Image Classification based on Marvel Dataset

s2306209 **s2449908**

Abstract

In this report we have done image classification on a "Marvel Avengers" Dataset using two models. One classical model and one Deep Learning model. We have chosen Support Vector Machine Classifier (SVC) as the classical model and DenseNet201 as the deep learning model. On training the model on the dataset given to us, we were able to achieve an accuracy of 27% in SVC and 49% on DenseNet. Further we applied 8 different perturbations on the trained model against the test data to understand the chances in accuracy and F1-Score for 10 different magnitudes of the filters. Overall, The DenseNet201 gave us better results than the classical model.

1 Introduction

Image classification is one of the most pressing problems that require novice solutions in the current world. There are many classification models that can be used to classify images but selecting the best one based on a certain level of expertise is necessary in order to obtain the desired results. In the problem statement that was provided to us, we were asked to run image classification models on a Marvel dataset. we were expected to utilise two image classification models, one using a classical model and the other using a deep learning model. We will be using Support Vector Machine with SIFT and Bag of Visual Words for our classical model and Desenet 201 for our deep learning model. Initially, for our deep learning model, we tried implementing the classical Convolution Neural Networks (CNN) after which we tried Densenet 201 which proved to be more efficient. The existing research on image classification models shows that deep learning models perform comparatively better and faster than classical machine learning models such as Support Vector Machines and K-Nearest Neighbours. This is because high dimensionality of the feature space due to which traditional models generalize poorly on image classification tasks [1]. CNN can take the images as the direct input, and robust to rotation, translation and scaling deformation of images. CNN has undergone multiple modifications and developments to increase its efficiency in classifying images. Densenet 201 is yet another CNN model that is 201 layers deep. It can classify up to a million images with a 1000 labels.

2 Analysis

We were given an "Marvel Avengers" dataset consisting of 8 labels with 2584 images in the training dataset and 451 images on the test dataset. Considering that we have to utilise one traditional (classical) method and one deep learning method, a few research papers were read to understand which would give us the more optimal solution. For the traditional method, we went ahead with Support Vector Classifier as it performs better than K Nearest neighbours for image classification [3]. SVC is used with a combination of Scale-Invariant Feature Transform (SIFT) [4]. The other option was to use it with Histogram of Gradients (HOG). But in comparison SIFT performs better than HOG for feature extraction. As the name suggests SIFT is scale and rotation invariant whereas HOG is not invariant to these factors. SIFT provides a robust mechanism for detecting distinctive invariant image features which provide robust matching between different views of an image and HOG is used to characterize local object appearance and shape by the distribution of local intensity gradients or

edge directions without the prior knowledge about edge location. Since in this problem statement we were tasked to extract features from mostly faces we went ahead with SIFT. Bag of Visual Words (BoVW) is used after the features are extracted from SIFT to create a dictionary of visual words as the problem statement demands.

3 Data Preparation

The image dataset provided contained two different folders, one for train and the other valid for testing the model which was trained using the images from the train folder which was validated against unseen data in the valid folder.

Inorder to preprocess the data we tried to remove outliers using an outlier detection algorithm after cropping out the faces from the images using openCV libraries. But on doing so almost 2000 images were detected to be outliers which left us with 500 images in our train set. Since the number of training data decreased, we did not proceed with this, as the size wasn't sufficient to train the model.

The label encoding to run the models is done as => Loki: 0, Black Widow: 1, Hulk: 2, Doctor Strange: 3, Captain America: 4, Spider Man: 5, Thanos: 6, Ironman: 7.

4 Methodology

4.1 Deep Learning Model - DenseNet201

DenseNet201 is a Convolutional Neural Network which is 201 layers deep.DenseNet works on the basis that convolutional networks can be considerably deeper, more accurate, and efficient to train if they have shorter connections between layers close to the input and those close to the output. In our algorithm, we have utilised the resized images provided to us. We take this image from the train data and resize it to 128,128 before processing it. Then we trained the data on DenseNet201 using adam optimiser and applied log loss to this. We then take the data from valid to test the trained model against some unseen data. After which we tried to add 8 perturbations which were given to us and plotted graphs against each of these against F1-score and the variation in perturbations.

4.2 Classical Model - Support Vector Machine Classifier

Support Vector Machine Classifier is a supervised learning algorithm. This is a classification algorithm which works on a basis of plotting data on a n-dimensional space, n being the number of features, with the value of each feature being the value of a particular coordinate. In our algorithm SVC is used with SIFT feature extraction to obtain the features from the dataset. Further these features are used with Bag of Visual Words (BoVW) to generate a dictionary of digital words. The images in this are not resized initially as SIFT works on the basis of obtaining features from data of any size. BoVW uses mini batch K-Means clustering algorithm to create this dictionary of visual words. In this case, as the number of iterations increases the effect of new data is reduced. Therefore, convergence can be detected when no changes in the clusters occur in several consecutive iterations. The train data is used to train this model and is validated against the valid dataset. Again after which 8 perturbations are plotted for checking the robustness. The implemented algorithm can be visualised in Figure 1

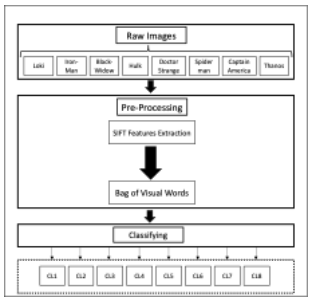

Figure 1: Classification algorithm (SVC) architecture

5 Results

5.1 Deep Learning Model - Densenet201

The figures 3 to 10 shows the perturbation results obtained. The test data was ran with 8 different types of filters, namely, Guassian noise, Guassian Blur, contrast increase, contrast decrease, brightness increase, brightness decrease, salt & pepper noise and occulsion. It was observed that the label "spider man" obtained better F! scores in comparision to the other labels. We deducted that this was becuase this label had lesser outliers and that the face was just a red mask with big white eyes which enabled the model to learn this easily.

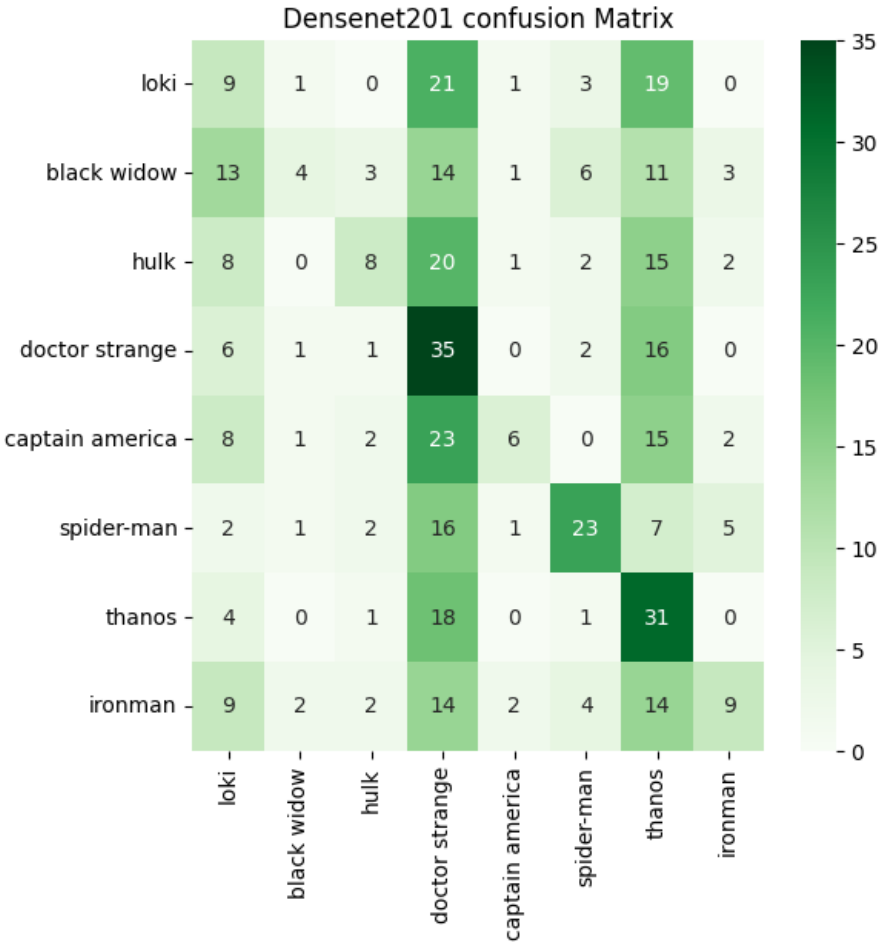

Figure 2: Confusion Matrix: DenseNet201 confusion matrix across the labels

It is observed that on using DenseNet201 for the problem statement, we were able to secure an accuracy of 49.0%. This performed considerably well in comparision to SVC. We can assume that this is mainly due to the deep layers of the convoluted network. The figure 2 shows the confusion matrix obtained after running the trained model against the test data. It can be deducted from the figure that the labels "Doctor Strange ", "Spider man" and "Thanos" obtained good true positive and true negative scores and low false positive and false negative scores.

5.1.1 Perturbation results - DenseNet201

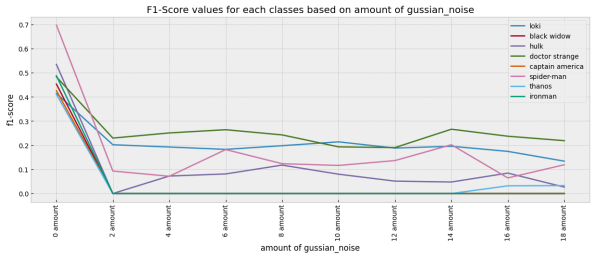

Figure 3: Perturbation - Guassian Noise DenseNet
From the graph above applying Guassian noise on the valid data set decreases the accuracy score significantly. However, Dr.Strange accuracy score became almost constant after 2nd iteration.

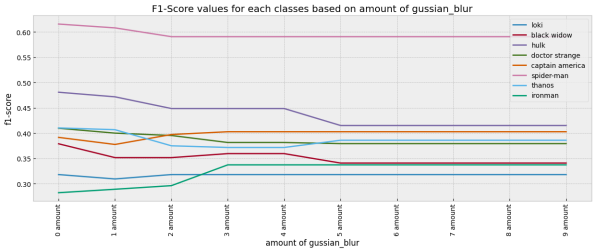

Figure 4: Perturbation - Guassian Blur DenseNet
From this it can be concluded that increasing the amount of gaussian blur has minimal effect on the f1 score.

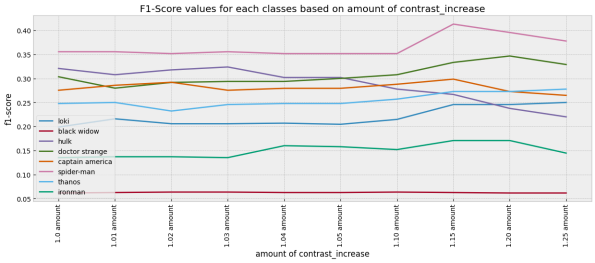

Figure 5: Perturbation - Contrast Increase DenseNet
From this it can be concluded that increasing the contrast has small change on the f1 score. However, for spiderman there was a significant increase in the f1 score for 1.15 contrast setting.

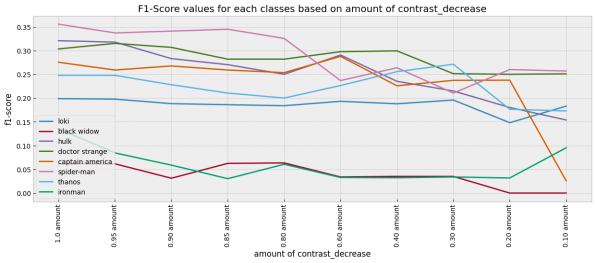

Figure 6: Perturbation - Contrast Decrease DenseNet
It can be concluded the decreasing the contrast doesn't have a significant effect on the f1 score. However, after 0.20 the f1-score falls drastically for captain America.

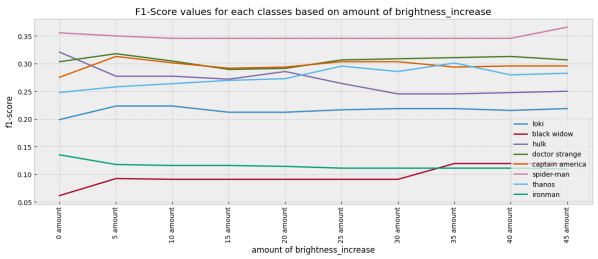

Figure 7: Perturbation - Brightness Increase DenseNet
From the graph it can be concluded that increasing the amount of brightness has minimal or almost constant effect on the f1-score.

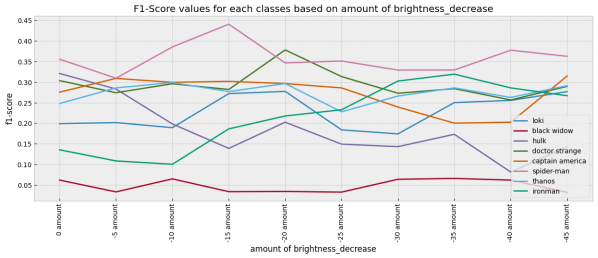

Figure 8: Perturbation - Brightness Decrease DenseNet
From the graph it can be concluded that on applying brightness decrease filter, we obtain the best result for different class at different setting of the brightness. For instance, for spiderman the best f1 score was at -15. This also suggests that using different brightness for different classes, maximum f1 score for the whole dataset can be obtained

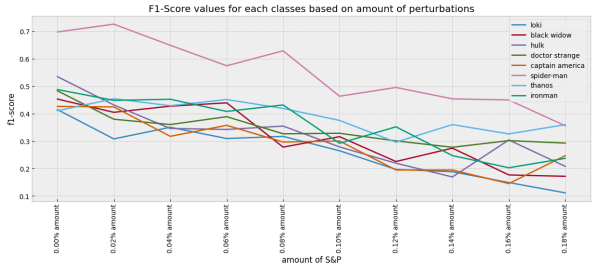

Figure 9: Perturbation - Salt and Pepper DenseNet
From this graph it can be observed that there is a decreasing tread in the f1 score as the amount of salt and pepper filter is increased.

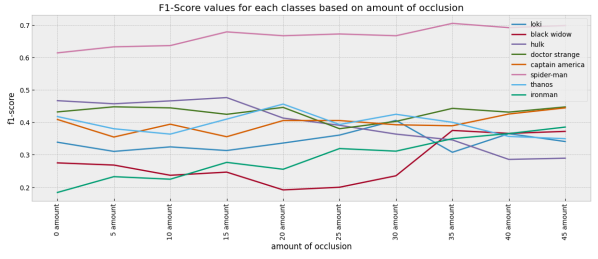

Figure 10: Perturbation - Occlusion DenseNet
From this it can be observed that, as the amount of occlusion is increased, it does not have a significant effect on other classes except black widow. However, it can also be observed that the f1 score for black widow has increased significantly here as compared to all the other filters.

5.2 Classical Model - Support Vector Machine Classifier

On using Support Vector Machine Classifier, we were able to obtain an accuracy of 27%. This model did not perform up to the mark for the given dataset. We can assume that this is due to the large number of outliers.

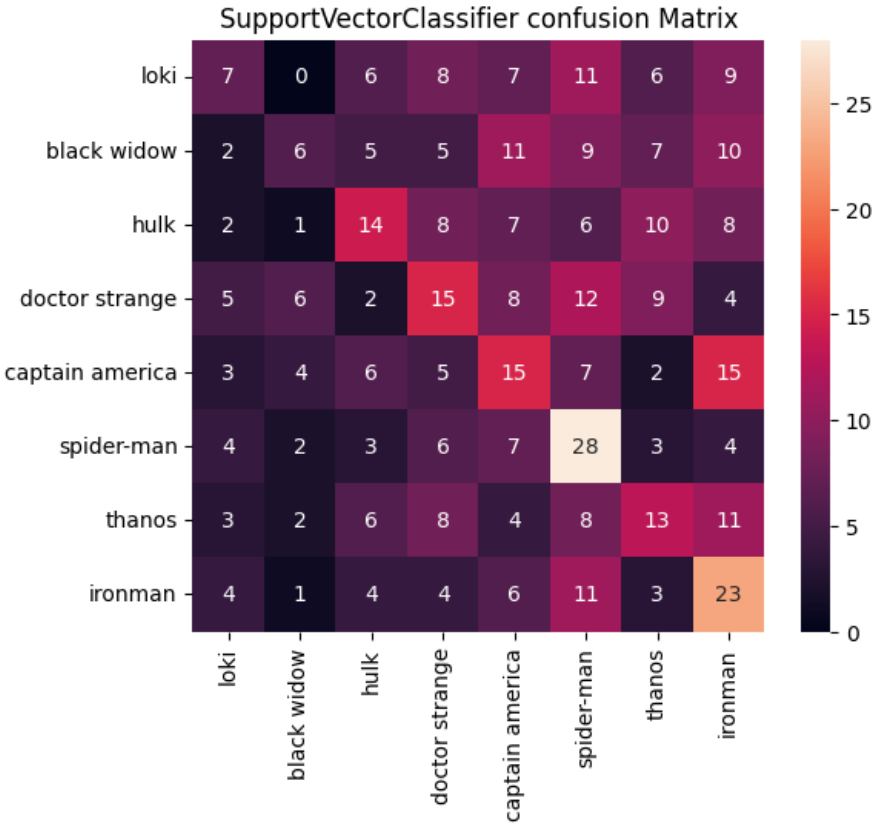

Figure 11: Confusion Matrix: SVC confusion matrix across the labels
Figure shows the confusion matrix for this model. It can be noticed that the label "Spider man" has the highest correlation core while "black widow" has the lowest. The interpreted reason behind this is mentioned in Section 5.1.

5.2.1 Perturbation results - Support Vector Machine Classification

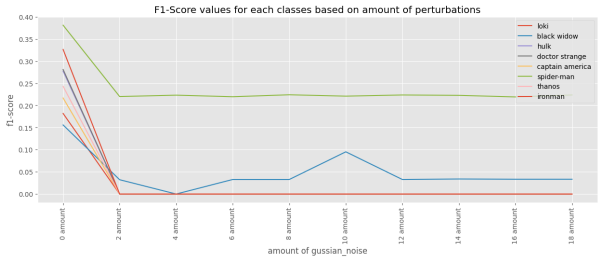

Figure 12: Perturbation - Guassian Noise SVC
From the graph we can observe that as we add guassian noise the F1 score dropped significantly and after that it was constant for all classes except for black widow where the F1 score increased again and decreased after which it became constant.

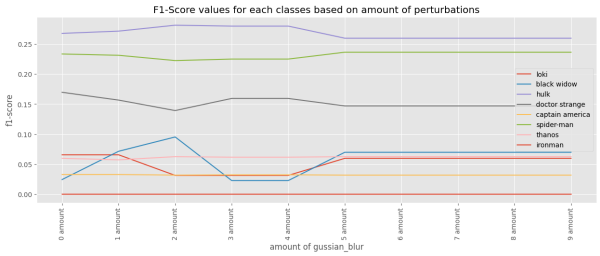

Figure 13: Perturbation - Guassian Blur SVC

From this graph we can observe that increasing the Gaussian Blur had minimal effect on most classes but for black widow there was significant increase and decrease in F1 score. And after amount 5 the F1 score remains constant for all.

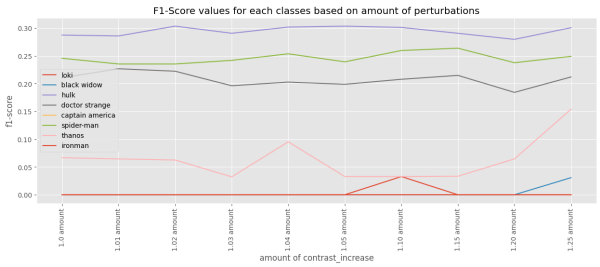

Figure 14: Perturbation - Contrast Increase SVC

From this graph we can observe that increasing the Contrast had minimal effect on most classes but for thanos there was significant increase and decrease in F1 score and it reached it's maximum at amount of 1.25.

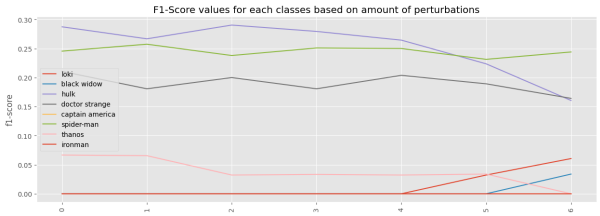

Figure 15: Perturbation - Contrast Decrease SVC

From this graph we can observe that decreasing the Contrast had minimal effect on most classes but for hulk we can observe a decreasing trend in F1 score and there was increase in F1 score for loki after the amount of 4. The pertubrations after 0.4 i.e, 0.3, 0.2 wasn't able to detect the features due to decreased shapness in the images due to which an error was thrown.

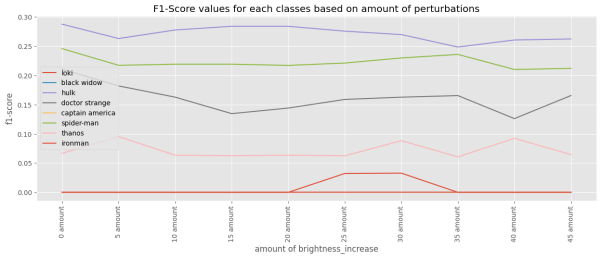

Figure 16: Perturbation - Brightness Increase SVC

From the graph it can be concluded that increasing the amount of brightness has minimal or almost constant effect on the f1-score.

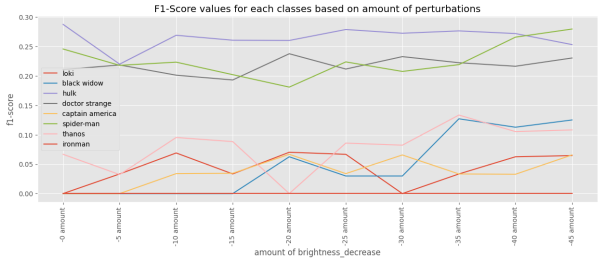

Figure 17: Perturbation - Brightness Decrease SVC
From the graph it can be concluded that on applying brightness decrease filter, we obtain the best result for different class at different setting of the brightness. For instance, for thanos the best f1 score was at -35. This also suggests that using different brightness for different classes, maximum f1 score for the whole dataset can be obtained

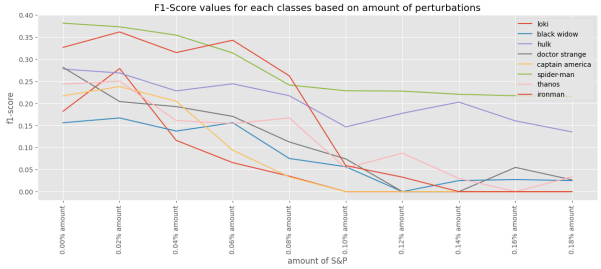

Figure 18: Perturbation - Salt and Pepper SVC
From this graph it can be observed that there is a decreasing tread in the f1 score as the amount of salt and pepper filter is increased.

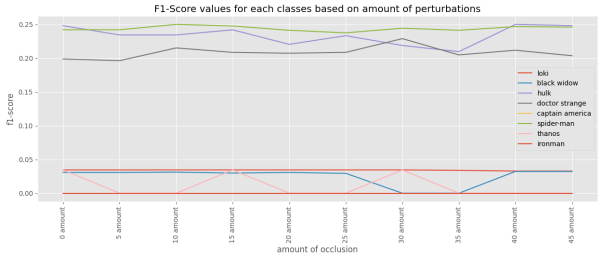

Figure 19: Perturbation - Occlusion SVC
From this it can be observed that, as the amount of occlusion is increased, it does not have a significant effect on F1 score.

6 Limitations & Future Recommendations

After further analysis and understanding the results obtained, we figured that better results could be obtained if we could combine both the classical and deep learning models [1] [2]. We can use the classical models to extract features which can be further trained on the deep learning models which could fetch us better accuracy and results. Apart from this, we also noticed that the dataset given to us was considerably small for proper training and classification for both the models. If we were allowed to combine the both the train and valid datasets for the training model we could have achieved better accuracy and F1-Scores. The dataset also contained a considerable number of outliers which again decreased the overall accuracy of the models. A better dataset would have been easier to train to fetch accurate results.

The run time for the training models were incredibly high and the perturbations for Support Vector Classifier (SVC) took around 30 minutes for each filter. Given the Graphics Processing Unit (GPU) strength of our personal laptops it was very heavy for the system to handle. This was the reason why used the resized image dataset. If we had access to systems with a better GPU we would have been able to use the original images rather than the resized one and we assume that it would get better results due to its better pixel sizes.

7 Conclusions

In conclusion, it can be observed from the results that DenseNet201 model performed much better than SVC. Since DenseNet201 applies Convolution Neural Networks over 201 layers it was able to train and classify the labels better than the n-dimensional classification of SVC. The accuracies obtained by both the models weren't as high due to multiple outliers in the dataset. The outlier removal algorithm proved to be not optimal as it segregated a large amount of images as outliers which diminished the dataset size. The size of the resultant data was not sufficient to train the model. We further checked the robustness of these two models against 8 different filters. The results obtained were very insightful as to how applying these can vary the accuracy of the models. On the whole, this project was a great learning curve to understand the different image classification models and their applications.

Citations and References

References

[1] O. Chapelle, P. Haffner, and V.N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

[2] Shanshan Guo, Shiyu Chen, and Yanjie Li. Face recognition based on convolutional neural network and support vector machine. In *2016 IEEE International Conference on Information and Automation (ICIA)*, pages 1787–1792, 2016.

[3] Mr. Vishal B. Padole and Dr V R Mankar. analysis of knn and svm classifier for image classification: Semantic scholar.

[4] Sidheswar Routray, Arun Kumar Ray, and Chandrabhanu Mishra. Analysis of various image feature extraction methods against noisy image: Sift, surf and hog. *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, 2017.

Appendix

A Code Snippets

In [43]:

```
#SaltnPepper Filter
def ApplyingRandomNoice(amount_):
  _data=[]
  for filepath,label in zip(test_data['Filepath'].values,test_data['Label'].values):
    img = cv2.imread(filepath)
    img = cv2.resize(img,(128,128))
    img = random_noise(img, mode='s&p',amount=amount_)
    img = np.array(255*img, dtype = 'uint8')
    img = np.array(img)/255.0
    _data.append([img,label])
  spX_test = np.array([ x[0] for x in _data])
  spY_test = np.array([Y[1] for Y in _data])
  return spX_test,spY_test

#Gussian Pixel Noise
import math
def Applyinggussainnoise(var):
  _data=[]
  for filepath,label in zip(test_data['Filepath'].values,test_data['Label'].values):
    img = cv2.imread(filepath)
    img = cv2.resize(img,(128,128))
    var = math.sqrt(var)
    img = random_noise(img, mode='gaussian',mean=0, var=var)
    img = np.array(255*img, dtype = 'uint8')
    img = np.array(img)/255.0
    _data.append([img,label])
  spX_test = np.array([ x[0] for x in _data])
  spY_test = np.array([Y[1] for Y in _data])
  return spX_test,spY_test

#Contrastbrightness
def Applycontrastbright(alpha, beta):
  _data=[]
  for filepath,label in zip(test_data['Filepath'].values,test_data['Label'].values):
    img = cv2.imread(filepath)
    img = cv2.resize(img,(128,128))
    img = cv2.addWeighted(img, alpha, np.zeros(img.shape, img.dtype), 0, beta)
    img = np.array(255*img, dtype = 'uint8')
    img = np.array(img)/255.0
    _data.append([img,label])
  spX_test = np.array([ x[0] for x in _data])
  spY_test = np.array([Y[1] for Y in _data])
  return spX_test,spY_test

#Gussianblur
def Applygussianblur(val):
  _data=[]
  for filepath,label in zip(test_data['Filepath'].values,test_data['Label'].values):
    img = cv2.imread(filepath)
    img = cv2.resize(img,(128,128))
    img = cv2.GaussianBlur(img,(3,3),val)
    img = np.array(img)/255.0
    _data.append([img,label])
  spX_test = np.array([ x[0] for x in _data])
  spY_test = np.array([Y[1] for Y in _data])
  return spX_test,spY_test

#Occlusion
def Applyocclusion(square_edge_length):
  _data=[]
  for filepath,label in zip(test_data['Filepath'].values,test_data['Label'].values):
    img = cv2.imread(filepath)
    img = cv2.resize(img,(128,128))
    h, w, _ = img.shape
    img = cv2.rectangle(img, (square_edge_length,square_edge_length), (w //2 , h//2), (
    img = np.array(img)/255.0
    _data.append([img,label])
  spX_test = np.array([ x[0] for x in _data])
  spY_test = np.array([Y[1] for Y in _data])
  return spX_test,spY_test
```

Figure 20: Code snippet for perturbations
This figure gives the code for perturbations in DenseNet201

SVM				
Character	precision	recall	f1-score	support
loki	0.23	0.13	0.17	54
black widow	0.27	0.11	0.16	55
hulk	0.3	0.25	0.27	56
doctor strange	0.25	0.25	0.25	61
captain America	0.23	0.26	0.25	57
spider-man	0.3	0.49	0.38	57
thanos	0.25	0.24	0.24	55
ironman	0.27	0.41	0.33	56
accuracy			0.27	451
macro avg	0.26	0.27	0.25	451
weighted avg	0.26	0.27	0.26	451

Table 1: Classification report for Support Vector Machine Classifier

DenseNet				
Character	precision	recall	f1-score	support
loki	0.41	0.43	0.42	54
black widow	0.47	0.44	0.45	55
hulk	0.51	0.55	0.53	56
doctor strange	0.51	0.46	0.48	61
captain america	0.45	0.4	0.43	57
spider-man	0.73	0.67	0.7	57
thanos	0.4	0.45	0.42	55
ironman	0.47	0.52	0.49	56
accuracy			0.49	451
macro avg	0.49	0.49	0.49	451
weighted avg	0.49	0.49	0.49	451

Table 2: Classification report for DenseNet201