**Weather classification using single image by fusion of features**

**Name,email,college**

**Abstract:**

*Vision based driver assistance systems are required to perform well even under adverse weather conditions. This in turn requires an automated accurate multi-class weather classification system using single images. The weather classification system sense the environment conditions, which will be used by the autonomous ground platforms to select the right set of sensor for environment perception. Self driving cars are equipped with multi model sensors for environment perception which include camera, Li-DAR and RADAR for handling adverse weather conditions. Most of the papers takes the streams of camera, Li-DAR and RADAR, detect obstacle and carry out late fusion them to detect obstacles, without sensing the environment. In order to improve the detection of obstacle in adverse weather conditions, a reliable weather detection/classification system is essential. This paper presents a method which uses single image to detect the weather, which will be used in conjunction with environment perception module using multi-model sensors. Using weather classification module proper weightages can be assigned to sensor, which will improved the obstacle detection system accuracy. The proposed method uses the combination of Histogram of Gradient (HOG) \& deep features, feature reduction and classification. High dime nsional feature vector were subjected to dimensionality reduction before feeding to classifiers. Extensive experiments on the bench-mark dataset using various features extraction and selection methods were carried out in conjunction with various classifiers. The proposed method uses fusion of Histogram of Gradient (HOG) & DenseNet-161 features and SVM classifier to achieve the classification accuracy of 99.65% which outperforms other methods trained on the same dataset.*

**Keywords:**

HOG, DenseNet, ResNet, Support Vector Machine, PCA, Mutual Information

**1) Introduction:**

Self-driving cars or autonomous ground platform perceive their surrounding by combining information from a variety of sensors. These sensors work well under good weather conditions and fails to perform under adverse weather conditions. The weather condition plays critical role in functioning of self driving car, a robust weather classification system is critical, which provide valuable information. The output of weather classification can be utilized to assign proper weightages to sensors to improve cognition.

2) Related Work:

3) Our Approach:

In the task of image classification, feature extraction plays a very important role. For weather images we need to have good features in order for our model to classify them with high accuracy. Starting from low-level feature extraction methods like SURF (Speeded Up Robust Features) and ORB (Oriented FAST and Rotated BRIEF) would be a good start as they are relatively fast methods to compute but have shown poor accuracies for the classification task at hand. An improvement would be using a feature extraction method called HOG (Histogram of oriented gradients) which computes the gradient of each pixel of the image after reshaping. These gradients are used to obtain a histogram of oriented gradients which after flattening give a feature vector. A feature extraction method called Scale invariant feature transform (SIFT) which works on similar principle is also considered for our task. However, extraction methods like SURF/SIFT/ORB are used mainly in image search, object mapping tasks because they are used to describe specific points in an image whereas HOG is used to describe the image as a whole which is very useful in image recognition/classification tasks such as our own.

The features obtained through HOG were put into the classification pipeline consisting of data pre-processing, dimensionality reduction and finally into the classification algorithms. For our experiments, we used classifiers like Support Vector Machine (SVM), Decision Trees, K-Nearest Neighbors and Naïve Bayes Classifier. In addition, ensemble learning methods like Bagging, Random Forests, AdaBoost and Voting Classifier were also applied. The accuracies obtained from the classifiers were not satisfactory enough for the purpose of our experiment. And we sought for a feature extraction method which would improve upon HOG.

We ventured into Deep Learning feature extraction methods to improve our accuracy. For this we used the ResNet-18 Convolution Neural Network to extract our features. Resnet-18 is a CNN that is 18 layers deep and pretrained on the ImageNet dataset. The input image shape for ResNet-18 is 224\*224 and we use average pooling which averages the feature values obtained in the feature map. The average pool layer is extracted as our feature vector for a particular image. All such feature vectors are combined to obtain our features. Following the similar process as HOG, we created a classification pipeline and fit our model to different classifiers. There was definitely an improvement over our previous methods but we strived to improve it further.

Deeper CNNs could have been used to obtain the features but we were performing these experiments on the basis of constraints in computation ability that would be allotted to our model. Hence, we need something that would be relatively faster to compute and also improve upon our previously obtained our results. For this our approach was to use the features obtained using both HOG and ResNet-18 to train our model. This would give us richer features that would help our model to fit the data well. But there was a major problem to this approach which incidentally is the problem for so many of Machine Learning models dealing with lots of features and that is the curse of dimensionality.

The features obtained using HOG for a single image were 3780 and features obtained using ResNet-18 were 512. Using them both would give our feature count over 4000 and is not desirable. Our solution to that was to apply dimensionality reduction (Principal Component Analysis) to each of these features separately and then concatenate them together. This gave us 750 odd features which had the advantages of both HOG and ResNet features and also were in reasonable amount which ensured our classification models would not be underfit. This worked extremely well compared to previously taking each of those features individually to fit our classifiers. All the results were up to the highest of standards compared to other models on the same dataset.

3.1)Dataset used:

For the purpose of our experiment, we used the Multi Class Weather Recognition Dataset (MCWRD) which consists of 1125 images of classes Cloudy, Shine, Sunrise and Rain. We also tested our model on the Multi Weather Image (MWI) dataset to compare with other prominent classification models in the field of weather image classification. For the sake of simplicity, we considered a total images of 2000 from the MWI dataset. The dataset consists of 500 images each of classes Haze, Rainy, Snowy , Sunny.

**3.2) Image Preprocessing Module:**

**3.2.1) Loading The Images:** We loaded the images using Python Image Library (PIL). Converted all the images to RGB for uniformity.

**3.2.2) Resize:** This is required to convert all the images into the same dimension. In order to do this we used the inbuilt function, Resize in Torchvision module. This converts all the RGB images to a size of 224\*224\*3.

**3.2.3) Label encoding:** Classification deals with numerical labels. In order to change categorical labels to numerical labels, we assigned integral labels as folllows, Cloudy:0, Rain:1, Shine:2, Sunrise:3

**3.2.4)** **Data Shuffling:** This is done to reduce model biasing to a particular class. For the \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, data is shuffled randomly. For the fused features, we shuffled the data after concatenating the features.

* **Data**  **Splitting:** We split the data into training and testing set in a 75-25% train-to-test fashion. We do this so that our model can be tested first on *unseen* data before being deployed into real-world scenarios.

3.3) **Feature Extraction:**

3.3.1) **Feature descriptor:** It is a simplified representation of the image the contains only the most important information about the image.

3.3.2) **Histogram of Oriented Gradients (HOG):** The HOG descriptor focuses on the structure or the shape of an object. In case of edge features, we only identify if a pixel is an edge or not. HOG is able to provide the edge direction as well. This is done by extracting gradients and orientation of the edges.[1]

To obtain features using HOG, the images are first resized into the shape of where the ratio of height is to width is 1:2. (Preferably 64\*128)

Our Work: Before extracting features, all the images undergo preprocessing process. In order to obtain HOG features of an image, we use ***hog*** function in ***skimage.features***. When passed along with the parameters, it returns a feature vector of size 3760.

Drawbacks with HOG features: It is very senstitve to image rotation.

To improve the results, deep learning methods were used to extract features.

The introductiion of very Deep Convolutional Neural Networks helped the models to archieve state-of-art results on the tasks like image recognition and image classification.

Over the years, the deep neural netwoks got deeper and deeper to solve more complex tasks.

Deep neural networks are hard to train because of the vanishing gradient problem. Models like ResNet, DenseNet were introduced to reduce this issue. In order to solve the problem of vanishing gradient, this architecture introduced the conecpt called residual blocks. In this network, we use a technique called skip connections. The skip connection connects activations of a layer to further layers by skipping some layers in between. This forms a residual block. Resnets are made by stacking these residual blocks together.

The approach behind this network is instead of layers learning the underlying mapping, we allow the network to fit the residual mapping.

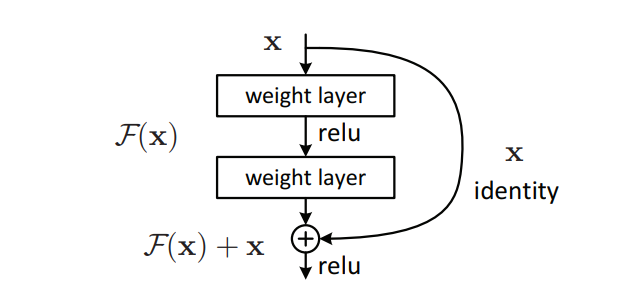
So, instead of say H(x), initail mapping, let the network fit,

3.3.3) **Residual Networks(ResNets):** In order to solve the problem of vanishing gradient, this architecture introduced the conecpt called residual blocks. In this network, we use a technique called skip connections. The skip connection connects activations of a layer to further layers by skipping some layers in between. This forms a residual block. Resnets are made by stacking these residual blocks together.

The approach behind this network is instead of layers learning the underlying mapping, we allow the network to fit the residual mapping.

So, instead of say H(x), initail mapping, let the network fit,

***F(x) := H(x) - x* which gives *H(x) := F(x) + x*.**



To extract features from ResNet models, we imported the five pretrained ResNet models from Pytorch. After importing the models, we replaced model.fc with nn.Identity. Identity() will just retrun the input without any clone usage or manipulation of the input and since the features are its input, the ouput of the entire model will be the features. We gave the dataset as the input to our ResNet models and obtained the corresponding feature vectors.

The pretrained ResNet models are:

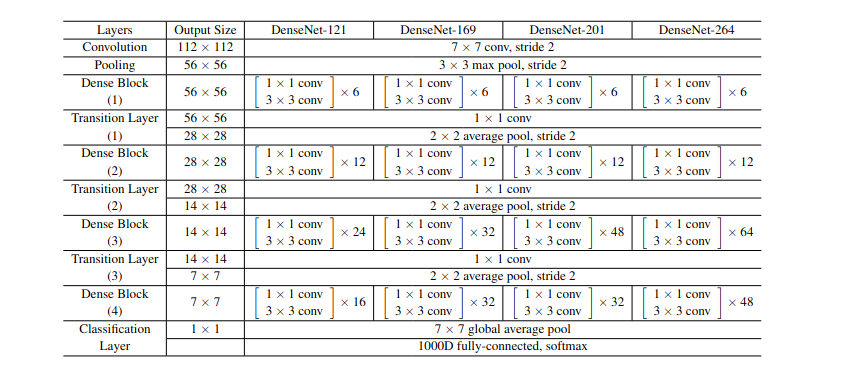
* + ResNet-18
  + ResNet-34
  + ResNet-50
  + ResNet-101
  + ResNet-152

3.3.4) **Densely Connected Networks (DenseNets):** A **DenseNet** is a type of convolutional neural network that utilizes dense connections between layers, through Dense Blocks, where we connect *all layers* (with matching feature-map sizes) directly with each other. To preserve the feed-forward nature, each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers.

To extract features from DenseNet models, we imported the four pretrained DenseNet models from Pytorch. After importing the models, we replaced model.classifier with nn.Identity. Using a similar approach as used in extracting features from ResNet models, we extracted the corresponding feature vectors from DenseNet models.

The pretrained DenseNet models are:

* + DenseNet-121
  + DenseNet-161
  + DenseNet-169
  + DenseNet-201

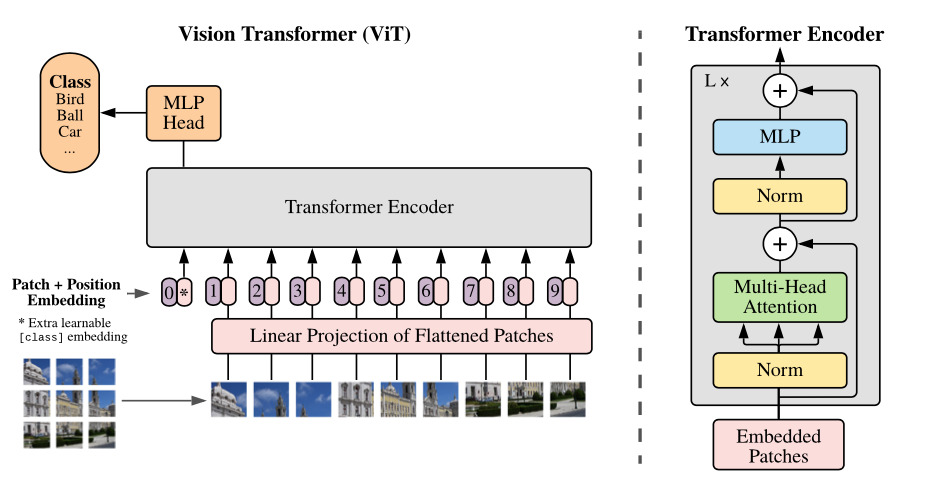


3.3.5) **Vision Image Transformers (ViTs):** The **Vision Transformer**, or **ViT**, is a model for image classification that employs a Transformer-like architecture over patches of the image. An image is split into fixed-size patches, each of them are then linearly embedded, position embeddings are added, and the resulting sequence of vectors is fed to a standard Transformer encoder. In order to perform classification, the standard approach of adding an extra learnable “classification token” to the sequence is used.

To extract features from ViT models, we imported the four pretrained ViT models from Pytorch. After importing the models, we replaced model.head with nn.Identity. Again using the similar approach, we extracted the corresponding feature vectors from ViT models.

The pretrained ViT models are:

* + Vit\_b\_16
  + Vit\_b\_32
  + Vit\_l\_16
  + Vit\_l\_32



3.4) **Feature Selection and Dimensionality Reduction:**

3.4.1) **Mutual Information based feature selection:** Feature selection is used to choose a subset of relevant features for effective classification of data. In high dimensional data classification, the performance of a classifier often depends on the feature subset used for classification. This method utilizes feature–feature mutual information to find an optimal subset of features to minimize redundancy and to maximize relevance among features. The effectiveness of the selected feature subset is evaluated using multiple classifiers on the datasets used in this paper.

The mutual information between two random variables *X* and *Y* can be stated formally as follows:

**I (****X : Y) = H(X) – H (X | Y)**

Where I (X: Y) is the mutual information for X and Y, H(X) is the entropy of X and H(X|Y) is the conditional entropy for X given Y. Mutual information is a measure of dependence between two random variables.

3.4.2) **Dimensionality Reduction using PCA:** *The curse of dimensionality* basically means that the error increases with the increase in the number of features. It refers to the fact that algorithms are harder to design in high dimensions and often have a running time exponential in the dimensions.

Principal Component Analysis (PCA) helps us to identify patterns in data based on the correlation between features. In a nutshell, PCA aims to find the directions of maximum variance in high-dimensional data and projects it onto a new subspace with equal or fewer dimensions than the original one.

In our extensive experiments, we found that the dimensions of the raw features were quite high

owing to the fact that the features obtained through HOG were of the dimension 3780. This contributed to a high dimension feature space which slows the execution of the classification model.

Over the course of our research, we found out that the best results were obtained when we fused the features obtained through a Deep learning method and a non-DL method. The non-DL features obtained through HOG were able to enrich the information of the feature space which previously contained the feature vectors from the DL models.

We used different approaches to reduce dimensionality and apply feature selection. First was to apply PCA to the individual feature vectors from DL models and HOG and thereby concatenating them to obtain the final features.

The second approach was to apply feature selection to the individual features after applying PCA and then concatenating them.

A different approach was to first apply feature selection and concatenating them to obtain the final features. Also, a variation of this would be to apply PCA after feature selection and finally concatenating them to obtain the fourth feature space.

3.5) **Classification Model:**

Combining the features of the base DL models, base DL models and HOG along with the four aforementioned feature spaces, we trained our classifier models on these six feature spaces.

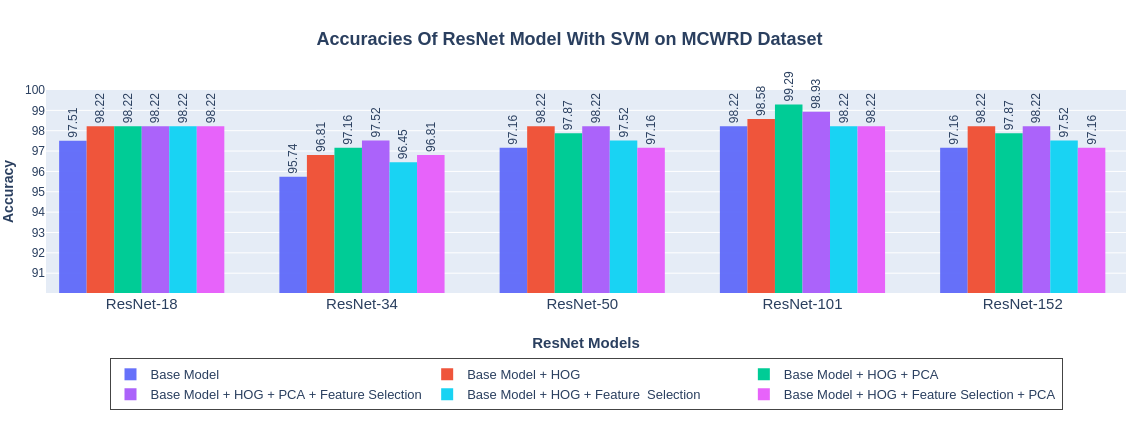
We trained classical Machine learning models and obtained the results. In our experiments, we found out that the classifier with the highest accuracy was a Support Vector (Machines) classifier which outperformed Random Forest, K-Nearest Neighbors, Decision Tree and Naive Bayes Classifiers.

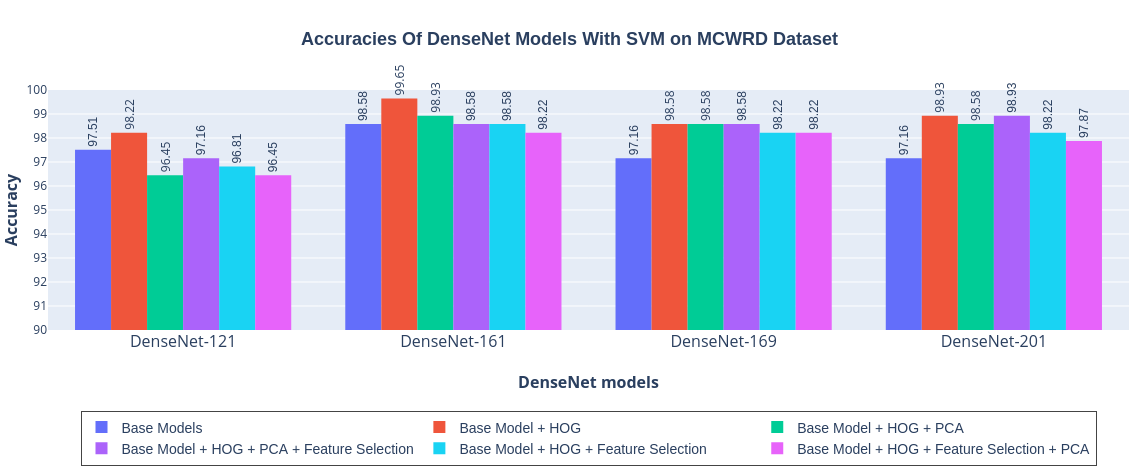
**4) Experiment Results:**

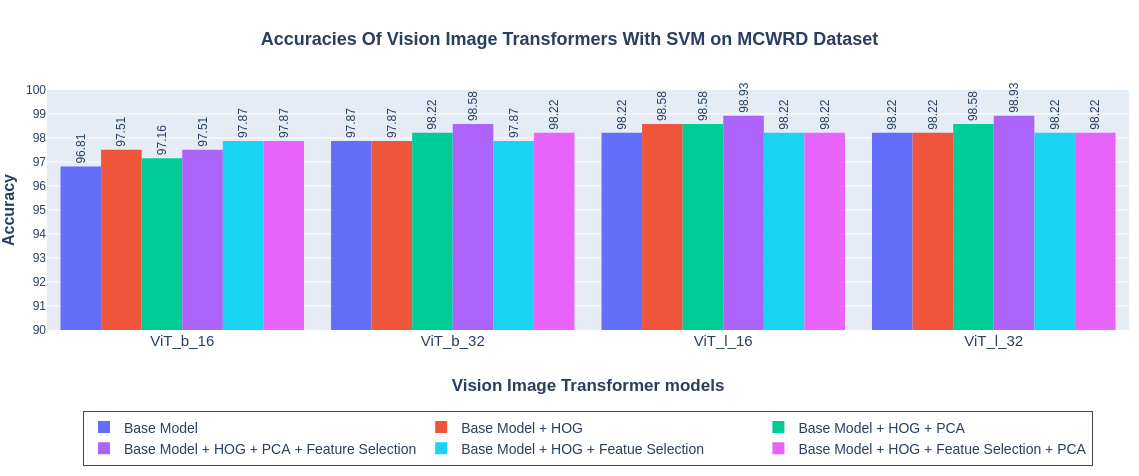
4.1) Why SVM?

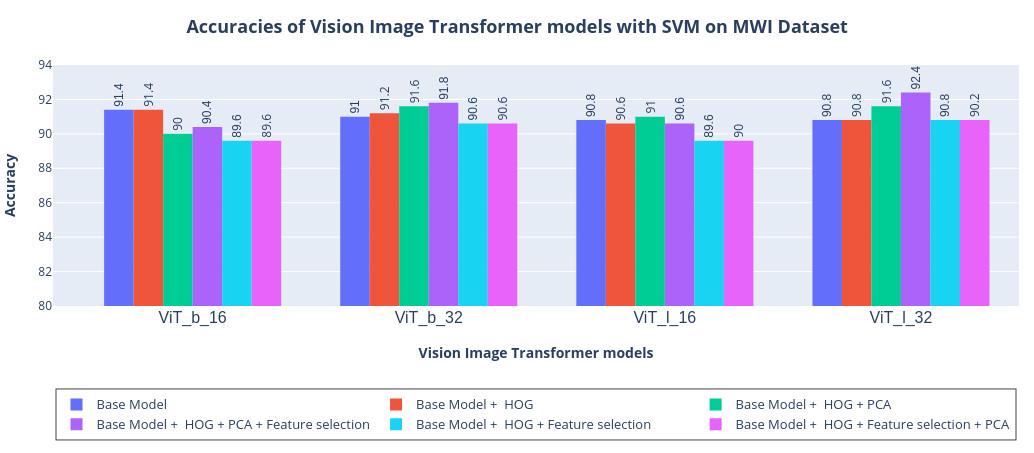
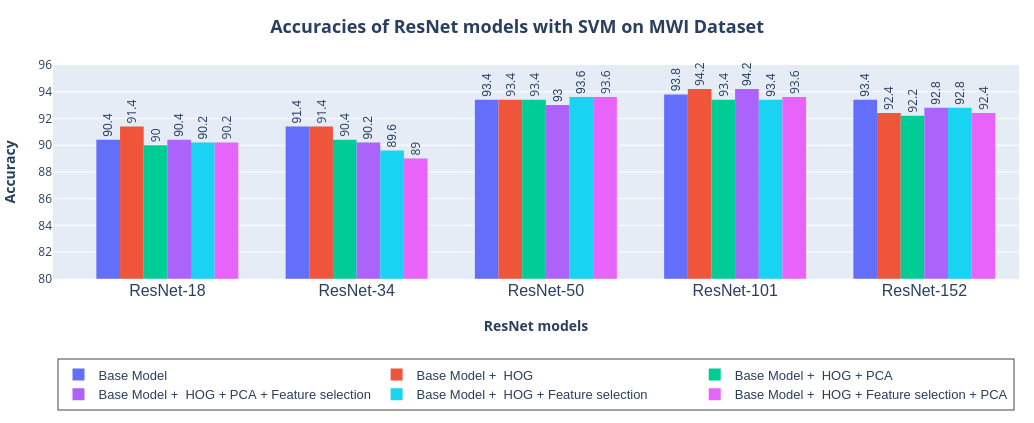
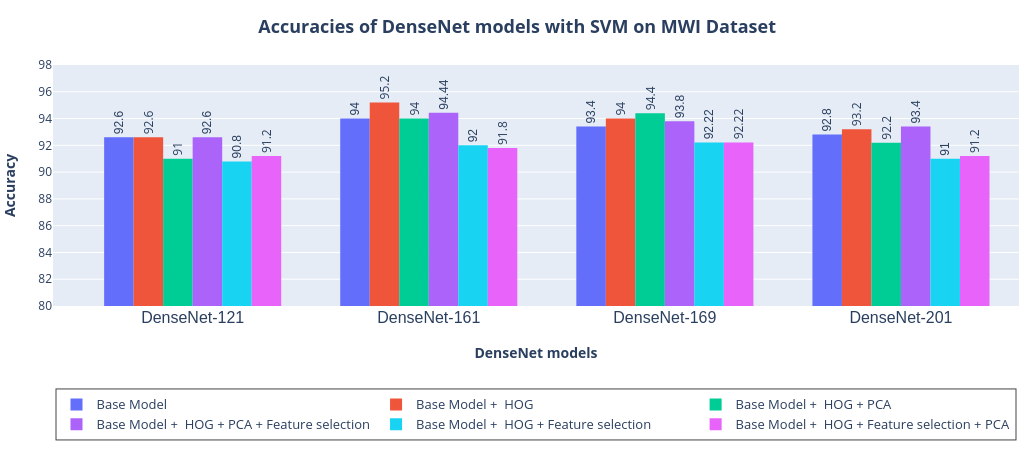
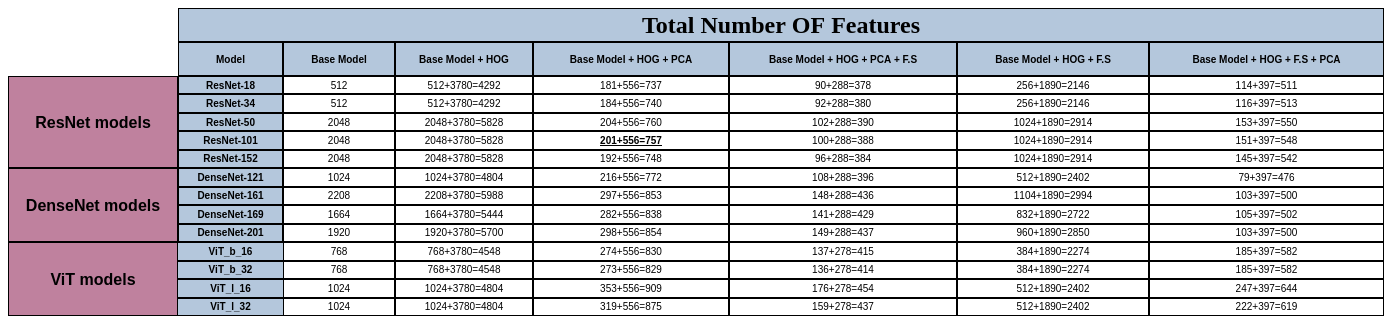
4.2) Table of Accuracies(Test) with SVM for various models

4.3) Bar plots of the same



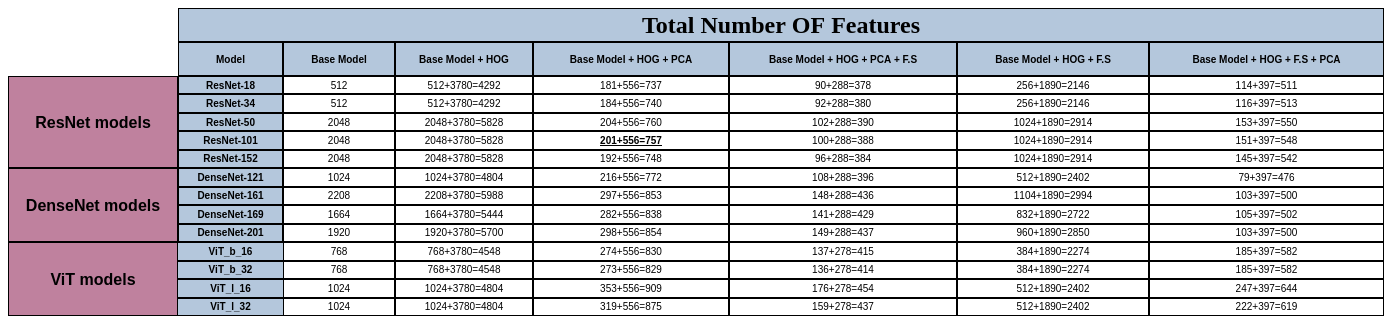


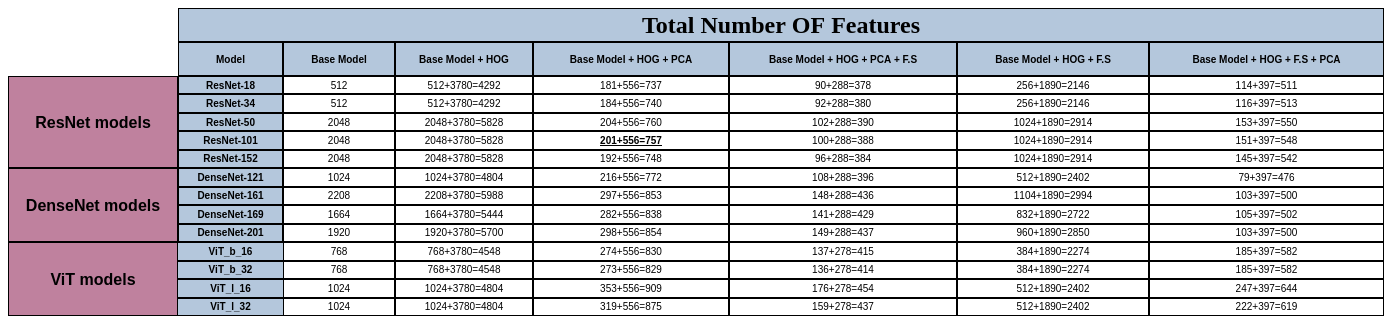
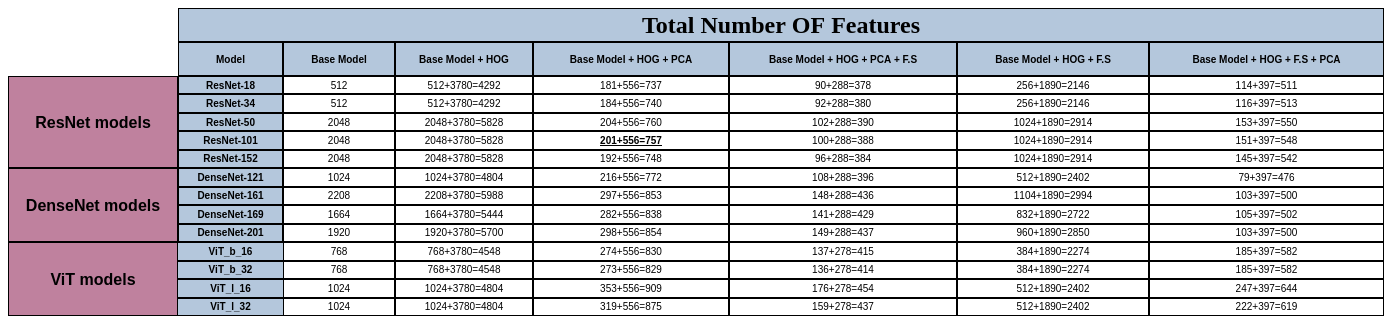


Total Number of Features: -

The table below shows the number of features the classifier model is trained with in different approaches as discussed in the previous sections.





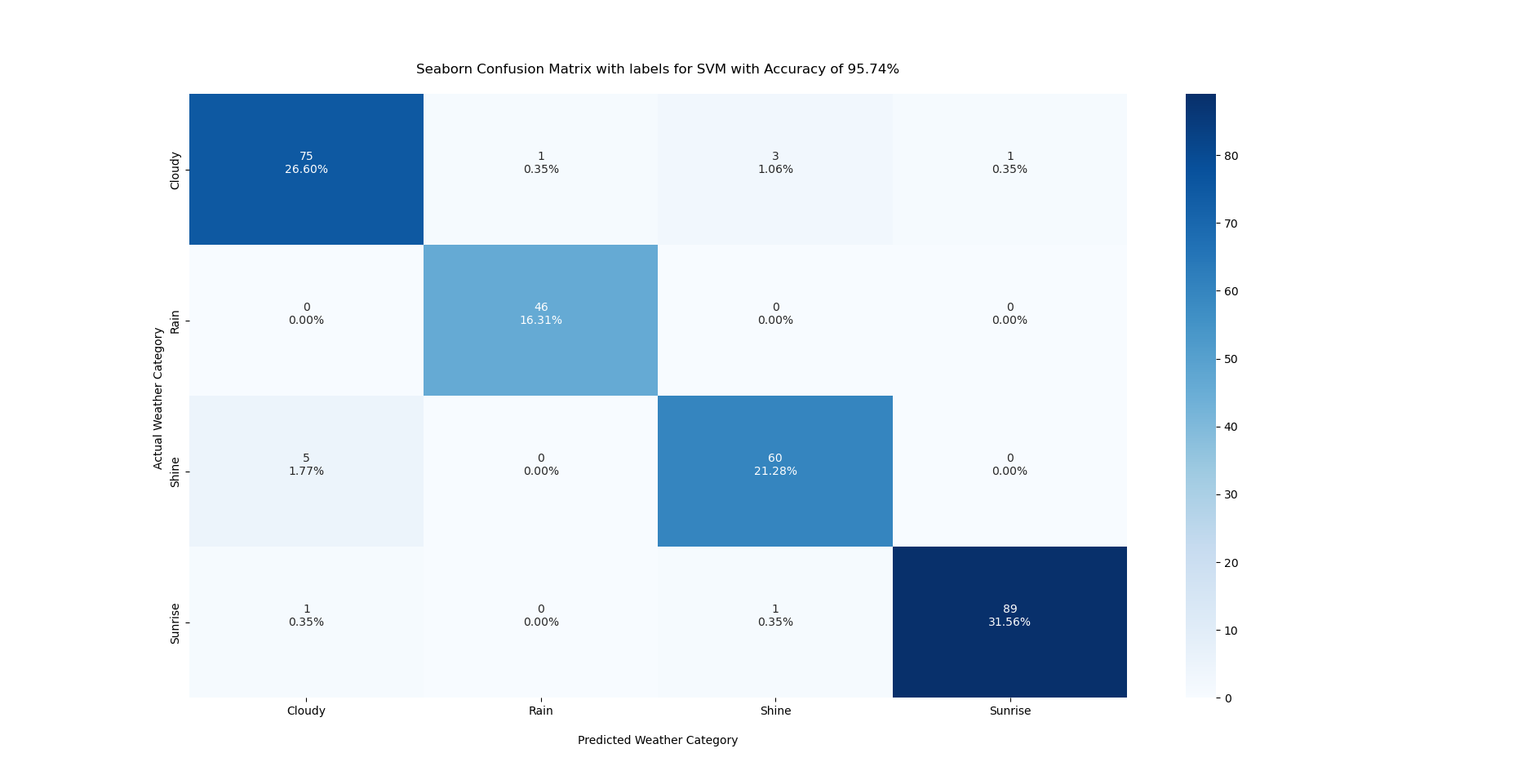
4.4) Classification report and confusion matrices of the best 2 features extraction methods.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification Report for DenseNet161+HOG with SVM on MCWRD Dataset | | | | |
| Classes | Precision | Recall | F1-score | Support |
| Cloudy | 0.97 | 0.94 | 0.96 | 80 |
| Rain | 1.00 | 1.00 | 1.00 | 46 |
| Shine | 0.93 | 0.97 | 0.95 | 65 |
| Sunrise | 0.99 | 0.99 | 0.99 | 91 |
| Accuracy |  |  | 0.97 | 282 |
| Macro avg | 0.97 | 0.97 | 0.97 | 282 |
| Weighted avg | 0.97 | 0.97 | 0.97 | 282 |

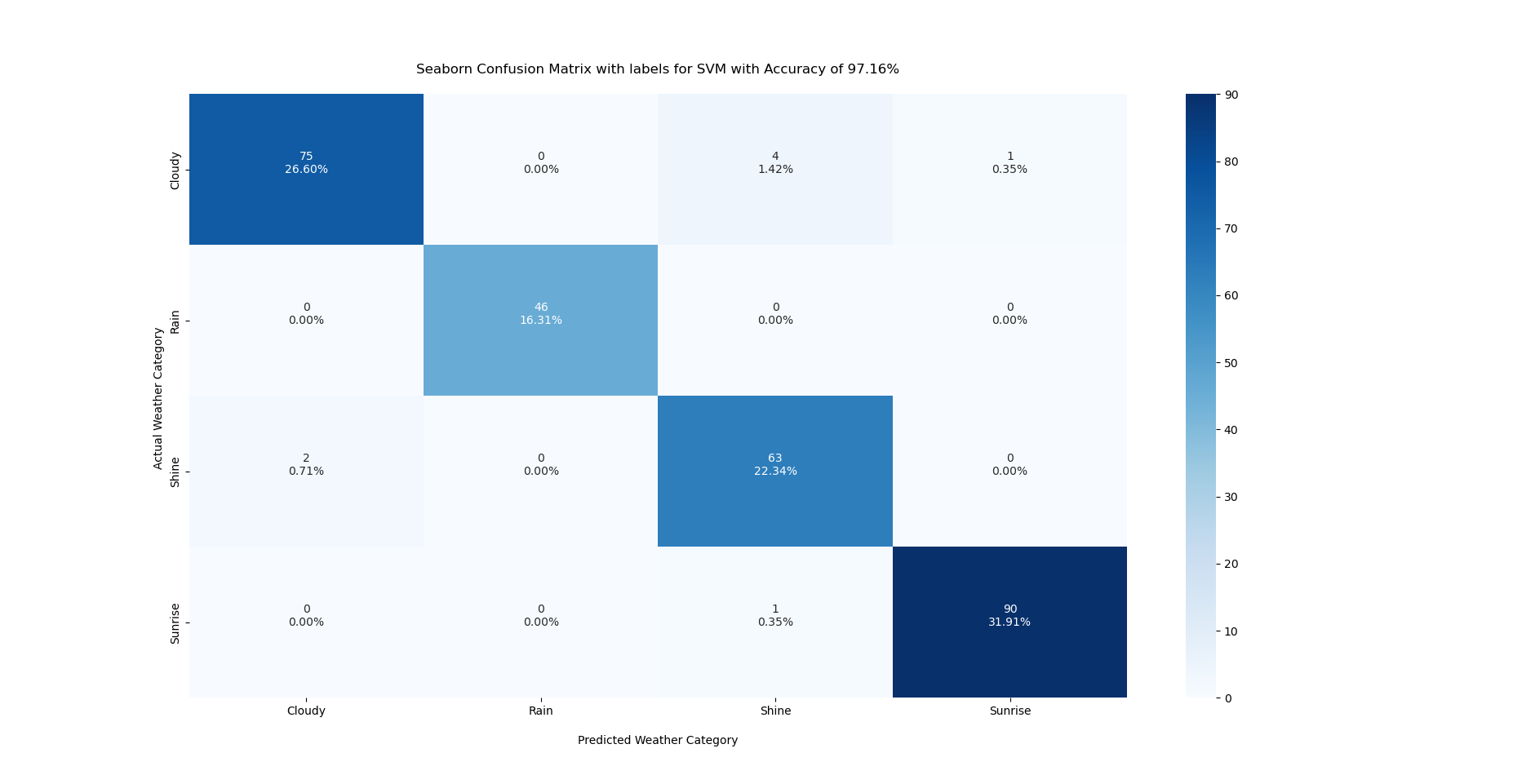
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Classification Report for ResNet101+HOG+PCA with SVM on MCWRD Dataset | | | | |
| Classes | Precision | Recall | F1-score | Support |
| Cloudy | 0.93 | 0.94 | 0.93 | 80 |
| Rain | 0.98 | 1.00 | 0.99 | 46 |
| Shine | 0.94 | 0.92 | 0.93 | 65 |
| Sunrise | 0.99 | 0.98 | 0.98 | 91 |
| Accuracy |  |  | 0.96 | 282 |
| Macro avg | 0.96 | 0.96 | 0.96 | 282 |
| Weighted avg | 0.96 | 0.96 | 0.96 | 282 |

Confusion Matrix for 2 Best Models:

ResNet101+HOG+PCA on MCWRD dataset:



DenseNet161+HOG on MCWRD dataset:



4.5) Table for comparison of various proposed models

5) Conclusion:

6) References:

* 1. <https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/>
  2. [**https://www.analyticsvidhya.com/blog/2021/06/build-resnet-from-scratch-with-python/**](https://www.analyticsvidhya.com/blog/2021/06/build-resnet-from-scratch-with-python/)
  3. [**https://www.analyticsvidhya.com/blog/2022/03/introduction-to-densenets-dense-cnn/**](https://www.analyticsvidhya.com/blog/2022/03/introduction-to-densenets-dense-cnn/)
  4. <https://www.analyticsvidhya.com/blog/2021/03/an-image-is-worth-16x16-words-transformers-for-image-recognition-at-scale-vision-transformers/>
  5. <https://en.wikipedia.org/wiki/Principal_component_analysis>
  6. <https://huggingface.co/blog/fine-tune-vit>
  7. <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiTorKvwNf4AhVCxDgGHWQjA_UQFnoECAoQAw&url=https%3A%2F%2Fanalyticsindiamag.com%2Fcurse-of-dimensionality-and-what-beginners-should-do-to-overcome-it%2F&usg=AOvVaw0ptC9qcpMEqOpsB4lfIqfw>
  8. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rja&uact=8&ved=2ahUKEwiBieftwdf4AhX16jgGHfDtAM0QFnoECAgQAw&url=https%3A%2F%2Ftowardsdatascience.com%2Fprincipal-component-analysis-for-dimensionality-reduction-115a3d157bad&usg=AOvVaw3k9vfO1CoP5vJCgu59NOys