**Comparison of the generalization ability of CNN**

**& SVM with perturbations**

1. **Abstract**

Classification of Marvel characters using two different methods i.e classical and deep learning. For the classical method we use KNN and for deep learning we use a pre-trained model Resnet 18. The training model has been split into training and validation sets for better precision, we then test the models for accuracy and F1 score. The images then go through a set of perturbations and are then put through the models again to measure its accuracy and are compared which model is better.

1. **Introduction**

The dataset consists of 2584 images which are split into training and validation sets with 80-20 split and the test dataset consists of 451 images. The dataset is divided into 8 categories namely, black widow, captain america, doctor strange, hulk, ironman, loki, spider-man, thanos, which are characters from the Marvel movies. The data set was found on kaggle. We train the models to classify these 8 classes and recognize them when given new input.

The world we live in consists of a lot of data in the form of images, speech text and many others. Managing our data has become our priority and a necessity. With Artificial Intelligence and Internet of Things peaking, a lot of data is being generated at large volumes and hence the demand for classification and image recognition. Most of the data we receive from cameras and sensors are unstructured, hence we need to build and advance image recognition techniques like machine learning algorithms. The applications of Image classification are wide, it is used in different fields like agriculture, environmental change, urban planning, surveillance, geographic mapping, disaster control, object detection, satellite images, machine vision etc.

The classical Machine Learning techniques use local descriptors for finding similarities between images. The bag of Visual Words approach is first taken which consists of extracting local features from the input image and cluster them to create visual words. Features are small “interesting”, descriptive or informative patches in any images. Then for each image we make a histogram of its visual words which is used as a global feature to classify it. The three main types of machine learning techniques are unsupervised learning, supervised learning and reinforcement learning classification and analysis. Three classical algorithms are Decision tree, Random Forest and Support Vector Machine. It can model nonlinear class boundaries .

Deep learning is a subset of machine learning which is based largely on Artificial Neural Networks (ANNs), which is a computing paradigm inspired by the functioning of the human brain. It’s about learning across many layers of a neural network accurately, efficiently and without supervision. The Convolutional Neural Network(CNN) are multilayered neural networks which are designed to recognize visual patterns directly from the pixel images, which is a special form of Artificial Neural Network architecture. The main advantage of deep learning in spreading vision-based application is its rapid progressions and improvement in device capabilities like power, memory capacity, power consumption, image sensor resolution have increased the performance and making it cost effective. It allows users with greater accuracy in tasks like image classification, semantic segmentation etc.

1. **Resnet-18**

A residual network, or ResNet for short, is an artificial neural network that helps to build a deeper neural network by utilizing skip connections or shortcuts to jump over some layers which helps build deeper network layers without falling into the problem of vanishing gradients. “One of the main challenges is that it requires a large-scale image data to train a deep learning model from scratch, such as the ImageNet dataset which includes millions of labeled images. Till now, the problem has been addressed by two important methods. The first approach is fine-tuning that takes an already learned model, adapts the architecture, and resumes training from the model weights already trained [8]. Another solution is using a pre-trained deep learning with a large-scale dataset as a fixed feature extractor for small-scale data.” We are using a pretrained model for classifying the dataset into 8 categories. There are different versions of ResNet, including ResNet-18, ResNet-34, ResNet-50, and so on. The numbers denote layers, although the architecture is the same. Resnet18 is a 72-layer architecture with 18 deep layers which aims at enabling large amounts of convolutional layers to function efficiently.

If we keep increasing the number of layers, we will see that the accuracy will saturate at one point and eventually degrade. And, this is usually not caused due to overfitting. Hence, the main concept behind ResNet models is that after starting off with a single Convolutional layer and Max Pooling, there are 4 similar layers with just varying filter sizes – all of them using 3 \* 3 convolution operation, after every 2 convolutions, we are bypassing/skipping the layer in-between. These skipped connections are called ‘identity shortcut connections” and use what are called residual blocks. For the uninitiated, residual neural network (ResNet) is an artificial neural network(ANN) that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks utilize skip connections, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple- layer skips that contain nonlinearities (ReLU) and batch normalization in between.

We trained our dataset with Resnet18 by following the various steps as mentioned below:

1. We import all the required libraries for image classification like torch, torchvision, numpy, matplotlib, openCV, Operating Systems interfaces etc
2. We then define preprocessing of our data, test and train i.e do data augmentation on the database like random horizontal flip, rotation, normalization, resizing etc.
3. Next, we set our training and testing directory path. Our train dataset model will be split into train and valid folders, with an approximate split of 80-20 respectively. We then load the train and valid dataset and also determine our number of classes/ categories which in our case is 8.
4. Then download the resnet18 pretrained model, with our saved pretrained weights. The downloaded model can be modified to tune its layers. We have used only SDG optimizer and Cross-Entropy loss.
5. The model is now ready for training. We train both the train and validation dataset. Using 60 epochs for increasing the accuracy, we load the batch data of images forward the input to get outputs, calculate the predictions and get loss value and update the network weights. The same is done for validation phase using the valid data loader. We run the set number of epochs, printing its loss, accuracy and time value.
6. The same is done with the test dataset, but instead of training the model here we evaluate the model. We run the set number of epochs, printing its loss, accuracy and time value for the testing dataset.
7. We then determine the predictions and labels for the test data. Using this information, we can build a confusion matrix importing it from torchmetrics. Through the confusion matrix we can visualize the classification performance of our model in the heat map from.
8. We obtain our classification report which consists of the precision, recall, f1 score and support. The accuracy of our model is determined to be 67%.
9. Once we’ve got our f1 score for our original test data. We then apply the given perturbations to our test data again to determine its change in accuracy and prediction.
10. Each perturbation at different levels is applied to the test dataset. We then run these new dataset through our testing model and obtain accuracy and F1 score for each individual perturbation and then plot the values for each class/label.
11. **Support Vector Machine**

Support Vector Machines are among the most robust and successful classification algorithms. Finding the best maximum marginal hyperplane (MMH) fit that classifies the dataset is the primary objective of SVM ( i.e maximizing the minimum distance from the separating hyperplane to the nearest example ). “Support vector machines (SVMs) have considerable potential as classifiers of remotely sensed data. A constraint on their application in remote sensing has been their binary nature, requiring multiclass classifications to be based upon a large number of binary analyses.” Although improvements have been suggested to handle multiclass classification instances, the basic SVM only supports binary classification. A way to tackle this is by adding constraints and parameters to the optimization problem to handle the separation of the different classes. When SVM is paired with kernels, its power is greatly increased.

The main working behind the multiclass classification using SVM is firstly to identify keypoints/ local features from an image which can be extracted by using extraction models such as sift. Secondly, the feature descriptors are selected from all identified feature descriptors for every image, filtering the necessary ones. Thirdly, we construct a bag of features to narrow down the features that we will check for the future dataset. All feature points are treated as visual words which are equivalent to the words in the text. K Means is used to cluster all the data. The key features present in the images are represented in the form of a histogram. In the end, we use SVM for our final image classification. SVM are typically anticipated to yield sparse solutions with high generalization capabilities. Moreover, SVM execution time, i.e. test time, depends on solution sparsity, which makes sparsity a desirable property in real applications with many documents to classify. Though the accuracy of the SVM models can be increased by adding batch normalization and residual connection but for now we have used a basic model to comply with our needs in the following manner:

1. We import all the required libraries for image classification like torch, torchvision, numpy, matplotlib, openCV, Operating Systems interfaces etc
2. We then assign directories for our training data. We get the training classes names and store them in a list. Here, we use folder names for class names. Then, saving the path to all our images in a list. (image paths and the corresponding label in image paths). To make it more easily accessible we define a function with the list of filenames.
3. All the descriptors are then stacked into an array and the K-means clustering and vector quantization is performed. We set our K value to 200 for better accuracy. The histogram of features are calculated and represented as vectors. The features are standardized by removing the mean and scaling to unit variance, in a way normalization.
4. We then train the Linear SVM and save it for future reference.
5. The classifier, class names, number of clusters are loaded from the stored pickle file generated while training. Get the path of the testing images and store them in a list, Create feature extraction and keypoint detector objects.
6. All the descriptors are then stacked into an array and vector quantization is performed. The histogram of features are calculated and represented as vectors. The features are standardized by removing the mean and scaling to unit variance, in a way normalization.
7. Report true class names so they can be compared with predicted classes. Perform the predictions and report predicted class names. a confusion matrix is build to understand out accuracy
8. We obtain our classification report which consists of the precision, recall, f1 score and support. The accuracy of our model is determined to be 20.8%
9. Once we’ve got our f1 score for our original test data. We then apply the given perturbations to our test data again to determine its change in accuracy and prediction.
10. Each perturbation at different levels is applied to the test dataset. We then run these new dataset through our testing model and obtain accuracy and F1 score for each individual perturbation and then plot the values for each class/label.
11. **Robustness**

Let’s first begin with understanding each of these perturbations and the seeing their individual impact on the f1 scores after each perturbation is applied and run through the test dataset from resnet18 and svm models that we trained:-

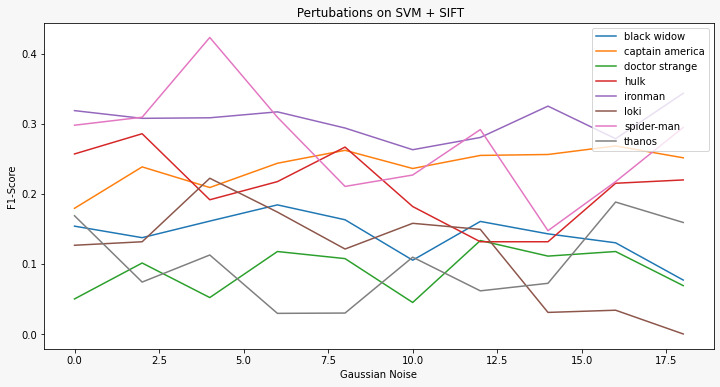
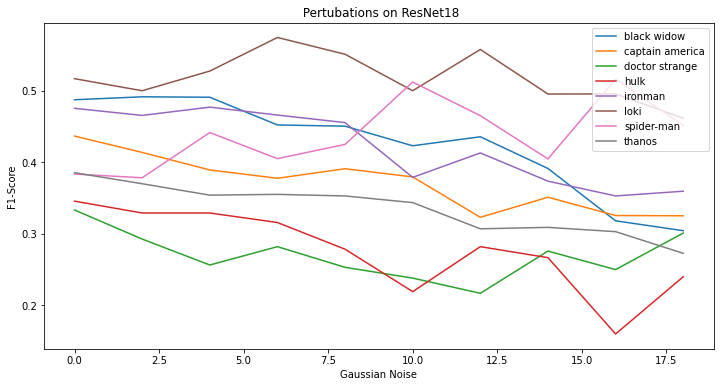
1. **GAUSSIAN PIXEL NOISE**-

It is statistical noise with a probability density function (PDF) equal to the normal distribution. Gaussian noise has a uniform distribution throughout the signal. The probability distribution function for a Gaussian distribution has a bell shape. Although other distributions are feasible, the main advantage of Gaussian noise is that the distribution itself behaves nicely i.e the central limit theorem, which states that the sum of different noises tends to approach a Gaussian distribution, makes it a reliable model. As a result, we adopt a broad strategy and employ a sampling method that augments the visual input x with pixel-wise uncorrelated Gaussian noise.

The gaussian pixel noise is run through the parameters changing the values of standard deviation from the range {0, 2, 4, 6, 8, 10, 12, 14, 16, 18 }.

**Resnet18**- The resnet18 training model after applying perturbations keeps decreasing from 48.7% to 38.5% for no perturbation and decreases to 30.4 to 27.2% for the 10th perturbation. There is a slight increase in the F1 scores when the standard deviation values are small but after that it’s going downwards.

**SVM-**The SVM testing model after applying perturbations varies in the range 31.8 to 5% without any perturbation, decreases and increases for different classes. While some are getting predicted better some are not. There is a variation and no constant pattern can be determined.

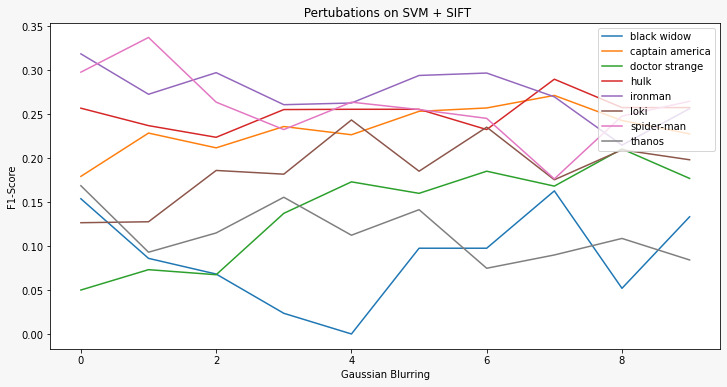
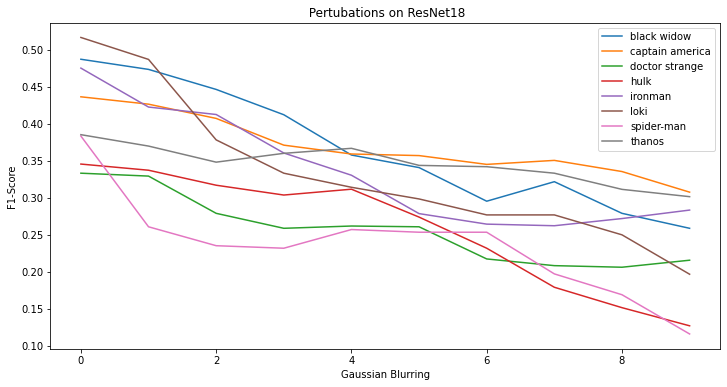


1. **GAUSSIAN BLURRING-**

This operator, a 2-D convolution operator, blurs images and eliminates noise and detail. In contrast to the mean filter, it employs a different kernel that simulates the shape of a Gaussian (or "bell-shaped") hump. It can be considered as a nonuniform low-pass filter that preserves low spatial frequency and reduces image noise and negligible details in an image. This approach substitutes a Gaussian kernel for a box filter. The width and height of the kernel should be either positive and odd. The value of σ controls the variance around a mean value of the Gaussian distribution, which determines the extent of the blurring effect around a pixel. Gaussian noise can be effectively removed from an image using gaussian blurring. A 3x3 mask is used for the gaussian blurring convoling repeatedly from 0 to 9 times.

**Resnet18-** After applying the gaussian blurring, the accuracy of the test data keeps decreasing as the image gets blurrier making it challenging for the model to recognize the classes. The prediction without any mask is from 48.7% to 38.5% and after applying the mask for 9 times, we get the value range as low as 25.8% to 30.1%.

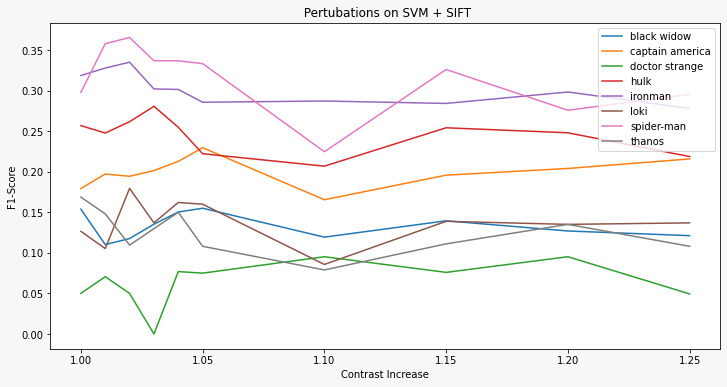
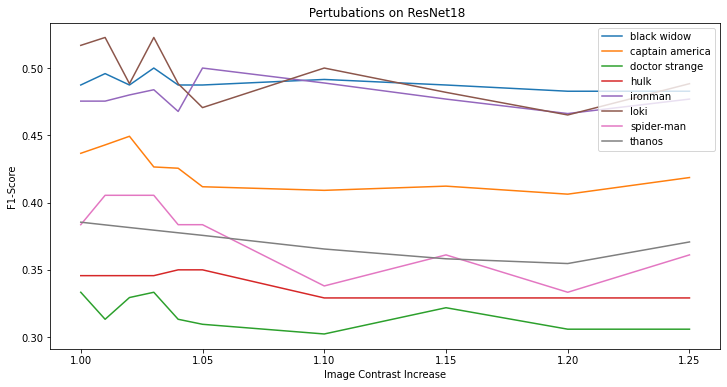
**SVM**- On applying gaussian blurring, the accuracy of the test data is varied. Some classes are getting better f1 scores while the others are getting worse with accuracy values as low as 8.4%. Making it complicated for the model.



1. **IMAGE CONTRAST INCREASE**- The difference between the maximum and minimum pixel intensities in an image can be used to define contrast.By adjusting the value of the max and min intensity pixels, one can alter the contrast of a picture. It will increase the difference between light and dark areas. In simple terms, the light areas will become lighter and bark areas darker. The testing model goes through the image contrast filter for 10 times ranging from { 1.0, 1.01, 1.02, 1.03, 1.04, 1.05, 1.1, 1.15, 1.20, 1.25 } by multiplying each pixel from the given range.

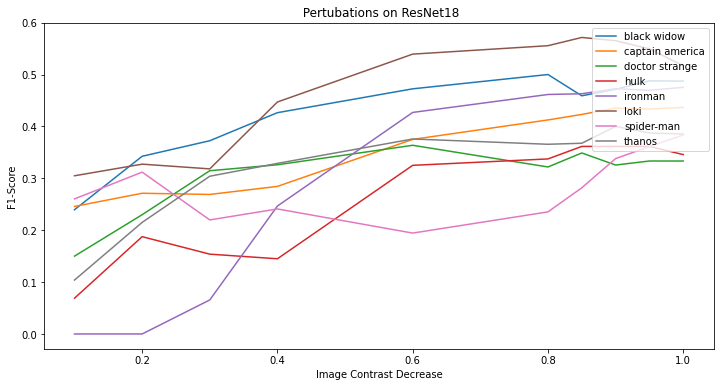
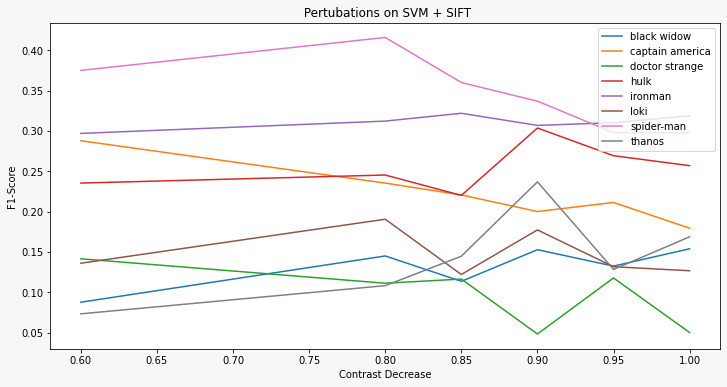
**Resnet18**- the training model does not show much variation in the F1 score compared to the original set of F1 scores for each class. The values vary and increase and decrease by a small margin but does not affect the testing data much as the images are only getting highlighted.

**SVM**- The F1 score is mainly sloping downwards here as the contrast increases it becomes more difficult. The margin is low around 5% comparing the original dataset to the contrast increased dataset.



1. **IMAGE CONTRAST DECREASE**- The difference between the maximum and minimum pixel intensities in an image can be used to define contrast. By decreasing the contrast will decrease the difference, thus softening the images. The same filter is applied but the pixel values are multiplied with values less than 1. The range given is { 1.0, 0.95, 0.90, 0.85, 0.80, 0.60, 0.40, 0.30, 0.20, 0.10} multiplied by each pixel.

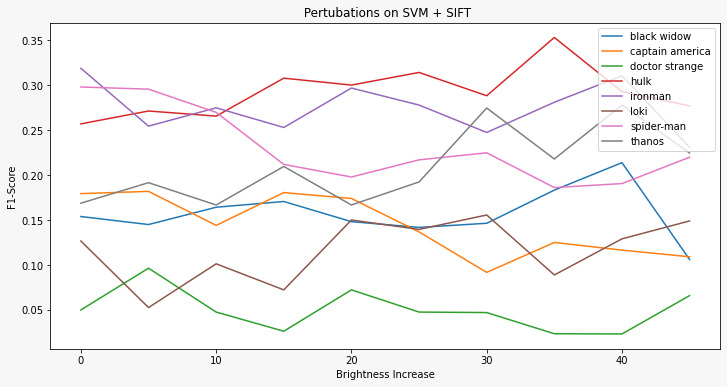
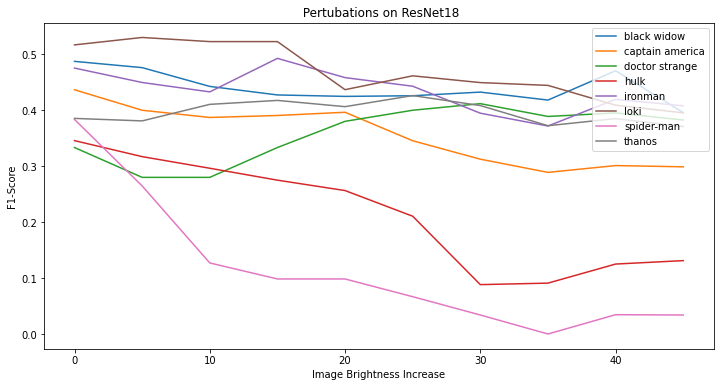
**Resnet18**- While in contrast increase maintains the value more or less, the same does not apply for decreasing the contrast of an image. The values of our F1 score for the predicted value keeps decreasing with the range as low as 26.0% to 6.8% for the last level of perturbation. This happens because the images get darker making it difficult for the model to recognize.

**SVM**- The same applies for decreasing the contrast of an image. The precision decreases as it gets more strenuous for the model to recognize and compute. 

1. **IMAGE BRIGHTNESS INCREASE**- Changing the values of all pixels by a constant, adjusts the brightness of an image. A Positive constant makes an image brighter. By increasing the brightness will light out all colors so the original light ones will become up to white. The image dataset is run by the brightness filter by adding value to each pixel varying from { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 }.

**Resnet18-** The F1 score for the models decrease as the brightness level is increased as the light out all colors hence making the original light tones tend towards white making it harder for models to maintain its accuracy and prediction.

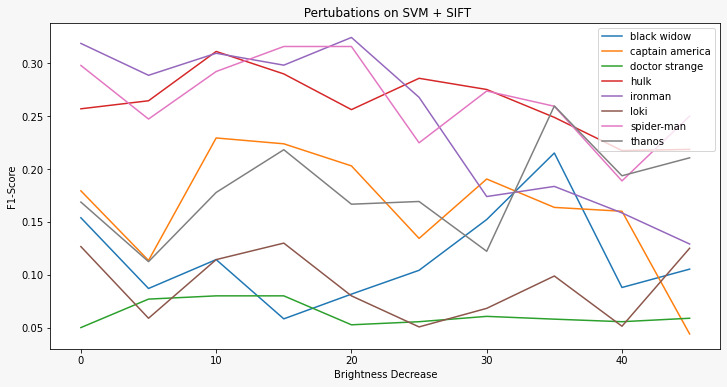
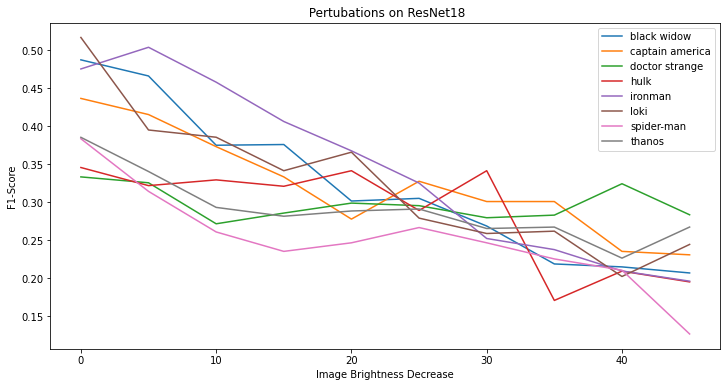
**SVM-** The overall accuracy of this model decreases with the increase in brightness. The accuracy varies from 25% to 4% on the last set of brightness increasing values. Images of hulk shows the highest recognition points indicating more brightness in hulk images.



1. **IMAGE BRIGHTNESS DECREASE**- While a negative constant makes the image darker. Decreasing brightness will darken all colors so the original shaded ones will become up to black. The image dataset is run by the brightness filter by subtracting value to each pixel varying from { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 } thus decreasing the image’s brightness.

**Resnet18**- The same happens to decreasing brightness. The image tends to get darker and thus making the image set difficult for recognition and prediction. The F1 score drops as low as 28.3% to 12.6% for different classes.

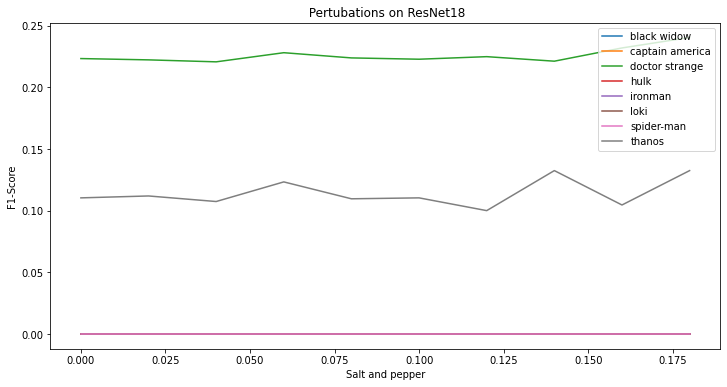
**SVM-** On decreasing brightness, the accuracy of the test data is varied. Some classes are getting better f1 scores while the others are getting worse with accuracy values. This may be because some images are brighter than others resulting in varied values.



1. **SALT AND PEPPER NOISE**- Salt-and-pepper noise is a type of impulse noise where the original value of the pixels is lost and is taken equal to the extremes smin and smax of the dynamic range of the pixel values of an image. For an 8-bit grayscale image, for instance, smin=0 and smax=255. Salt pepper noise of increasing strength varying from {0.00, 0.02, 0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.16, 0.18} are applied on the test dataset

**Resnet18-**  The accuracy is low but does not vary much, it has a low margin of deviation making it tough for the model to recognize the images.

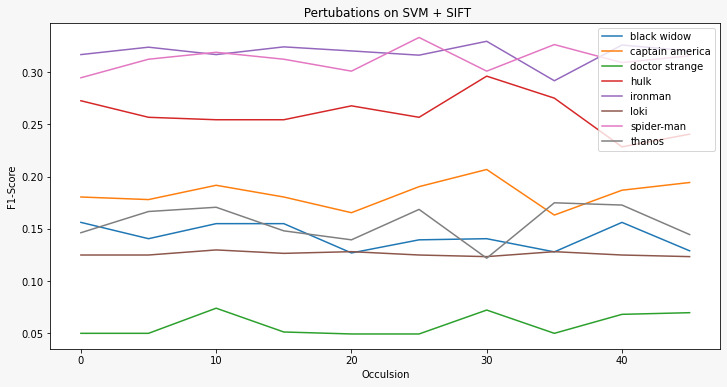
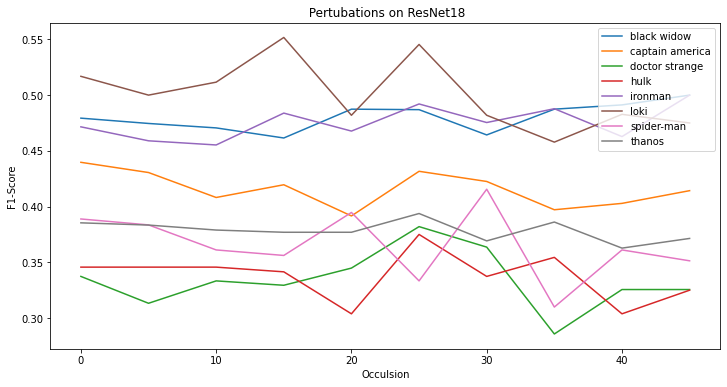
**SVM**- The accuracy is low and decreases as the images gets more noise making it strainful for the model to predict.



1. **OCCLUSION**- Occlusion in an image occurs when an object hides a part of another object. The areas that are occluded depend on the position of the camera relative to the scene. By adding occlusion we place black pixels in a random square region with varying edge lengths as { 0, 5, 10, 15, 20, 25, 30, 35, 40, 45 }.

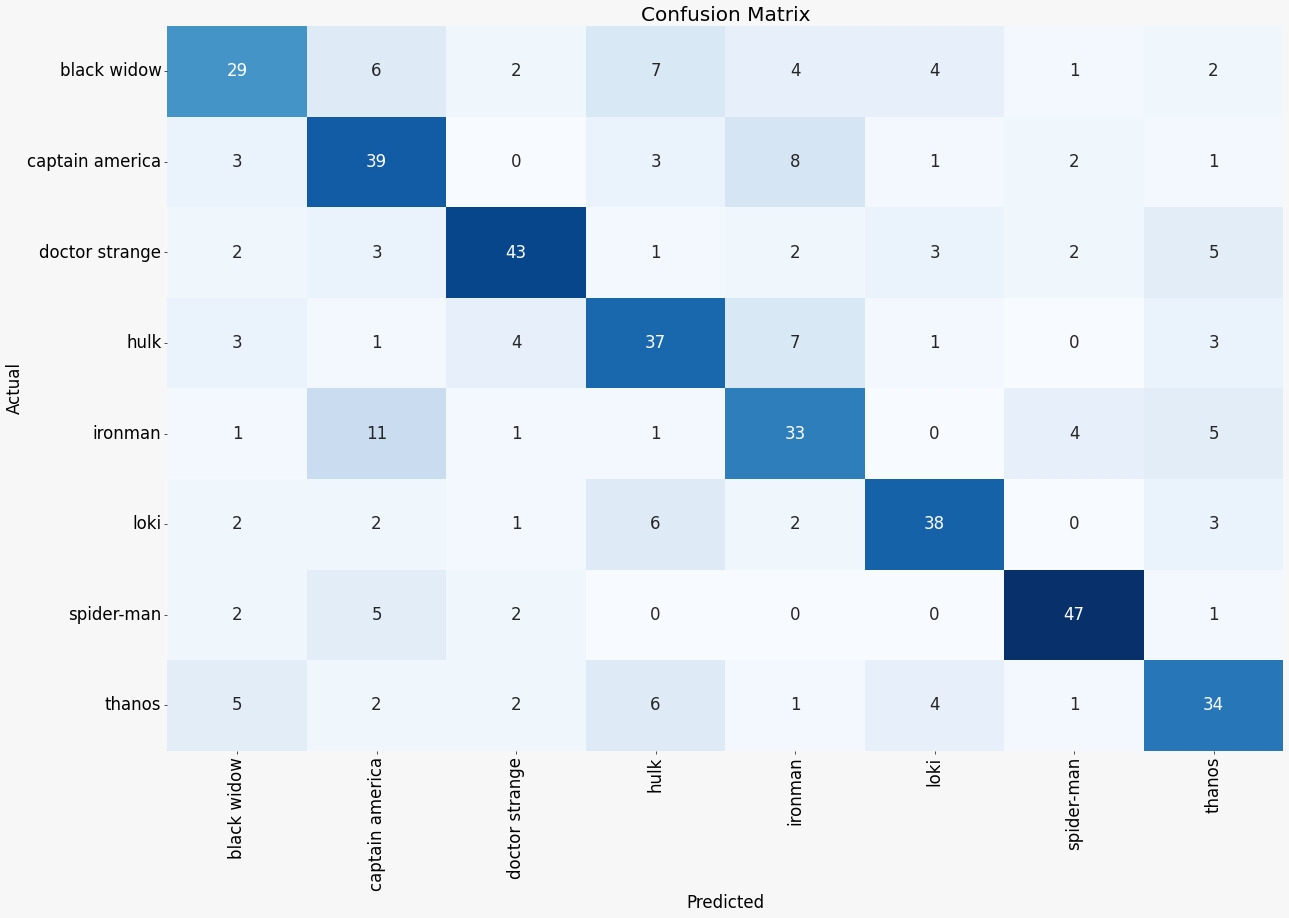
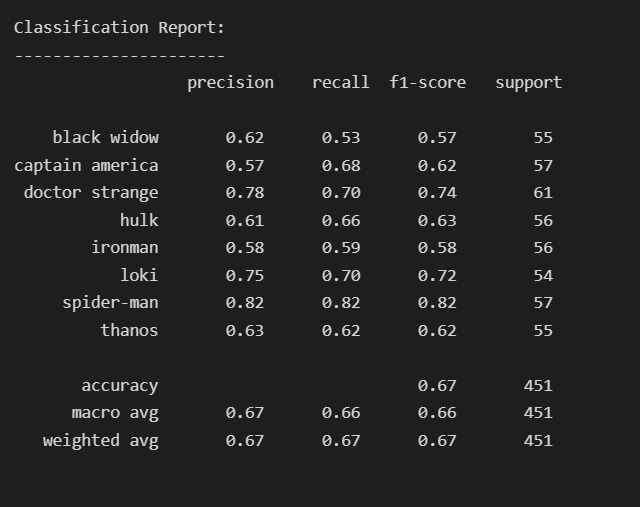
**Resnet18-** The F1 score of the test dataset varies by small margin but mainly increases. The maximum it reaches is 50% accuracy which is better than most of the perturbations. The black spots increases the models accuracy as the model focuses on the

**SVM-** The accuracy of the test data is almost constant. Some classes are getting a little better f1 scores while the others are getting a few values lower based on its different class recognition.

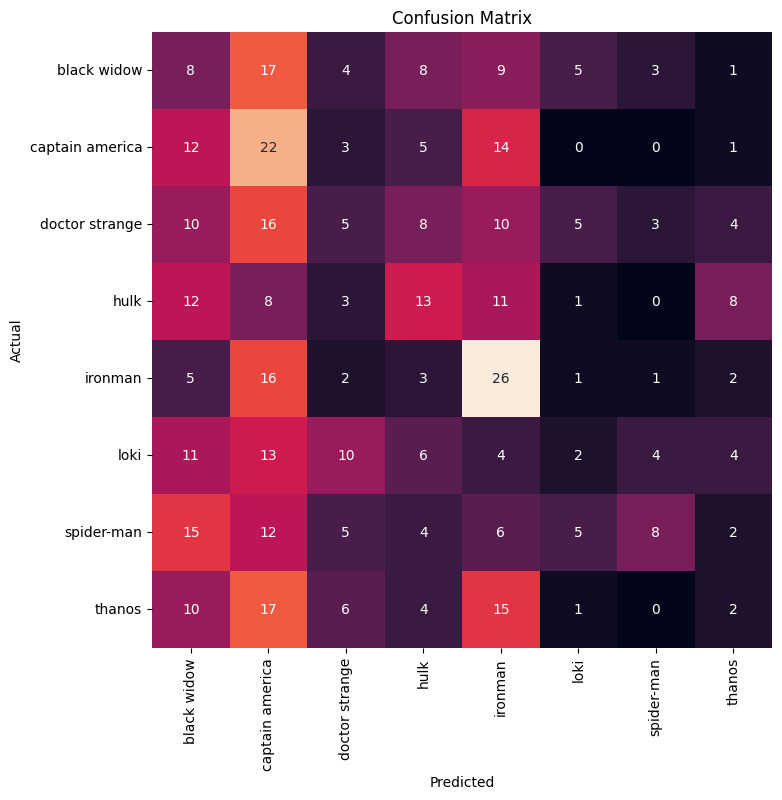
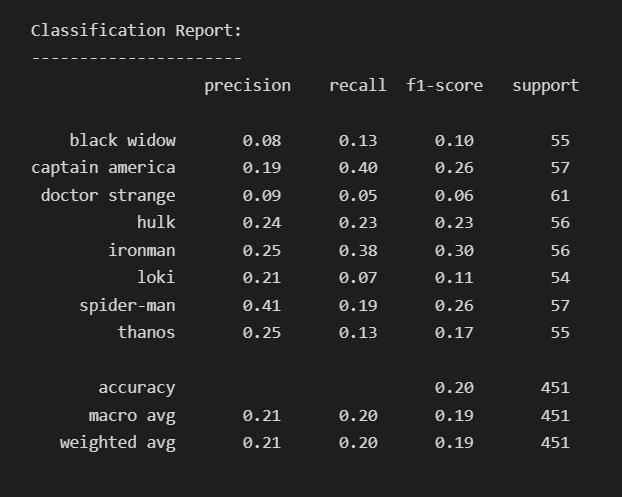


**6. Results and Evaluation**

Given below are the classification report and infusion matrix of the Resnet18 model, with the training set accuracy to 70% and the testing accuracy 67%. The classification also shows the precision and recall for each category. The confusion matrix evaluates the performance of a classification model. The matrix compares the actual target values with those predicted by the machine learning model.



Given below are the classification report and infusion matrix of the SVM using sift and Bag of Visual words model, with the testing accuracy of 67%. The classification also shows the precision and recall for each category. The matrix compares the actual target values with those predicted by the machine learning model.



**Evaluation summary**

From the analysis, after applying each through both the models, we have seen that there is no particular pattern followed by each of these perturbations. But the overall accuracy for the Resnet18 model is better than that of the SVM model. The precision varies as there are some images that are clearer and taken from better angles, while the others are based on brighter or darker lighting positions. The models are hence trained with a larger set so as to get a better grasp of each of the classes.

**7. Conclusion**

We assessed the suggested method using SVM and a deeper model Resnet18 on the marvel dataset consisting of 2000 images in the training set and 584 images in the validation set. The test dataset consists of approximately 450 images. The dataset consists of 8 Marvel characters. The training and validation dataset was evaluated and tested for both models giving an accuracy of 67% and 20.8% for Resnet and SVM respectively. Hence from the results, it can be observed that Resnet18 gives a better accuracy than SVM and gives an overall better performance. Resnet18 shines when there is a huge amount of data, as in the marvel dataset. It is also very useful in finding non-linear correlations. SVM, however suffers in predicting multilayer of features/classes since it is a non-probabilistic binary linear classifier.We also investigated robustness comparisons by varying parameters in each of the two classifiers, we found that in each case Resnet18 showed faster and more accurate results, while the training of Resnet18 was much longer than that of SVM, indicating that deeper models are better classifiers than the simple classifiers and lead to an overall better performance and is time efficient. Hence making the CNN models more widely used and state-of-the art method of classifying multiclass images.

**References**

1. L. Pan, S. Pouyanfar, H. Chen, J. Qin and S. -C. Chen, "DeepFood: Automatic Multi-Class Classification of Food Ingredients Using Deep Learning," 2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC), 2017, pp. 181-189, doi: 10.1109/CIC.2017.00033.
2. G. M. Foody and A. Mathur, "A relative evaluation of multiclass image classification by support vector machines," in IEEE Transactions on Geoscience and Remote Sensing, vol. 42, no. 6, pp. 1335-1343, June 2004, doi: 10.1109/TGRS.2004.827257.
3. Mohapatra, J., Weng, T. W., Chen, P. Y., Liu, S., & Daniel, L. (2020). Towards verifying robustness of neural networks against a family of semantic perturbations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 244-252).
4. <https://medium.com/nerd-for-tech/image-classification-using-transfer-learning-pytorch-resnet18-32b642148cbe>
5. <https://github.com/bnsreenu/python_for_microscopists/blob/master/069a-Train_BOVW_V1.0.py>
6. <https://github.com/bnsreenu/python_for_microscopists/blob/master/069b-Validate_BOVW_V1.0.py>
7. <https://www.maths.nottingham.ac.uk/plp/pmzkz/saltpepper.html>
8. <https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html>