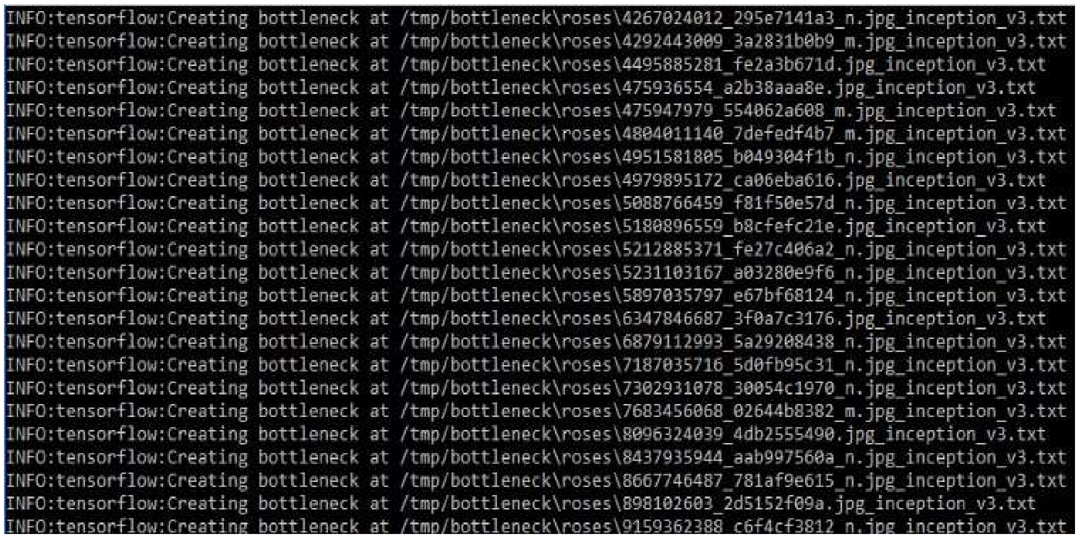


Figure 6.3. Traditional Machine Learning Vs. Transfer Learning

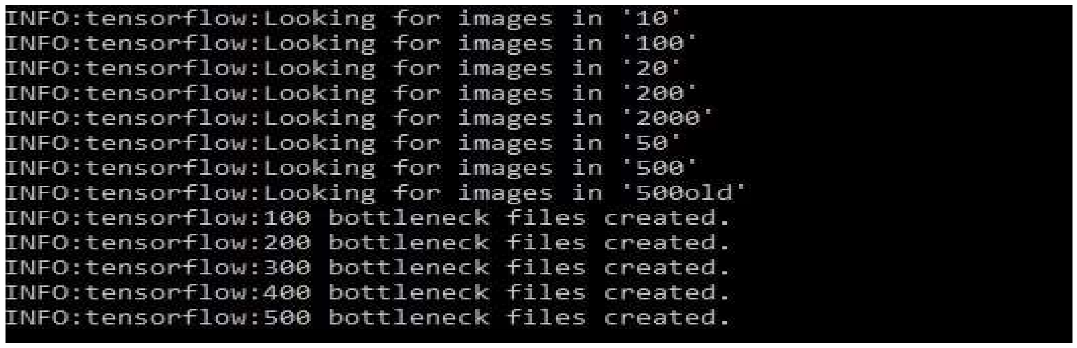
6.2 *Te₹₹ency* – A TENSORFLOW AND CNN BASED RETRAINED MODEL

Te₹₹ency is a TensorFlow based CNN model which has been created specifically for the Indian currency identification. For *Te₹₹*ency, the Inception-v3 model has been used for training. It is a model trained for ImageNet challenge 2012 with more than 2000 classes. In CNNs, the inception modules are used to support deeper and larger convolution layers. They simultaneously efficiently allow the computation. For example, 192 28×28 sized feature maps can be reduced to 64 28×28 feature maps through 64 1×1 convolutions. That means, at the end of inception module, all the large convolutions are concatenated into a larger feature map and fed into the next layer. The prediction accuracy of these CNN models is tested based on criteria called “top-5 error rate”. The Inception-v3 model ranks 5th with error-rate of 3.46% as compared to the GoogLeNet model with 15.3% error rate.

In the training or retraining of a CNN based Tensor model, the very first phase is the bottlenecks creation. It is the layer that does or helps the classification. The following figure shows the bottleneck creation during retraining of the model, *Te₹₹ency*.



```
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\4267024012_295e7141a3_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\4292443009_3a2831b0b9_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\4495885281_fe2a3b671d.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\475936554_a2b38aaa8e.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\475947979_554062a608_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\4804011140_7defedf4b7_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\4951581805_b049304f1b_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\4979895172_ca06eba616.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\5088766459_f81f50e57d_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\5180896559_b8cfefc21e.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\5212885371_fe27c406a2_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\5231103167_a03280e9f6_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\5897035797_e67bf68124_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\6347846687_3f0a7c3176.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\6879112993_5a29208438_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\7187035716_5d0fb95c31_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\7302931078_30054c1970_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\7683456068_02644b8382_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\8096324039_4db2555490.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\8437935944_aab997560a_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\8667746487_781af9e615_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\898102603_2d5152f09a.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\roses\9159362388_c6f4cf3812_n.jpg_inception_v3.txt
```

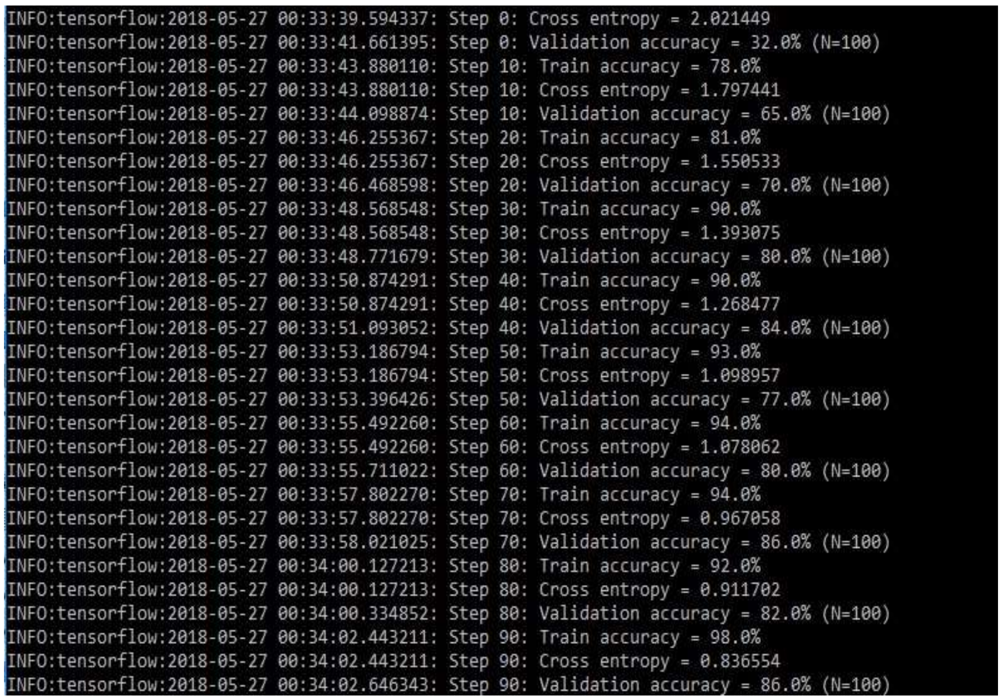


```
INFO:tensorflow:Looking for images in '10'
INFO:tensorflow:Looking for images in '100'
INFO:tensorflow:Looking for images in '20'
INFO:tensorflow:Looking for images in '200'
INFO:tensorflow:Looking for images in '2000'
INFO:tensorflow:Looking for images in '50'
INFO:tensorflow:Looking for images in '500'
INFO:tensorflow:Looking for images in '500old'
INFO:tensorflow:100 bottleneck files created.
INFO:tensorflow:200 bottleneck files created.
INFO:tensorflow:300 bottleneck files created.
INFO:tensorflow:400 bottleneck files created.
INFO:tensorflow:500 bottleneck files created.
```

Figure 6.4. The bottlenecks creation

Here, the term bottleneck does not mean it may create congestion in the network. It means near to the output and compact in shape in the network. For every single image in training, the bottleneck is created and consumes significant time in calculating the layers behind the bottlenecks. So, if the training is done for the first time, it may take considerable time from 30-60 minutes to weeks for the bottleneck creation depending on the number of images in the dataset.

Once the bottlenecks are created, and if retraining is carried out, the same bottlenecks are used and save the time in retraining. The actual training starts after the creation of the bottlenecks. By default, for 4000 iterations, each iteration selects ten images (this can be set) randomly from the training set. Their appropriate bottlenecks are taken from cache and fed into the final layer for predictions. These predictions are compared with the actual labels, and the result is backpropagated to the final layer to update the weights. The following figure shows training accuracy, validation accuracy, and cross entropy calculation.



```
INFO:tensorflow:2018-05-27 00:33:39.594337: Step 0: Cross entropy = 2.021449
INFO:tensorflow:2018-05-27 00:33:41.661395: Step 0: Validation accuracy = 32.0% (N=100)
INFO:tensorflow:2018-05-27 00:33:43.880110: Step 10: Train accuracy = 78.0%
INFO:tensorflow:2018-05-27 00:33:43.880110: Step 10: Cross entropy = 1.797441
INFO:tensorflow:2018-05-27 00:33:44.098874: Step 10: Validation accuracy = 65.0% (N=100)
INFO:tensorflow:2018-05-27 00:33:46.255367: Step 20: Train accuracy = 81.0%
INFO:tensorflow:2018-05-27 00:33:46.255367: Step 20: Cross entropy = 1.550533
INFO:tensorflow:2018-05-27 00:33:46.468598: Step 20: Validation accuracy = 70.0% (N=100)
INFO:tensorflow:2018-05-27 00:33:48.568548: Step 30: Train accuracy = 90.0%
INFO:tensorflow:2018-05-27 00:33:48.568548: Step 30: Cross entropy = 1.393075
INFO:tensorflow:2018-05-27 00:33:48.771679: Step 30: Validation accuracy = 80.0% (N=100)
INFO:tensorflow:2018-05-27 00:33:50.874291: Step 40: Train accuracy = 90.0%
INFO:tensorflow:2018-05-27 00:33:50.874291: Step 40: Cross entropy = 1.268477
INFO:tensorflow:2018-05-27 00:33:51.093052: Step 40: Validation accuracy = 84.0% (N=100)
INFO:tensorflow:2018-05-27 00:33:53.186794: Step 50: Train accuracy = 93.0%
INFO:tensorflow:2018-05-27 00:33:53.186794: Step 50: Cross entropy = 1.098957
INFO:tensorflow:2018-05-27 00:33:53.396426: Step 50: Validation accuracy = 77.0% (N=100)
INFO:tensorflow:2018-05-27 00:33:55.492260: Step 60: Train accuracy = 94.0%
INFO:tensorflow:2018-05-27 00:33:55.492260: Step 60: Cross entropy = 1.078062
INFO:tensorflow:2018-05-27 00:33:55.711022: Step 60: Validation accuracy = 80.0% (N=100)
INFO:tensorflow:2018-05-27 00:33:57.802270: Step 70: Train accuracy = 94.0%
INFO:tensorflow:2018-05-27 00:33:57.802270: Step 70: Cross entropy = 0.967058
INFO:tensorflow:2018-05-27 00:33:58.021025: Step 70: Validation accuracy = 86.0% (N=100)
INFO:tensorflow:2018-05-27 00:34:00.127213: Step 80: Train accuracy = 92.0%
INFO:tensorflow:2018-05-27 00:34:00.127213: Step 80: Cross entropy = 0.911702
INFO:tensorflow:2018-05-27 00:34:00.334852: Step 80: Validation accuracy = 82.0% (N=100)
INFO:tensorflow:2018-05-27 00:34:02.443211: Step 90: Train accuracy = 98.0%
INFO:tensorflow:2018-05-27 00:34:02.443211: Step 90: Cross entropy = 0.836554
INFO:tensorflow:2018-05-27 00:34:02.646343: Step 90: Validation accuracy = 86.0% (N=100)
```

Figure 6.5. Training accuracy, validation accuracy, and cross entropy calculation

Following list defines the terms briefly:

i. The training accuracy: It shows how many images that have been used in current training batch are correctly classified. It is represented in percentage.

ii. Validation accuracy: It is the precision (percentage of correctly-classified images) on a randomly-selected group of images from a different set. Validation accuracy is the true measure of accuracy for the network.

iii. Cross-entropy: To understand cross entropy, let us first understand entropy. Entropy is the extent of randomness in the data or information which is being processed. Cross-entropy is a loss function that gives an idea about the progress of the learning process. (Lower numbers are better).

Apart from the iterations, another parameter that may affect the performance of training is the learning rate. Decreasing or increasing the learning rate affects the performance. Here, that default value of learning rate 0.01 has been used in our model. In addition to this, the resolution of the input image also affects the classification. If image resolution is higher, it may take more time for training, but it may give higher accuracy. The relative size of the model can also be set. These parameters are set during the configuration of the convolution network. The steps for creating *Te₹₹ency* are given below:

```
1. Prepare the dataset for training images.
2. Set the number of iterations, learning rate, the relative size of the model, input image
   resolution, etc.
3. (Re)train the network:
      a. Configure the convolution network.
      b. Start training, actually retraining:
            i. Bottleneck creation.
           ii. Training accuracy, validation accuracy, and cross entropy calculation.
4. Export the model.
5. Test the classifier.
```

TensorFlow based Retraining and Testing Algorithm: *Te₹₹ency*

6.3 THE TESTING AND PERFORMANCE ANALYSIS OF *Te₹₹ency*

As stated in section 6.2, *Te₹₹ency* is based on training of TensorFlow model for a specific number of iterations. The following table 6.1 shows the time taken by the *Te₹₹ency* model for a different number of iterations. During the training of the network, bottlenecks are created to train the model. As stated in the previous section, these bottlenecks are created only once for the same model for the same set of images. Hence, the training time would always be significantly high for the first time. For the successive training, at the same learning rate (here 0.01), the training time would be quite less than for the first time training. The *Te₹₹ency* has been trained for a different number of iterations starting from 100 to 500, 1000, 2000 and 4000.

Denomination	# of Train Images	Training Time for # of iterations (Seconds)				
		100	500	1000	2000	4000
5	400	2610.44*	205.40	260.03	484.18	969.01
10	620					
20	580					
50	540					
100	660					
200	352					
500	216					
500_old	360					
1000	84					
2000	216					

* Due to bottlenecks creation for the first time

Table 6.1 Time taken to train the *Te₹₹ency* Model for different number of iterations in seconds for the learning rate 0.01

The following subsections discuss the performance analysis of the *Te₹₹ency* with reference to the *HORBoVF*'s results for K=24.

6.3.1 For Iterations=100

The following table shows the performance of the *Te₹₹ency* for 100 iterations. Here, F indicates the fully visible images whereas P indicates the partially visible images.

Denomination	# of Test Images		# of Correct Classes		Execution Time (Seconds)		Classification Accuracy (%)	
	F	P	F	P	F	P	F	P
5	130	80	114	54	414.40	140.53	87.69	67.5
10	226	80	197	49	413.98	140.48	87.17	61.25
20	220	80	197	47	414.40	140.53	89.55	58.75
50	228	80	198	51	412.06	139.98	86.84	63.75
100	268	80	234	49	415.66	139.65	87.31	61.25
200	139	52	116	31	411.39	140.32	83.45	59.62
500	160	78	142	52	417.89	140.87	88.75	66.667
500_old	165	80	137	47	416.67	140.23	83.03	58.75
1000	83	80	68	45	412.39	140.69	81.93	56.25
2000	200	80	174	51	414.40	141.03	87.00	63.75
Average time taken for an image & Accuracy					**2.2777**	**1.8237**	**86.69**	**61.81**
Overall Accuracy =79.2970%					**Average Time = 2.1427 Seconds**			

Table 6.2 The *Te₹₹ency* Performance for training iterations=100

For 100 iterations, in comparison to 2370 out of 2589 images for the *HORBoVF*, the *Te₹₹ency* identifies 2053 images correctly giving an overall accuracy of 79.297% as compared to the *HORBoVF*'s 91.541%. Here, for the fully visible images, the accuracy is 86.695%, and the same for the partially visible images is 61.818% as compared to 98.351% and 75.454% of the *HORBoVF* respectively. The total number of correctly classified images and its accuracy are shown in the following figure 6.6 and figure 6.7 respectively.

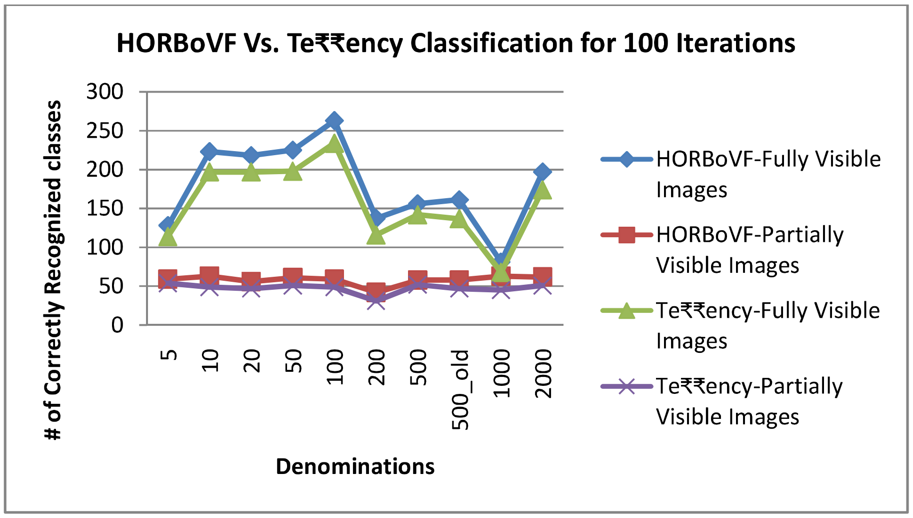


Figure 6.6. Number of correctly identified images using the *HORBoVF* and the *Te₹₹ency*

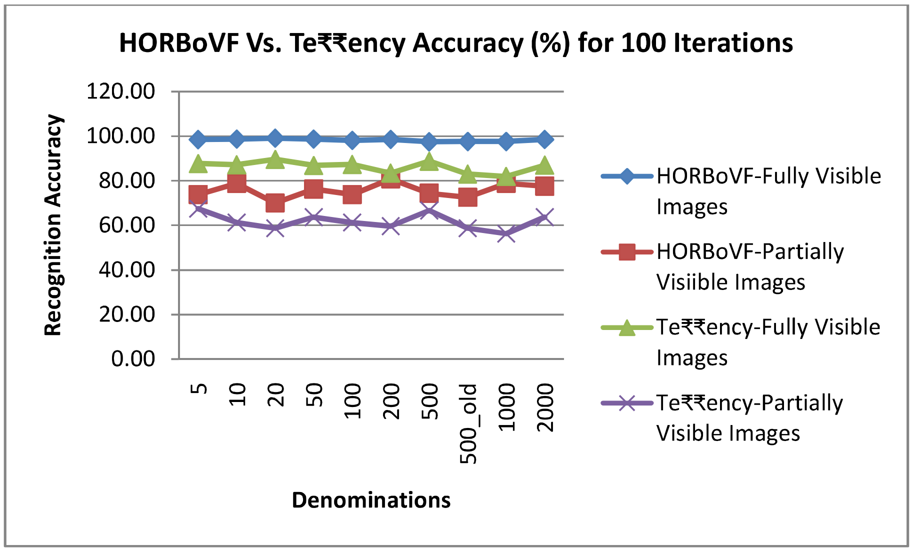


Figure 6.7. Recognition accuracy of the *HORBoVF* and the *Te₹₹ency*

Here, for 100 iterations, the algorithm takes 2.277 seconds and 1.823 seconds to label the images of the type fully and partially visible images, as visible in figure 6.8, for which the *HORBoVF* takes 0.117 seconds for both cases.

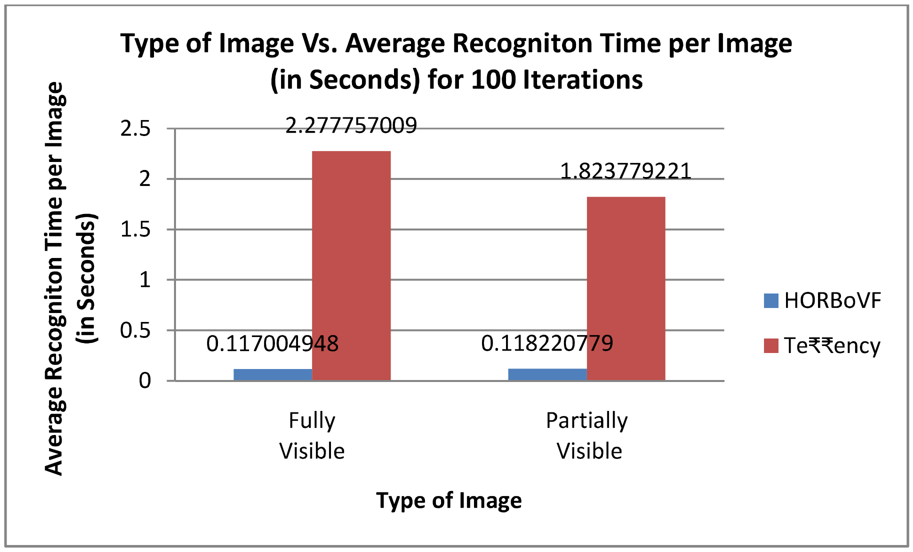


Figure 6.8. Average time taken per image by the *HORBoVF* and the *Te₹₹ency*

The overall performance is shown in figure 6.9 proving the *HORBoVF* a better approach.

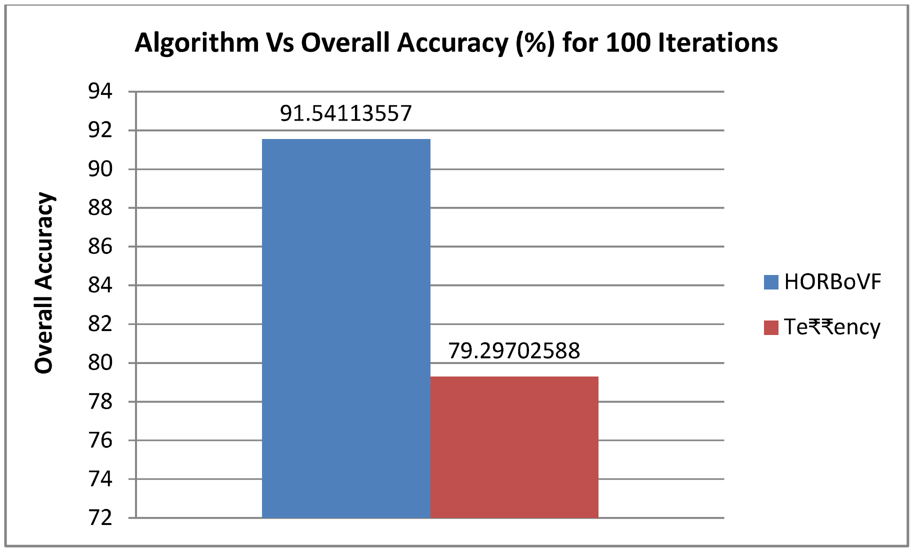


Figure 6.9. The overall performance of the *HORBoVF* and the *Te₹₹ency*

6.3.2 For Iterations=500

The following table shows the performance of the *Te₹₹ency* for 500 iterations. For 500 iterations, in comparison to 2370 out of 2589 images for the *HORBoVF*, the *Te₹₹ency* identifies 2113 images correctly giving an overall accuracy of 81.614% as compared to the *HORBoVF*'s 91.541%. Here, for the fully visible images, the accuracy is 89.389%, and the same for the partially visible images is 63.246% as compared to 98.351% and 75.454% of the *HORBoVF* respectively.

Denomination	**# of Test Images**		**# of Correct Classes**		**Execution Time (Seconds)**		**Classification Accuracy (%)**	
	F	**P**	**F**	**P**	**F**	**P**	**F**	**P**
5	130	80	119	54	405.68	153.68	91.54	67.5
10	226	80	199	49	412.38	149.23	88.05	61.25
20	220	80	199	47	416.32	146.98	90.45	58.75
50	228	80	203	52	414.78	141.06	89.04	65
100	268	80	239	49	414.23	141.16	89.18	61.25
200	139	52	123	32	413.80	139.77	88.49	61.54
500	160	78	149	52	416.56	154.23	93.13	66.667
500_old	165	80	141	54	414.88	140.64	85.45	67.5
1000	83	80	72	47	413.79	139.86	86.75	58.75
2000	200	80	182	51	415.08	141.65	91.00	63.75
Average time taken for an image & Accuracy					**2.2746**	**1.8808**	**89.38**	**63.24**
Overall Accuracy = 81.6145%					**Average Time = 2.1574 Seconds**			

Table 6.3 The *Te₹₹ency* Performance for training iterations=500

The total number of correctly classified images and its accuracy are shown in the following figure 6.10 and figure 6.11 respectively.

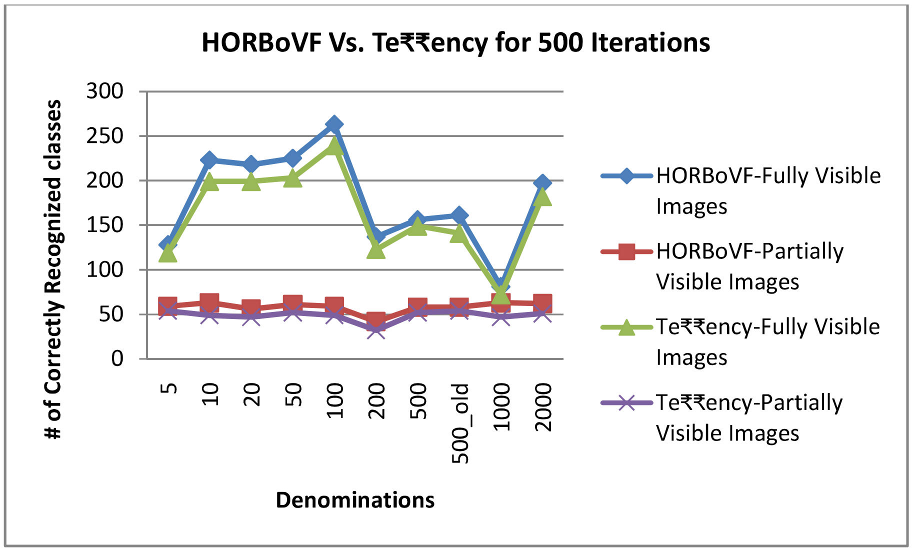


Figure 6.10. Number of correctly identified images using the *HORBoVF* and the *Te₹₹ency*

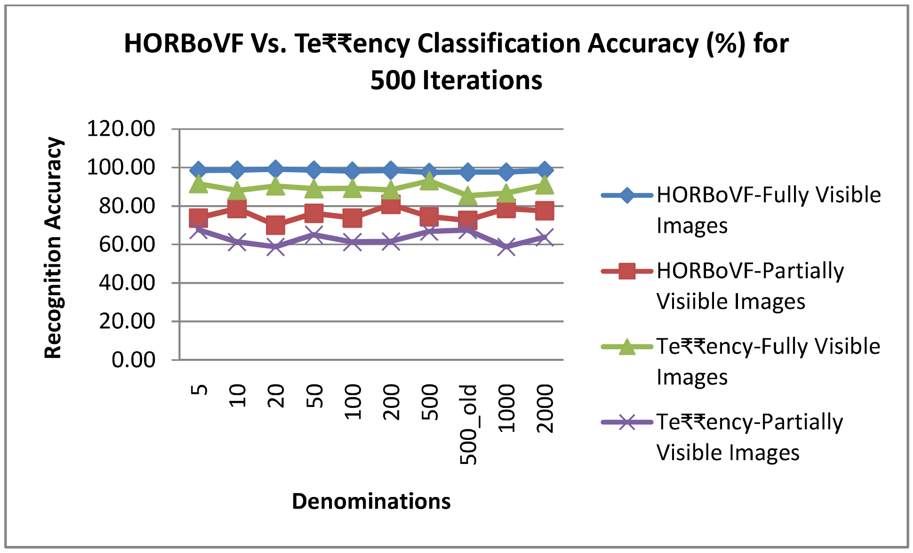


Figure 6.11. Recognition accuracy of the *HORBoVF* and the *Te₹₹ency*

Here for 500 iterations, the algorithm takes 2.274 seconds and 1.88 seconds to label the images of the type fully and partially visible images, as visible in figure 6.12, which the *HORBoVF* takes 0.117 seconds for both cases. It can be observed that the labeling time remains almost intact irrespective of the number of iterations.

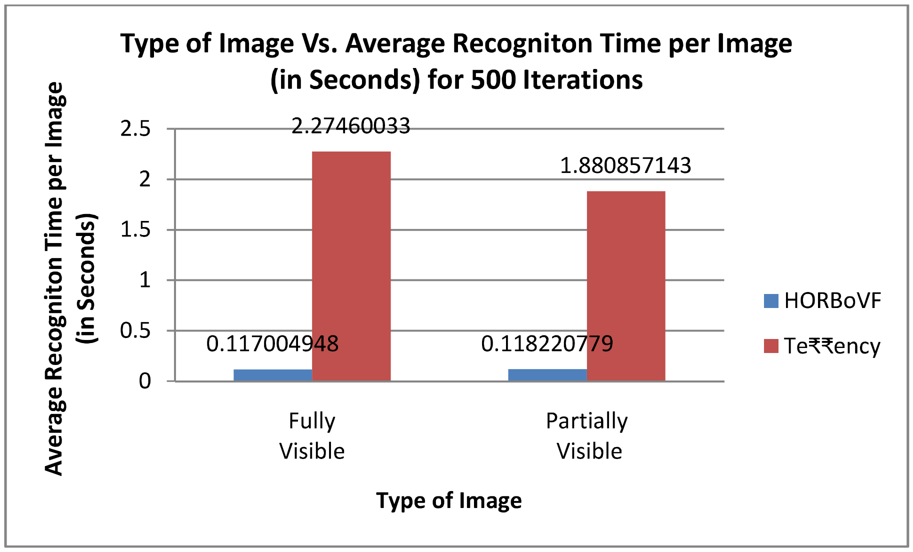


Figure 6.12. Average time taken per image by the *HORBoVF* and the *Te₹₹ency*

The overall performance is shown in figure 6.13.

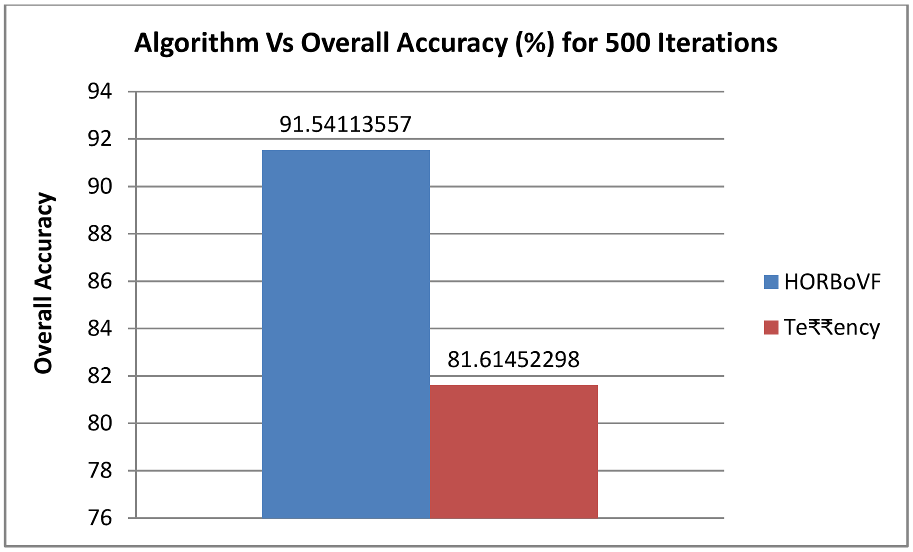


Figure 6.13. The overall performance of the *HORBoVF* and the *Te₹₹ency*

6.3.3 For Iterations=1000

The following table shows the performance of the *Te₹₹ency* for 1000 iterations.

Denomination	# of Test Images		# of Correct Classes		Execution Time (Seconds)		Classification Accuracy (%)	
	F	P	F	P	F	P	F	P
5	130	80	121	56	414.56	159.36	93.08	70
10	226	80	199	49	410.89	154.23	88.05	61.25
20	220	80	201	49	409.87	149.78	91.36	61.25
50	228	80	210	54	411.65	139.32	92.11	67.5
100	268	80	239	49	416.38	140.65	89.18	61.25
200	139	52	127	35	414.78	147.69	91.37	67.31
500	160	78	149	54	419.06	143.65	93.13	69.231
500_old	165	80	146	59	418.95	141.23	88.48	73.75
1000	83	80	75	51	414.23	142.93	90.36	63.75
2000	200	80	184	53	412.06	140.05	92.00	66.25
Average time taken for an image & Accuracy					**2.2773**	**1.8946**	**90.76**	**66.10**
Overall Accuracy = 83.4299%					**Average Time = 2.1635 Seconds**			

Table 6.4 The *Te₹₹ency* Performance for training iterations=1000

For 1000 iterations, in comparison to 2370 out of 2589 images for the *HORBoVF*, the *Te₹₹ency* identifies 2160 images correctly giving an overall accuracy of 83.429% as compared to *HORBoVF*'s 91.541%. Here, for the fully visible images, the accuracy is 90.764%, and the same for the partially visible images is 66.103% as compared to 98.351% and 75.454% of the *HORBoVF* respectively. The total number of correctly classified images and its accuracy are shown in the following figure 6.14 and figure 6.15 respectively.

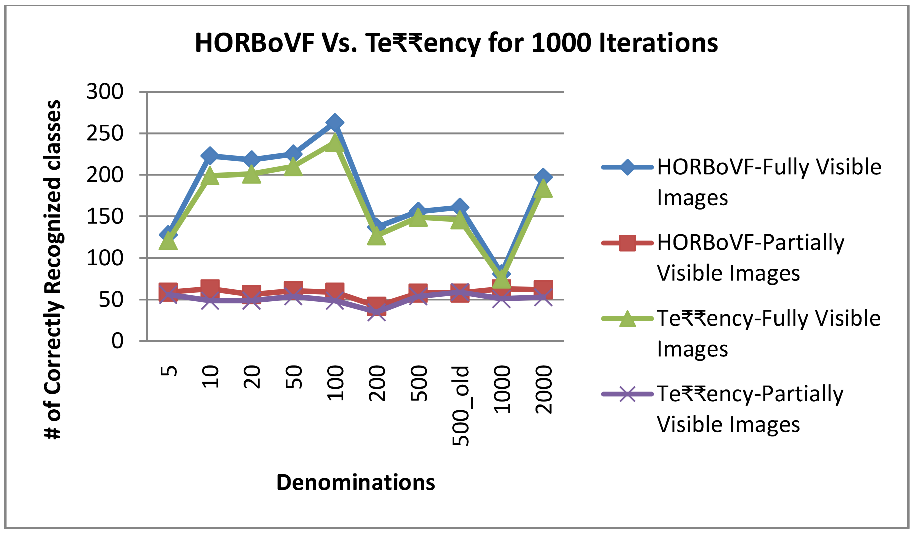


Figure 6.14. Number of correctly identified images using the *HORBoVF* and the *Te₹₹ency*

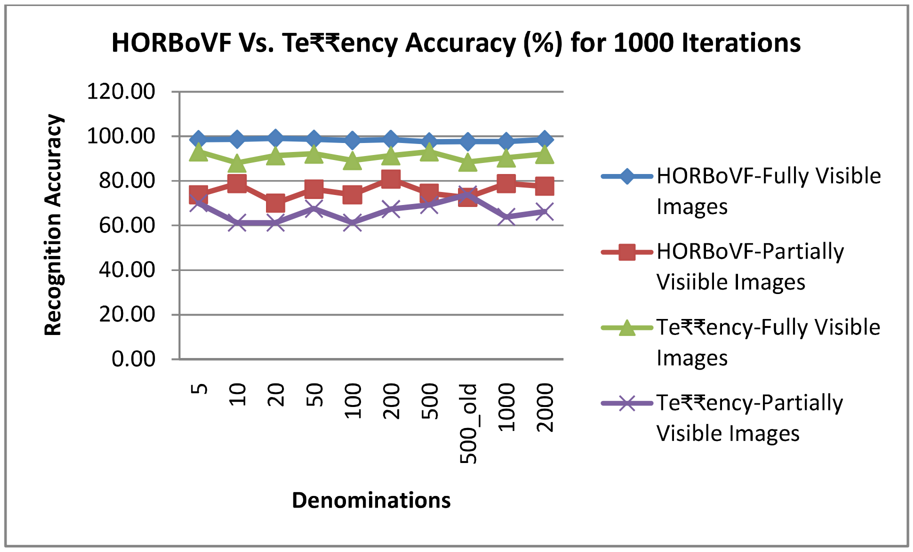


Figure 6.15. Recognition accuracy of the *HORBoVF* and the *Te₹₹ency*

Here for 1000 iterations, the algorithm takes 2.277 seconds and 1.894 seconds to label the images of the type fully and partially visible images, as visible in figure 6.16, which the *HORBoVF* takes 0.117 seconds for both cases. Here also, it can be observed that the labeling time remains almost intact irrespective of the number of iterations.

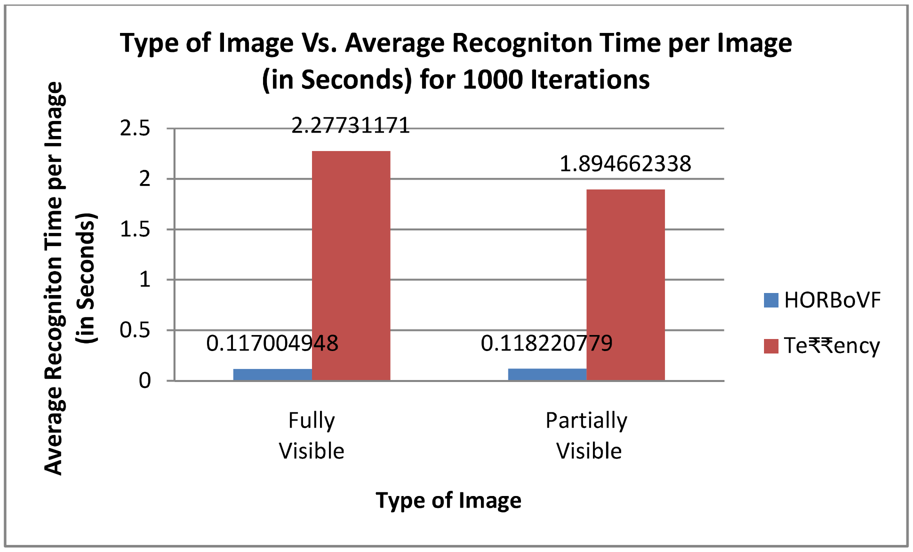


Figure 6.16. Average time taken per image by the *HORBoVF* and the *Te₹₹ency*

The overall performance is shown in figure 6.17.

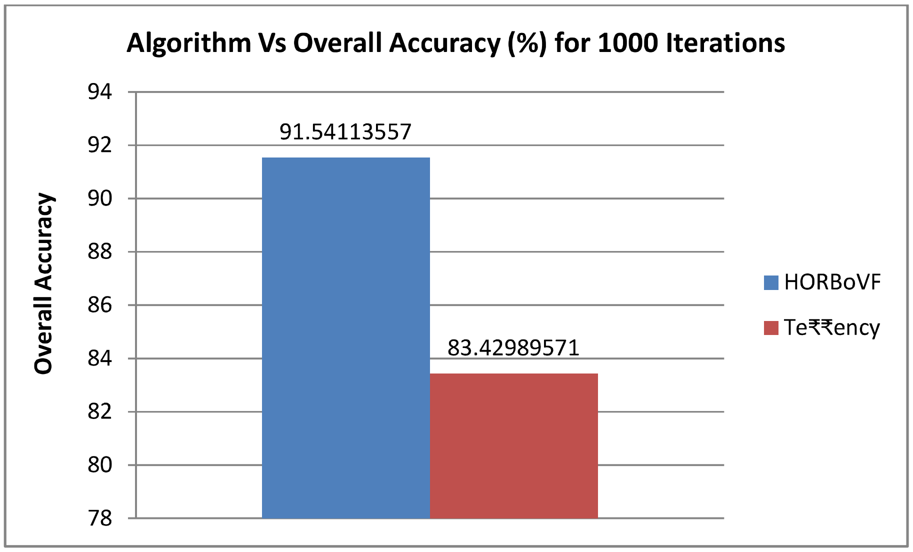


Figure 6.17. The overall performance of the *HORBoVF* and the *Te₹₹ency*

6.3.4 For Iterations=2000

The following table shows the performance of the *Te₹₹ency* for 2000 iterations. For 2000 iterations, in comparison to 2370 out of 2589 images for the *HORBoVF*, the *Te₹₹ency* identifies 2223 images correctly giving an overall accuracy of 85.863% as compared to the *HORBoVF*'s 91.541%. Here, for the fully visible images, the accuracy is 92.193%, and the same for the partially visible images is 70.909% as compared to 98.351% and 75.454% of the *HORBoVF* respectively.

Denomination	**# of Test Images**		**# of Correct Classes**		**Execution Time (Seconds)**		**Classification Accuracy (%)**	
	F	**P**	**F**	**P**	**F**	**P**	**F**	**P**
5	130	80	123	59	413.69	151.23	94.62	73.75
10	226	80	203	51	412.67	149.87	89.82	63.75
20	220	80	202	53	412.00	138.21	91.82	66.25
50	228	80	213	56	403.65	139.64	93.42	70
100	268	80	241	51	410.32	140.64	89.93	63.75
200	139	52	129	38	413.98	144.66	92.81	73.08
500	160	78	149	59	416.87	140.95	93.13	75.641
500_old	165	80	149	63	416.65	137.22	90.30	78.75
1000	83	80	79	58	416.19	155.68	95.18	72.5
2000	200	80	189	58	416.98	142.67	94.50	72.5
Average time taken for an image & Accuracy					**2.2721**	**1.8711**	**92.19**	**70.90**
Overall Accuracy = 85.8632%					**Average Time = 2.1528 Seconds**			

Table 6.5 The *Te₹₹ency* Performance for training iterations=2000

The total number of correctly classified images and its accuracy are shown in the following figure 6.18 and figure 6.19 respectively.

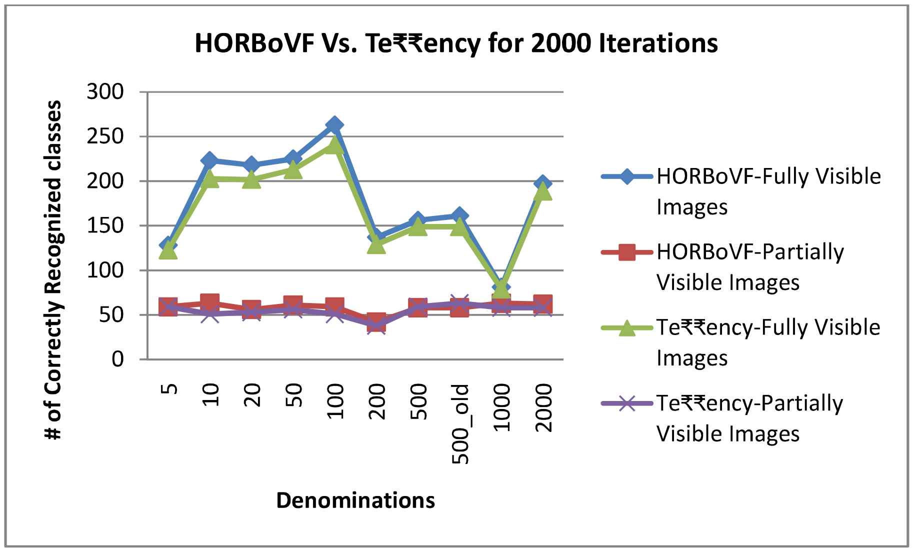


Figure 6.18. Number of correctly identified images using the *HORBoVF* and the *Te₹₹ency*

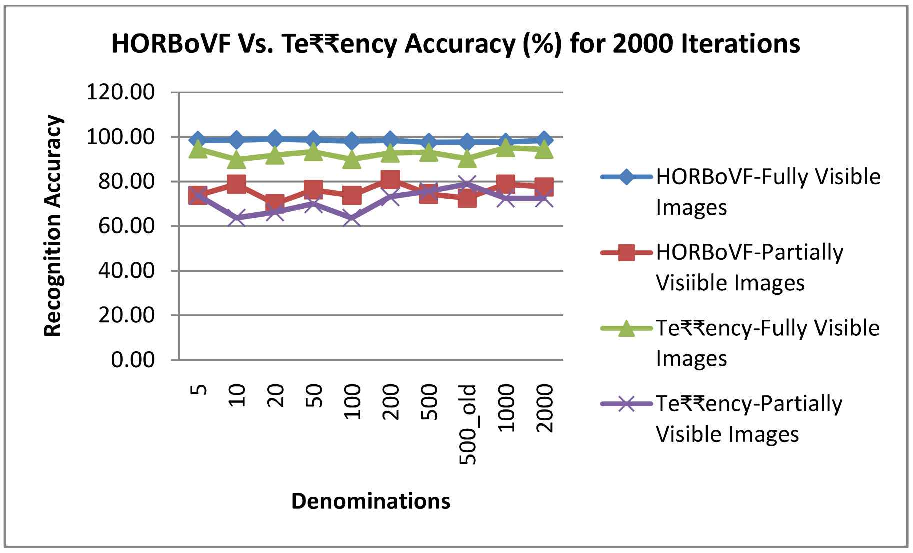


Figure 6.19. Recognition accuracy of the *HORBoVF* and the *Te₹₹ency*

For 2000 iterations, the algorithm takes 2.272 seconds and 1.871 seconds to label the image of the type fully and partially visible images, as visible in figure 6.20, which the *HORBoVF* takes 0.117 seconds for both cases. It can be observed that the labeling time remains almost unchanged for 2000 iterations too.

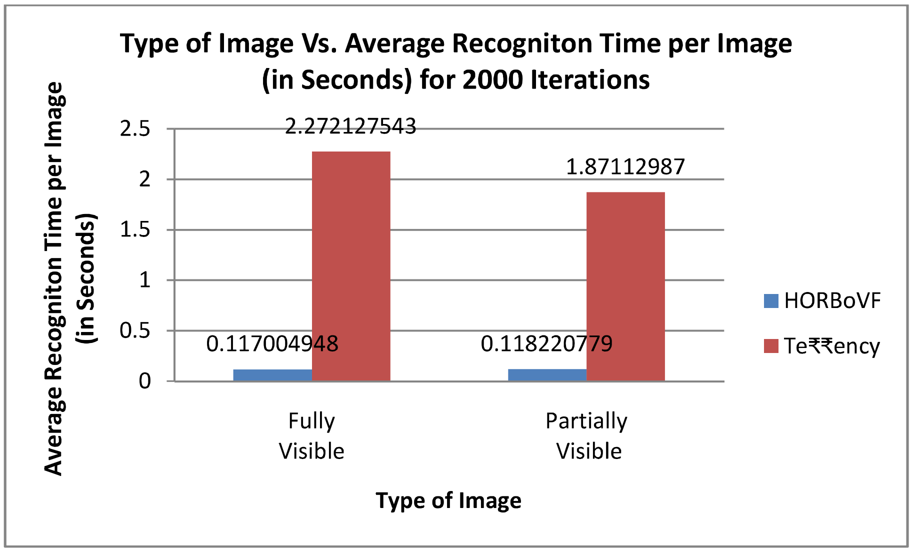


Figure 6.20. Average time taken per image by the *HORBoVF* and the *Te₹₹ency*

The overall performance is shown in figure 6.21.

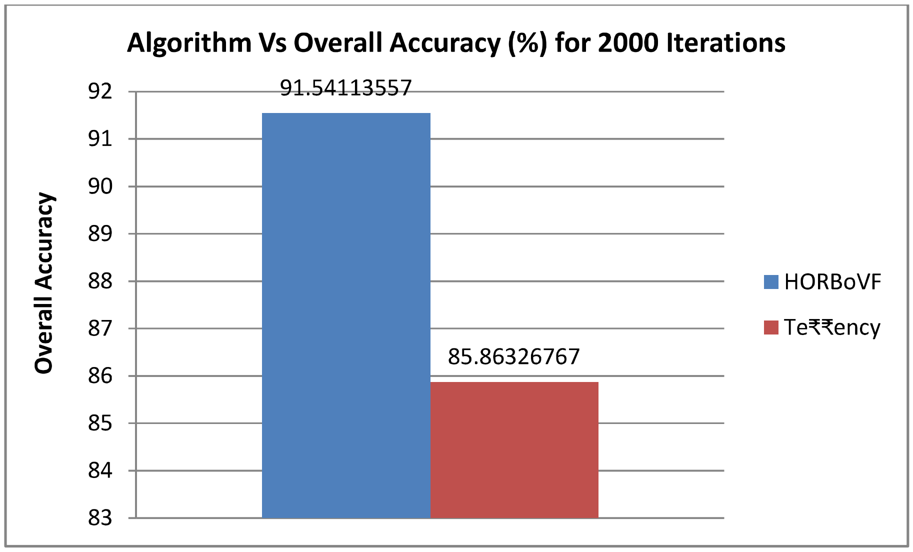


Figure 6.21. The overall performance of the *HORBoVF* and the *Te₹₹ency*

6.3.5 For Iterations=4000

The following table shows the performance of the *Te₹₹ency* for 4000 iterations.

Denomination	# of Test Images		# of Correct Classes		Execution Time (Seconds)		Classification Accuracy (%)	
	F	P	F	P	F	P	F	P
5	130	80	124	59	412.69	140.53	95.38	73.75
10	226	80	204	52	415.62	152.36	90.27	65
20	220	80	206	54	409.87	148.27	93.64	67.5
50	228	80	213	57	406.94	144.63	93.42	71.25
100	268	80	249	51	412.06	141.09	92.91	63.75
200	139	52	131	39	405.11	140.35	94.24	75.00
500	160	78	153	59	419.79	139.98	95.63	75.641
500_old	165	80	156	64	416.32	141.67	94.55	80
1000	83	80	79	59	419.68	152.46	95.18	73.75
2000	200	80	191	58	411.43	149.98	95.50	72.5
Average time taken for an image & Accuracy					**2.2702**	**1.8848**	**93.78**	**71.68**
Overall Accuracy = 87.2151%					**Average Time = 2.1555 Seconds**			

Table 6.6 The *Te₹₹ency* Performance for training iterations=4000

For 4000 iterations, in comparison to 2370 out of 2589 images for the *HORBoVF*, the *Te₹₹ency* identifies 2258 images correctly giving an overall accuracy of 87.215% as compared to the *HORBoVF*'s 91.541%. In this, for the fully visible images, the accuracy is 93.787%, and the same for the partially visible images is 71.688% as compared to 98.351% and 75.454% of the *HORBoVF* respectively. The total number of correctly classified images and its accuracy are shown in the following figure 6.22 and figure 6.23 respectively.

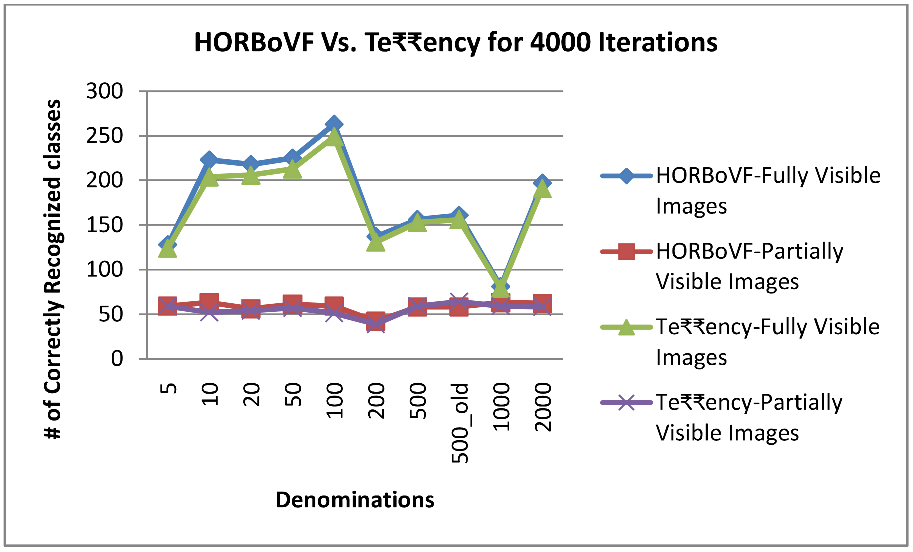


Figure 6.22. Number of correctly identified images using the *HORBoVF* and the *Te₹₹ency*

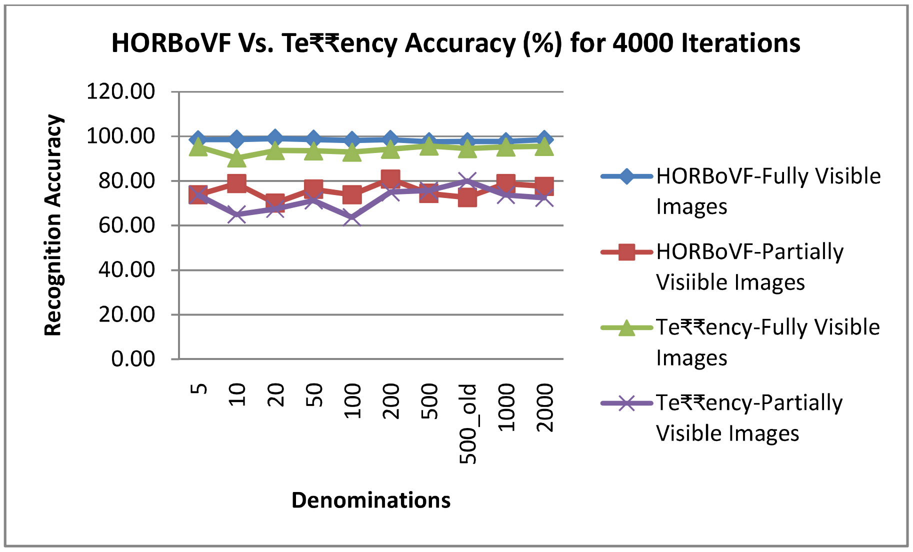


Figure 6.23. Recognition accuracy of the *HORBoVF* and the *Te₹₹ency*

For 4000 iterations, the algorithm takes 2.27 seconds and 1.884 seconds to label the image of the type fully and partially visible images, as visible in figure 6.24, which the *HORBoVF* takes 0.117 seconds for both cases. For 4000 iterations too, the labeling time remains the same.

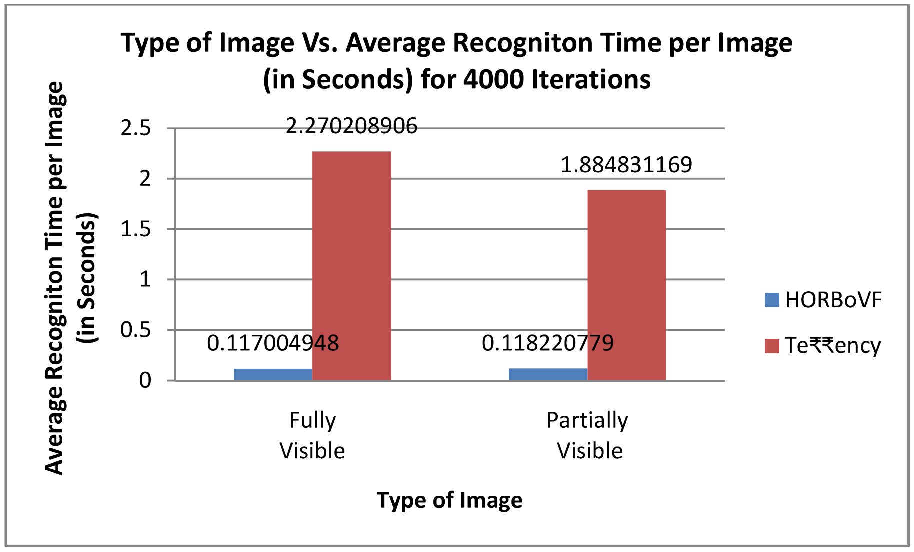


Figure 6.24. Average time taken per image by the *HORBoVF* and the *Te₹₹ency*

The overall performance is shown in figure 6.25.

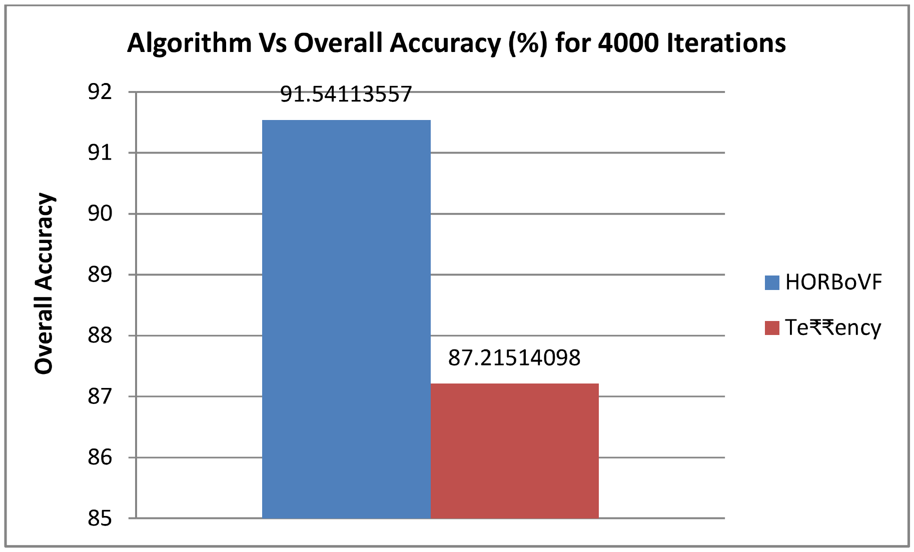


Figure 6.25. The overall performance of the *HORBoVF* and the *Te₹₹ency*

The summarized performance of all the iterations in terms of time and accuracy is given below.

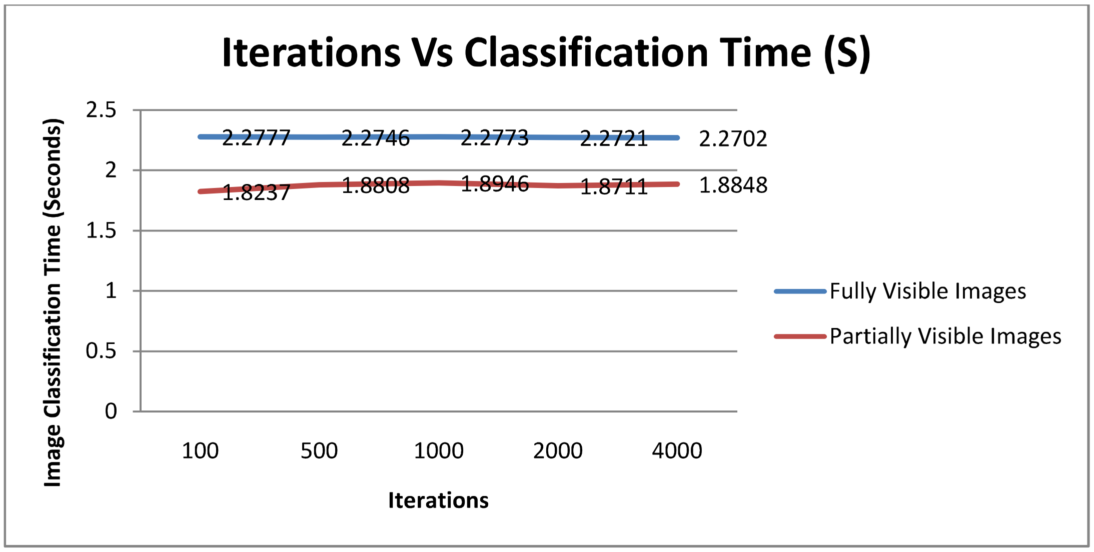


Figure 6.26. Average time consumption of the *Te₹₹ency* for different iterations

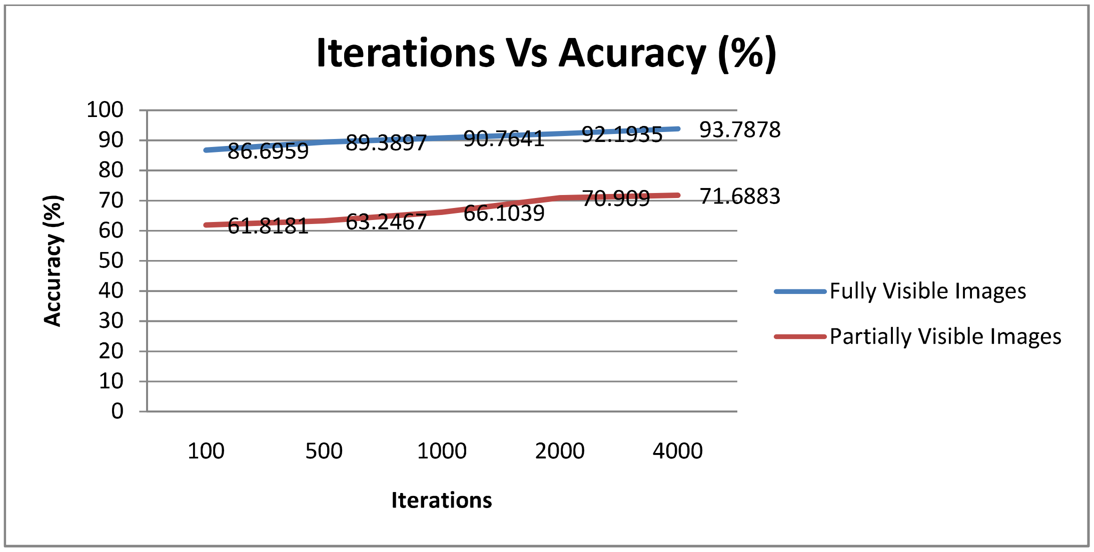


Figure 6.27. The overall accuracy of the *Te₹₹ency* for different iterations

Throughout the results, it can be observed that the *HORBoVF/ACORBoVF* consistently outperforms *Te₹₹ency* for both, the fully and partially visible, kinds of images. This proves that the bag of visual features works better than the tensor model. The following figures show the performance comparison of the *HORBoVF*, the *ACORBoVF,* and the *Te₹₹ency* in terms of accuracy and time taken for recognition, both.

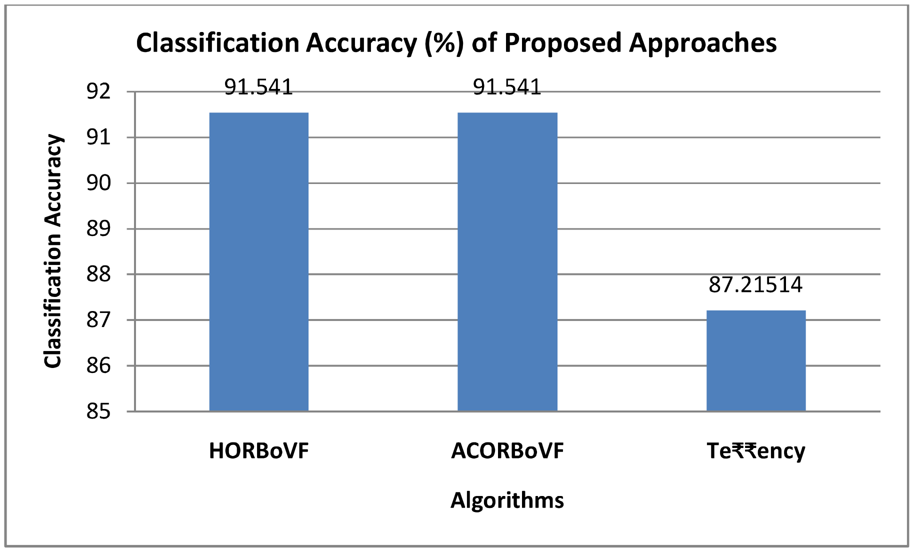


Figure 6.28. The Accuracy performance of the *HORBoVF*, the *ACORBoVF* and the *Te₹₹ency*

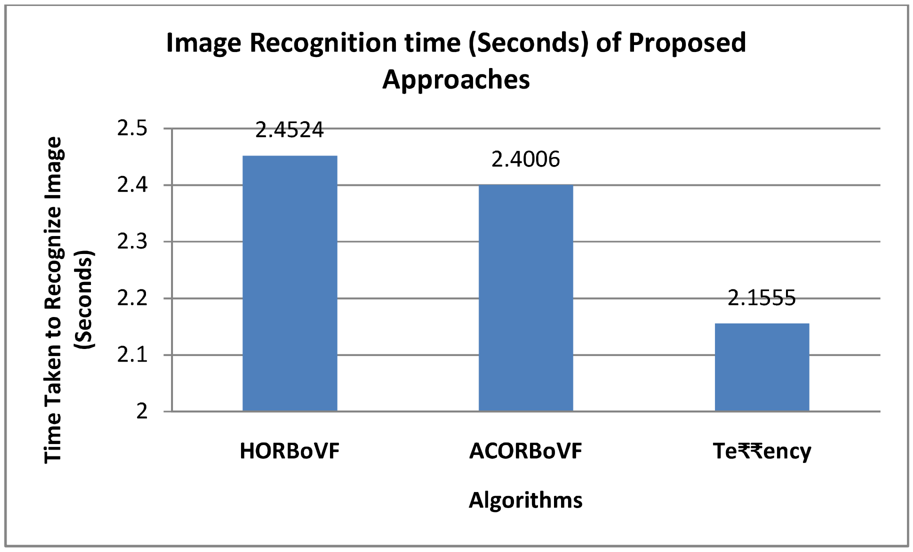


Figure 6.29. The time performance of the *HORBoVF*, the *ACORBoVF* and the *Te₹₹ency*

SUMMARY

This chapter briefly introduced the CNNs and TensorFlow technology in the beginning. Next, it gave a detailed description of a retrained TensorFlow model, *Te₹₹ency*, and its performance analysis with reference to the *HORBoVF*. Finally, it summarizes the performance of all three classifiers. The next chapter concludes the work and shows some directions for the future work.