NU302 R&D Progress Report
Image Classification
NIIT University

INSTRUCTOR: Dr. Prashant Srivastava
SUBMITTED BY: Anirudh Sharma, Bollam Sreekar Reddy,
Potlacheruvu Sai Krishna Vamsi, Shravan Sridhar and Tavva G N R S N Prudhvith

Table of Contents

List of Figures

List of Tables

Rationale/Abstract

On the surface, teaching a computer to do something like image classification seemed very intriguing to us. Moreover, there are countless real-world applications of this concept. It is in light of these reasons that we decided to work on Image Classification.

It is important to note here that we started off with very little knowledge. Thankfully though, this topic has been well-researched by the scientific community and we didn't break a sweat finding resources to learn from. So naturally, we perused a bunch of research papers that dealt with image classification, each from a different perspective. We then decided to actually implement image classification on a small-scale with the limited hardware we were in possession of. As difficult as it was, we started with SVM and a very small dataset to achieve an accuracy of 93%. Although SVM is a very strong technique, achieving such a high accuracy is still an anomaly. We realized that our results boasted such a high accuracy due to the lack of a large enough dataset. So, using data augmentation, we more than tripled the size of our dataset. On performing SVM now, we achieved an accuracy of 82%, a significant decrease. Unsatisfied with the results, we decided to move to other deep learning techniques. This quest led us to Neural Networks and in particular, CNN. On successfully implementing CNN, we achieved an accuracy of a staggering 93.57% on the very same dataset. This stands as a testimony to the increased potential of deep learning techniques over the more traditional machine learning techniques.

Certificate

This is to certify that the present research work titled *Image Classification* being submitted to NIIT University, Neemrana, Rajasthan, in the fulfillment of the requirements for the course *NU302: R&D Project* at NIIT University, Neemrana embodies authentic and faithful record of original research carried out by **Anirudh Sharma, Bollam Sreekar Reddy, Potlacheruvu Sai Krishna Vamsi, Shravan Sridhar and Tavva G N R S N Prudhvith** of B. Tech (CSE) at NIIT University, Neemrana. They have worked under our supervision and that the matter embodied in this project work has not been submitted, in part or full, as a project report for any other course of NIIT University, Neemrana or any other university.

Dr. Prashant Srivastava
Assistant Professor (CSE)
NIIT University

Objectives

- To develop an understanding of image classification from the ground-up, i.e, start with the traditional machine learning techniques first and then move onto deep learning techniques.

- To research the underlying principles and techniques of image classification.

- To highlight the potential of deep learning techniques and illustrate their importance over the more traditional machine learning techniques.

Review of Literature

Paper 1

Title: A Survey on Image Classification Methods

Authors:

- Jipsa Kurian
- V. Karunakaran

Publication: IJARECE 2012, Volume 1, Issue 4, October 2012

ISSN: 2278-909X

Abstract: Image classification is one of the most complex areas in image processing. It is more complex and difficult to classify if it contains blurry and noisy content. There are several methods to classify images and they provide good classification result, but they fail to provide satisfactory classification result when the image contains blurry and noisy content. The two main methods for image classification are supervised and unsupervised classification. Each classification has its own advantage and disadvantage. It is difficult to obtain better result with the noisy and blurry image than with normal image. The main aim of literature survey is to provide a brief overview about some of most common image classification method and comparison between them. Finally, it has shown that Self Organizing Tree Algorithm, an unsupervised classification method classify the images to 81.5% even it contain blurry and noisy content. Hence proved that it is the best classification method.

Summary: The aim is to provide a brief overview of some of the most common image classification methods and comparing the results of these methods. Mainly, all the classification techniques are classified as Supervised and Unsupervised. Explaining the different steps in each of these, here is a figure:

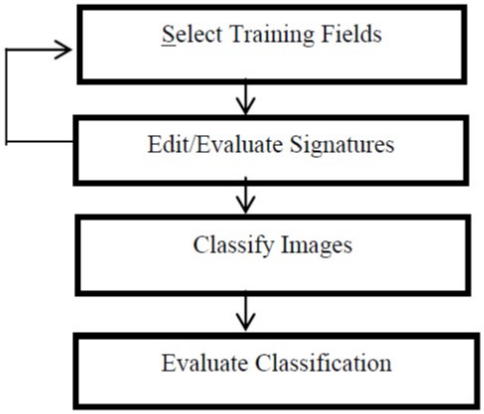


Fig 1. Supervised Classification

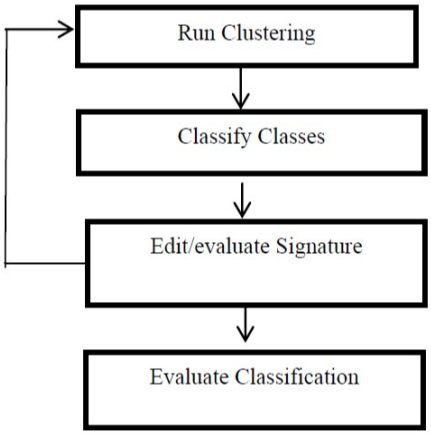


Fig 2. Unsupervised Classification

It is seen that better results we obtained using the Unsupervised techniques. The methods used in supervised classification are Bag of Regions with an accuracy of 62%, Linear Discriminant Analysis (LDA) with an accuracy of 70%, Support Vector Machine (SVM) with an accuracy of 71.10% and Artificial Neural Network (ANN) with an accuracy of 72.50%. Whereas the techniques used in unsupervised classification we have Self Organizing Tree Algorithm with an accuracy of 81.95%.

The paper discussed different image classification techniques for classifying blurry and noisy images. It even discussed the advantages and disadvantages of supervised and unsupervised classification. Finally a comparison of all the methods was done and concluded that Self Organizing Tree Algorithm is best one to classify images even when it contains blurry and noisy content with a very little computational cost.

Paper 2

Title: A Survey on Image Classification Approaches and Techniques

Authors:

Pooja Kamavisdar

Sonam Saluja

Sonu Agrawal

Publication: IJARECE 2013, Volume 2, Issue 1, January 2013

ISSN: 2319-5940

Abstract: Object Classification is an important task within the field of computer vision. Image classification refers to the labelling of images into one of a number of predefined categories. Classification includes image sensors, image pre-processing, object detection, object segmentation, feature extraction and object classification. Many classification techniques have been developed for image classification. In this survey, various classification techniques are considered: Artificial Neural Network(ANN), Decision Tree(DT), Support Vector Machine(SVM) and Fuzzy Classification.

Summary: This paper attempts to study and provide a brief knowledge of the different image classification approaches and different classification methods. The survey also provides the advantages and disadvantages of the different methods. The paper also discussed different approaches based on the different types of inputs, some of which are: characteristics (shape-based and motion-based), training samples (supervised and unsupervised), assumption of parameter on data (parametric and non-parametric), pixel information (per-pixel, sub-pixel, per-field and object oriented), number of outputs for each spatial element (hard and soft), spatial information (spectral, contextual and spectral-contextual) and multiple classifiers.

Paper 3

Title: Review of Image Classification Methods and Techniques

Authors:

Maneela Jain

Pushpendra Singh Tomar

Publication: IJERT 2013, Volume 2, Issue 8, August 2013

ISSN: 2278-0181

Abstract: Unsupervised classification has become one of the most challenging areas in image processing. Data, object and image classification is a very important task in image processing. If any image has noisy content or it contains blurry data, it becomes a very difficult task to classify these images. Digital image processing introduces many techniques for classification but if the image is blurry/noisy, they cannot provide satisfactory results. In this survey paper three classification methods are considered, namely Supervised Learning, Unsupervised Learning and Semi-Supervised Learning. The main motive of this literature survey is to give a brief comparison between the different image classification techniques and methods. We realize by the end that Semi-Supervised Biased Maximum Margin Analysis classifies images more accurately even if they contain blurry or noisy image.

Summary: The previous papers discussed the basic approaches to image classification. Going through them gave us enough insight to move onto more advance techniques and the image classification techniques explored in this survey paper are:

- **Artificial Neural Network** - Supervised Learning
- **SVM** - Supervised Learning
- **DAG-SVM** - Supervised Learning
- **Fuzzy Decision Trees** - Unsupervised Learning

The three classification methods that the paper talks about:

- **Supervised learning**: When dealing with labelled data.
- **Unsupervised Learning**: When dealing with unlabeled data.
- **Semi-supervised Learning**: When the training data contains combination of labelled and unlabelled data.

Model	Description	Disadvantages
Artificial Neural Network	ANN is a collection of layers it generally contains input, hidden and output layers. These layers are joined by weighted links.	Choosing the right network architecture is often difficult.
DAG-SVM	An extension of SVM technique for multiclass classification.	Performance of result evaluation shows that DAG-SVM is not a better classifier.
FDT	FDT uses the advantages of both the methods i.e. Fuzzy and decision tree.	No training so prior knowledge about the desired area is required.

Table 1. The Different Models Discussed

Paper 4

Title: Review of Image Classification techniques

Authors:

- Nupur Thakur
- Deepa Maheshwari

Publication: IJERT 2017, Volume 4, Issue 11, November 2017

ISSN: 2395-0072

Abstract: Image classification is an important tool for extracting information from digital images. The aim of this paper is to summarize information about few image classification techniques. The paper also elaborates different categories of image classification techniques. The image classification techniques considered in this paper are Parallel-Piped Technique, Minimum Distance Technique, Maximum Likelihood (ML) Technique, Artificial Neural Networks (ANN) and Support Vector Machine (SVM).

Summary: The paper tries to explain different techniques for Image classification and categorizing them into supervised and unsupervised techniques. The accuracies and other measuring parameters of these techniques majorly depends on the data used though. This paper helps the researches in selecting the most appropriate classification techniques based on their requirements. Many times you have different factors to be considered before you actually consider a method or techniques to solve a paper. There are many trade-offs. This paper gives a simple summary of each of these techniques in order to help you choose a better technique based in your requirement.

Methodology

The following diagram describes the different steps of image classification.

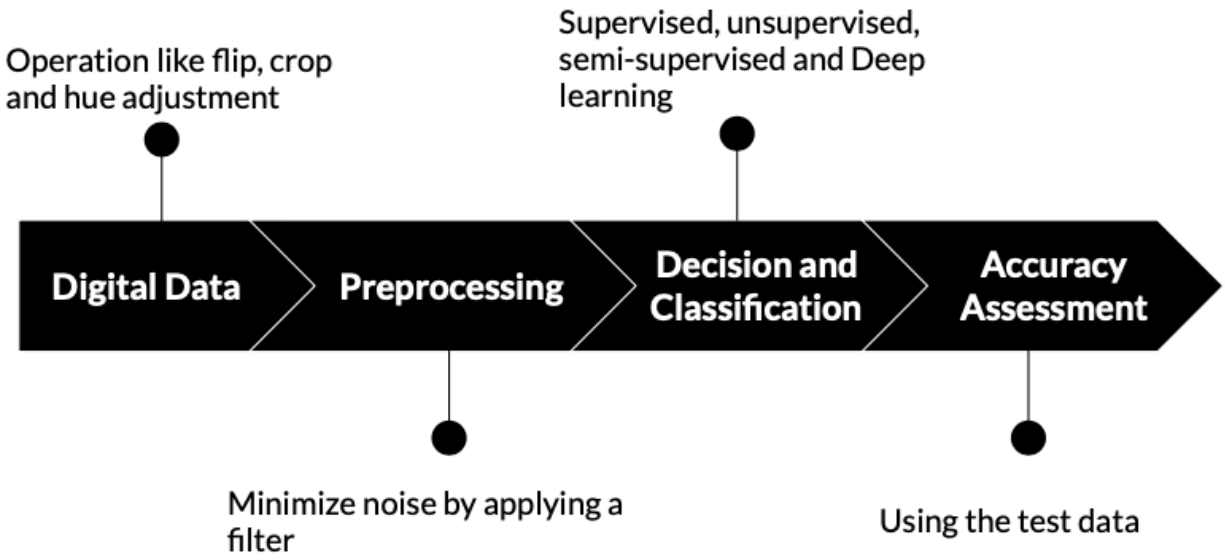


Fig 3. The Proposed Methodology

Digital Data

At the start of the project a dataset of 350+ images were collected and all the images were intact with very little noise. For the first implementation that is using the SVM for Image classification, as it is a machine learning technique this dataset was sufficient to see decent results. We acquired an accuracy of 93%. But, these results were not observed when images were a little blurred or have a brighter background. So, it was decided that we would use CNN for the same. The major problem with the dataset if we have to implement CNN is, it requires a lot of data for training the model. So, we explored some of the data augmentation techniques to increase the number of images in the dataset. A few more images were added as well. Making a total of 3000+ images. The different data augmentation techniques we used are:

- Changing the color space to CMY color space from RGB color space.

- Dithering of the image
- Creating Primary Colors Version of the image
- Random translation of the image
- Flipping the image Vertically
- Flipping the image horizontally

Conversion to CMY color space: The images in the dataset are in RGB color space and all the 350+ images are converted to CMY color space. It was done using the user defined function which is show below.

```
#Converting an image to CMY
def convert_CMY(image):
    width, height = image.size
    new = create_image(width, height)
    pixels = new.load()
    for i in range(width):
        for j in range(height):
            pixel = get_pixel(image, i, j)
            red =   255 - pixel[0]
            green = 255 - pixel[1]
            blue =  255 - pixel[2]
            pixels[i,j] = (int(red), int(green), int(blue))
    return new
```

The image on the left (Fig. 4) is converted to the image on the right(Fig. 5) after passing through the above function.

Dithering of the image to add noise to the image: The images in the dataset are dithered using the user defined function shown below. This process in specific was performed in order to train the model to detect and classify the image even when there is noise in the image.

```
#Dithering an Image
def convert_dither1(image):
    width, height = image.size
    new = create_image(width, height)
    pixels = new.load()
    for i in range(1,width-1):
        for j in range(1,height-1):
            pixel = get_pixel(image, i, j)
            red = pixel[0]
            green = pixel[1]
            blue = pixel[2]
            gray = (red * 0.299) + (green * 0.587) + (blue * 0.114)
            if(gray > 127):
                p = 255
            else:
                p = 0
            pixels[i, j] = (int(p), int(p), int(p))

    return new
```

The image on the left (Fig. 6) is converted to the image on the right(Fig. 7) after passing through the above function.

Creating Primary Colors Version of the image: The images in the dataset are converted to primary colored images using the following user defined function shown below. The primary colors are the red blue and green so, for any specific pixel, we take a threshold value of 127 and if the value of red, green or blue is greater than that then, it you assign a value of 255 to the same and 0 in all other cases to that specific value of R, G and B respectively.

```
#Converting the image to primary
def convert_primary(image):
    width, height = image.size
    new = create_image(width, height)
    pixels = new.load()
    for i in range(width):
        for j in range(height):
            pixel = get_pixel(image, i, j)
            red =   pixel[0]
            green = pixel[1]
            blue =  pixel[2]

            if(red > 127):
                red = 255
            else:
                red = 0
            if(green > 127):
                green = 255
            else:
                green = 0
            if(blue > 127):
                blue = 255
            else:
                blue = 0
            pixels[i, j] = (int(red), int(green), int(blue))
    return new
```

The image on the left (Fig. 8) is converted to the image on the right (Fig. 9) after passing through the above function.

Random translation of the image: Every image in the dataset is translated or moved by random number of pixels ranging from 0 to 30. These pixels are colored black in order to give the moved feeling and making it a totally different image for the model as it captures different feature this time. The code for the user defined function can be seen below.

```
#Image translation
def image_translation(image):
    a = 1
    b = 0
    c = random.randint(1,10000)%30
    d = 0
    e = 1
    f = random.randint(1,10000)%30
    image = image.transform(image.size, Image.AFFINE, (a, b, c, d, e, f))
    return image
```

The image on the left (Fig. 10) is converted to the image on the right(Fig. 11) after passing through the above function.

Flipping the image Vertically: Flipping the image vertically using the user defined function shown below.

```
#Flippling Image Vertically
def flip_vertically(image):
    image = image.transpose(Image.FLIP_LEFT_RIGHT)
    return image
```

The image on the left (Fig. 12) is converted to the image on the right(Fig. 13) after passing through the above function.

Flipping the image Horizontally: Flipping the image horizontally using the user defined function shown below.

```
#Flippling Image Horizontally
def flip_horizontally(image):
    image = image.rotate(180)
    return image
```

The image on the left (Fig. 14) is converted to the image on the right (Fig. 15) after passing through the above function.

The driver function used:

```
#driver for flip_vertically
images = glob.glob("*.jpg")
for image in images:
    img = Image.open(image)
    image = image[:-4]
    img = flip_vertically(img)
    img.save("Vertically Flipped\\"+image+"vf.jpg")
```

The above code is the driver function we used for each and every function we have. This actually selects all the images in the present directory and perform the operation required using the function and then, rename it and save it in a sub folder.

*** To show the different functions we used, only one image was put up and the variations are put in to show the

Preprocessing

At the start of the project a dataset of 350+ images were collected and all the images were intact with very little noise. Every image we get from the dataset goes through the different processes mentioned in the previous section to form more than 7-8 images with all the operations mentioned. This makes a total of more than 3000+ images in the dataset. This dataset is then loaded into the python using a user defined function which first converts all the images to 64 by 64 and then flattens all the images using the numpy library functions resulting in an array of the numbers for each image. These images or the array of arrays are then used for processing and classification. This type of data is very useful because, it is digital images converted to numbers and these can be easily processed be it be for image classification using SVM, CNN or any other technique.

Decision and Classification

Introduction:

As mentioned before, we began with the more traditional machine learning techniques for classification. After having gone through most of them, we chose to implement SVM. Although it made more sense to implement Neural Networks (NN), the more modern approach, we felt that SVM would be a better starting point and we weren't wrong. Now before we begin with what SVM is and how we implemented it, we think we require to explain why we thought that NN should've been the way to go.

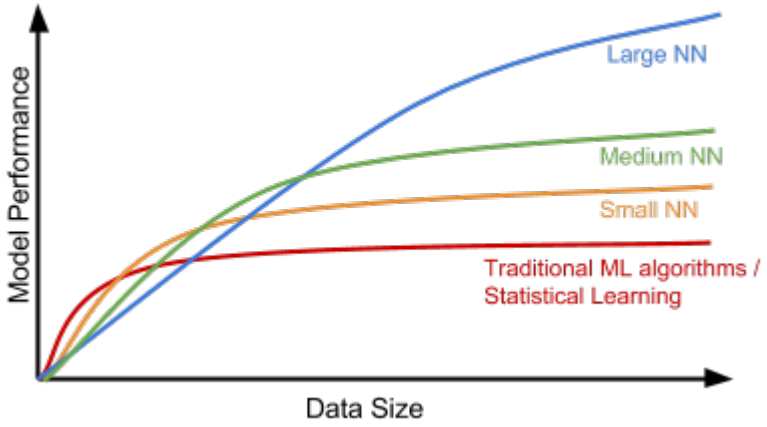


Fig 16. The Shift to NN

Accuracy of Image classification using NN far exceeds any machine learning algorithm and even other deep learning algorithms . More importantly, it's ability to learn features on its own puts it in a league of its own. The above figure illustrates exactly this. The performance of traditional machine learning algorithms have a saturation point and feeding more data beyond this point is not going to result in increased performance. This is where NNs shine. Moreover, the features given to an NN model are just the pixel values of the image. So it creates a layer of abstraction for people without a lot of knowledge on image processing. Now that we've expressed the importance of deep learning algorithms, we can move on to SVM.

SVM:

When we began understanding this technique, we were, and this might not come off as a surprise, overwhelmed. Note here that SVM is a very strong classification method by its own merit. But as mentioned before, it does not stand a chance in a data rich world. On further research, dividing the topic into three sub-topics helped us understand more efficiently. The sub-topics are:

- Visualization
- Tuning Parameters
- Implementation and Analysis

So let us begin with the first sub-topic, i.e., **Visualization**. Given a labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. Simply put, the following happens:

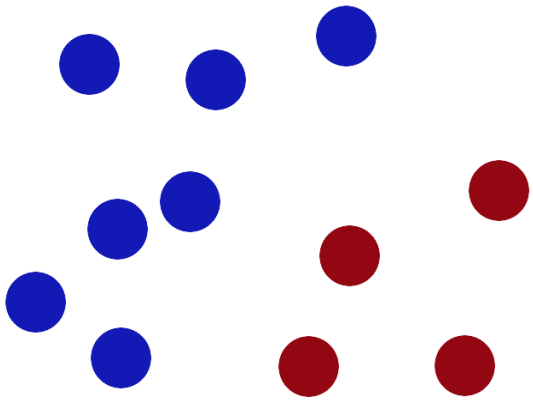


Fig 17(a). Initial Data

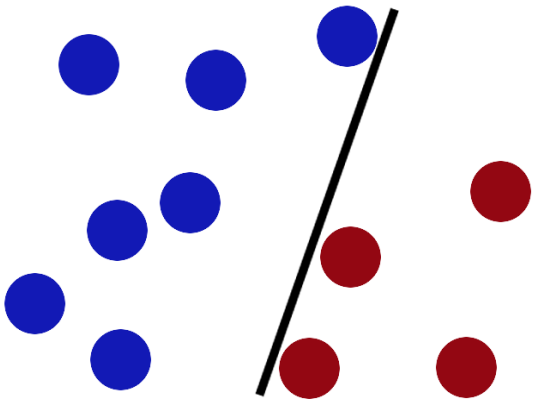


Fig 17(b). Simplest Classification

Say we have to create a classifier to distinguish the blue data points from the red. The simplest way to do so would be to create a line between the two.

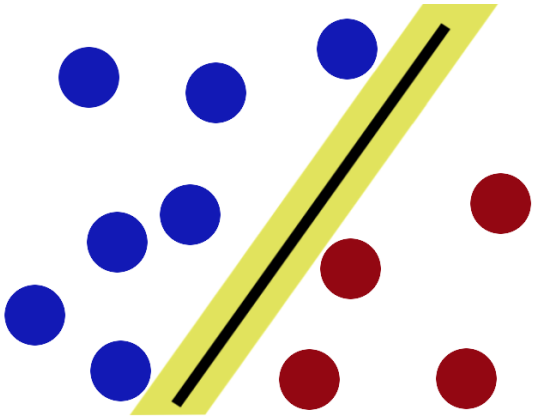


Fig 18(c). Most Efficient Classifier

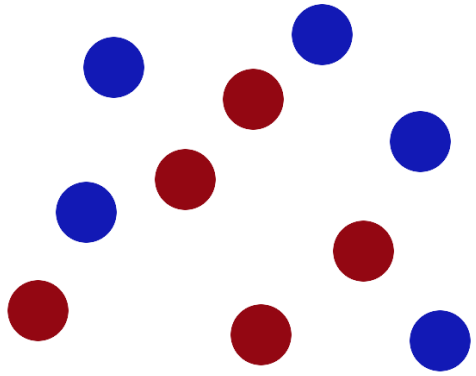


Fig 18(d). What Now?

Although we have made a classifier, it is not the most efficient. Using something called *the biggest gap trick optimization*, SVM finds out the most efficient classifier [Fig (c)]. But what if there is no line that can possibly classify the data points [Fig (d)]?

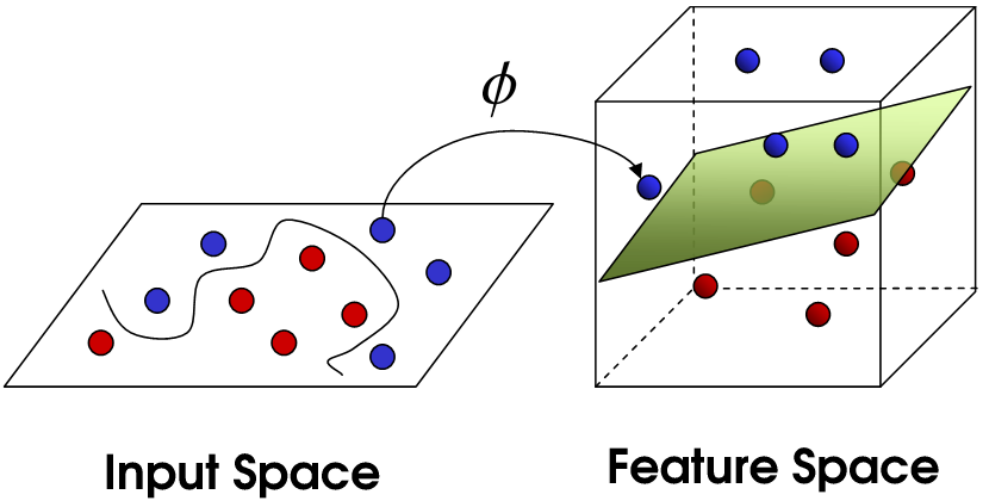


Fig 19(e). SVM

In such a scenario, SVM does something called *kernelling*. Now there are many different kernels which we'll talk about later. For now, let us consider that kernelling lets us visualise the data points in 3D space. That way, we can see that the data points can be

classified by using a plane this time, instead of a line. This plane is called *hyperplane*. On viewing the input space as is, we see that the hyperplane is now visible to us a curvy separator. Also, the data points closest to the hyperplane are called the *support vectors*. They are called so because only these points in the end contribute to the calculation of the most efficient classifier. The other points do not help. So in conclusion:

- *Hyperplanes*: It refers to the line after transformation/kernelling.
- *Kernels*: When drawing a line to classify was no longer an option, we were forced to transform and introduce a z-axis. Such transformations are called *kernels*.
- *Biggest Gap Trick Optimization*: Finding the most optimized position of the hyperplane such that we achieve the widest possible margins.

Now that we've visualized SVM, let us talk about the different **tuning parameters**.

1. **Gamma:** This parameter defines the influence of a single training example. (Gamma α [points closer to plausible line])

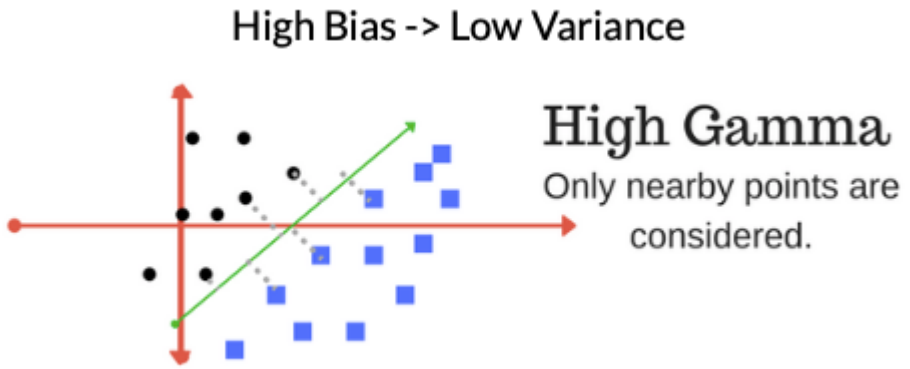


Fig. 20(a)

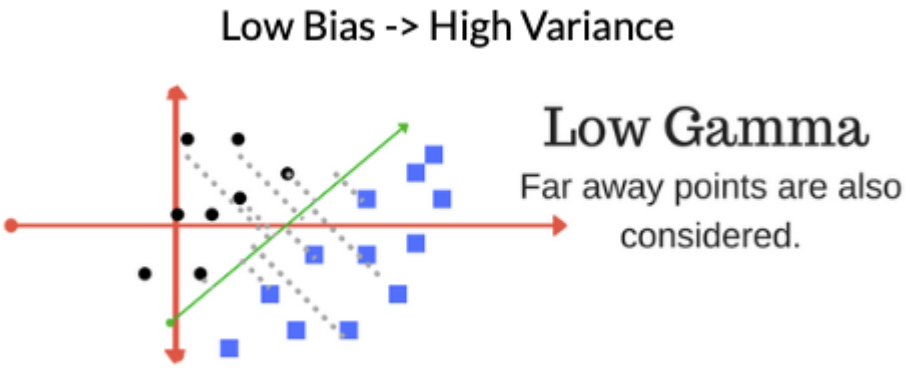


Fig. 20(b)

2. **Regularization Parameter:** This parameter tells the SVM how much you want to avoid misclassifying each training example. It is represented by the variable C in all our implementations (C **α** [penalty for misclassification]).

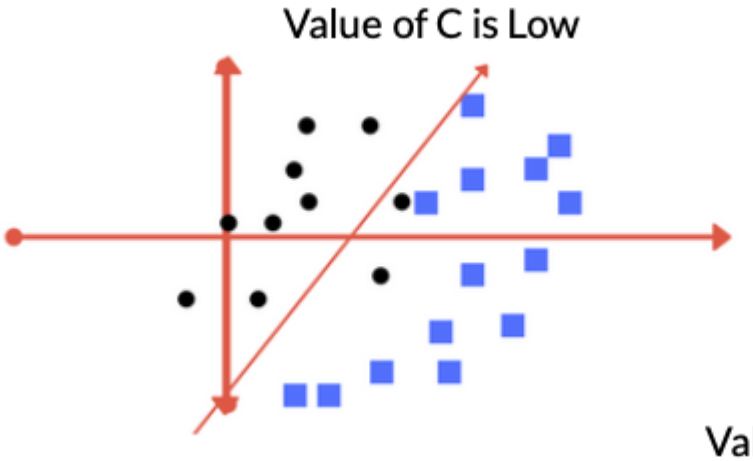


Fig. 21(a)

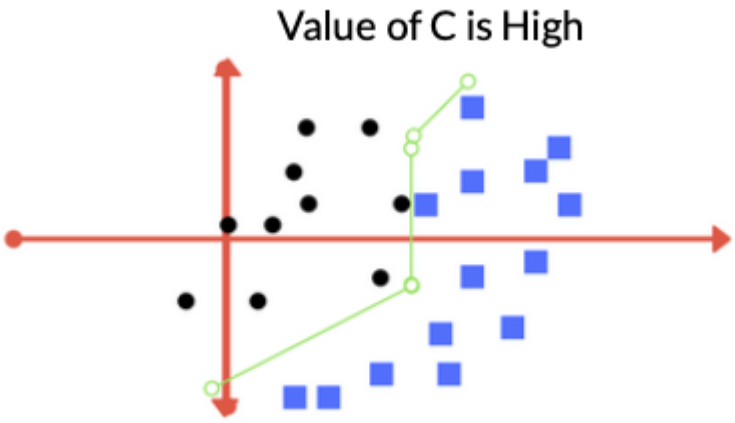


Fig. 21(b)

3. **Margin:** A good margin results in a better accuracy while classifying data.

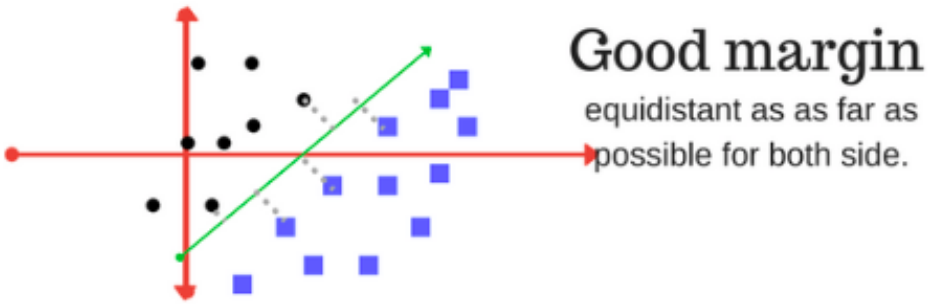
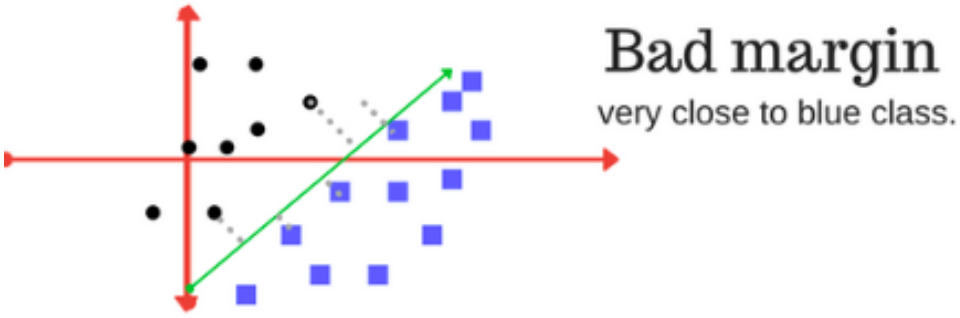


Fig. 22(a)

Fig. 22(b)



4. **Kernel:** We have used the following kernels:
    a. *Linear Kernel*: Used for linear classification.
    b. *Polynomial Kernel*: Here, the similarity function is a polynomial function.
    c. *RBF Kernel*: This kernel is used for nonlinear classification with Gaussian Function as the similarity function.

After understanding all the tuning parameters, we were pretty much set to implement SVM. We have 2 different implementations that each use a different dataset. The first implementation was using IRIS Dataset to understand how SVM is implemented. Once we understood how, we implemented SVM on our aforementioned dataset, i.e., *SVM using Image Pixel Values as Features*. Note here that when we implemented SVM, out dataset looked like the following:

The dataset contains RGB images of 5 different classes. Each image is stored in format '[class_name][serial_no].jpg'

1. 67 images containing dalmatian dog at different ages, animated non animated, with and without background.
2. 53 dollar bill images are cropped to fit the bill.
3. 53 images of pizza slices cropped from advertising posters, with and without background.

4. 65 soccer ball images.
5. 85 sunflower images taken in natural setting, some with the entire plant.

Dataset contains a total of 323 pictures. And classes are labelled as [0-4].

Now one might ask why we chose SVM over simple clustering. It's because dimensionality reduction using principal component analysis was performed on our feature vector so that it can be visualized graphically. This helped us to understand there are too many overlaps among the classes and hence cannot be clustered. The following PCA scatter plot illustrates the same.

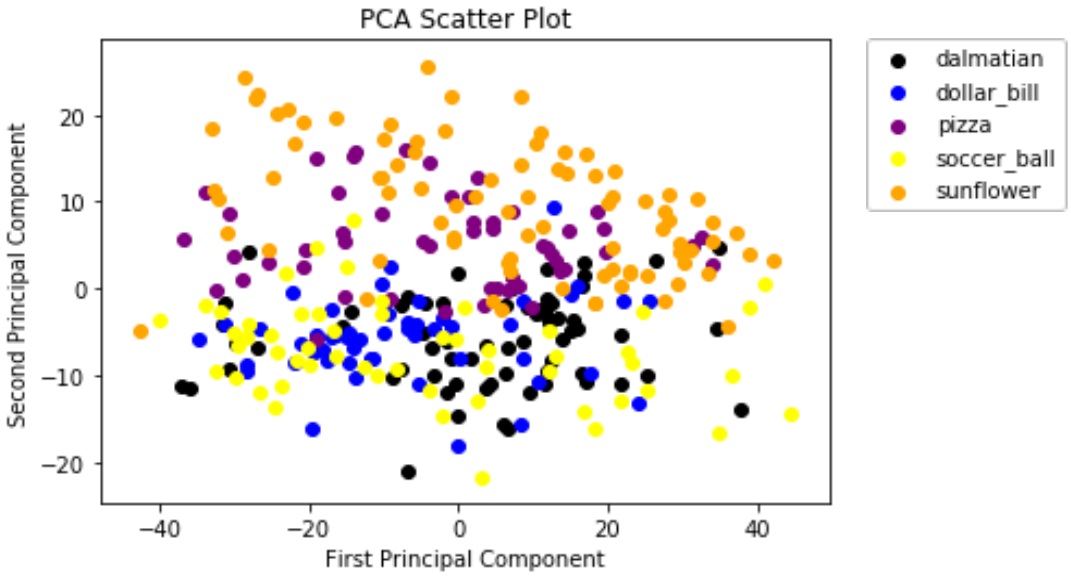


Fig 23. Graph Illustrating Disadvantages of Clustering

During implementation, we used two kernels. One was linear kernel as the data was not linearly separable but the accuracy in that case was very low, so we had to shift to RBF kernel which, unsurprisingly resulted in a better F1 score.

After conducting experiments on C values {1,10,100,1000} and gamma values {0.001, 0.0001}, we found out that C = 1 and gamma = 0.001 yielded the best results. Following are the results of the implementation:

Classes	Precision	Recall	F1 Score
Dalmatian	0.78	0.82	0.80
Dollar bill	0.88	0.93	0.90
Pizza	1.00	0.88	0.93
Soccer Ball	0.88	0.70	0.78
Sunflower	0.83	1.00	0.91
Micro-Average	0.86	0.86	0.86

Table 2. SVM Results (Accuracy Assessment)

CNN (NN):

After having successfully implemented SVM, we moved on to Neural Networks, in particular Convolutional Neural Networks (CNN).

The CNN Architecture we used is as follows:

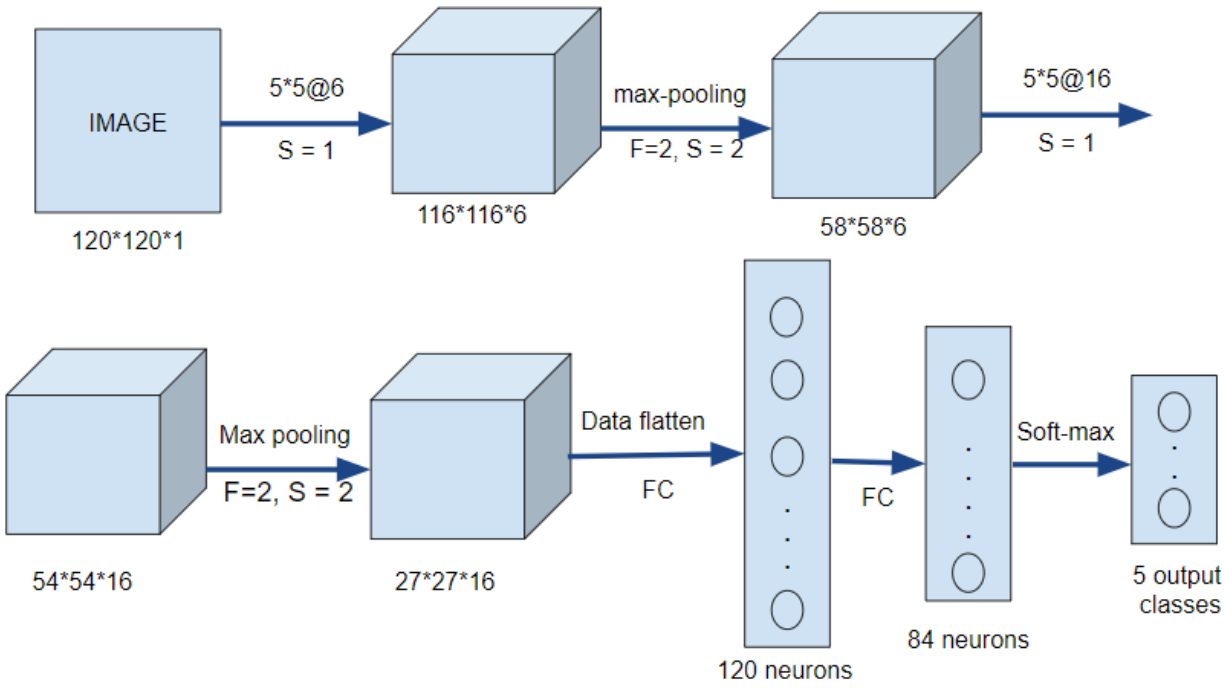


Fig 24. CNN Architecture

We have used CNN architecture similar to the architecture of **LeNet 5** (Fig. 24). In this first an image of shape 120x120x1 is convolved with a filter to produce 116x116x6 output. On this max-pooling with a stride and size of 2 operation is applied to reduce the output to 58x58x6 to which filter and max-pooling of stride and size 2 is applied to result to further reduce the output to 27x27x16. This result is then flattened until we obtain a

fully connected 84 neurons length layer on which soft-max is applied so that we get the result in terms of the 5 classes of images we have.

Max Pooling is done to provide the Convolutional Neural Network with the "spatial variance" capability. This in-turn serves to minimize the images as well as the number of parameters which, in turn, prevents an issue of "overfitting" from coming up.

Consider the following example (Fig. 25) where max pooling of size and stride 2 is done on a matrix of 4x4 to result into a matrix of 2x2. The maximum value from each frame or window with predefined size is taken in order to eliminate distortions.

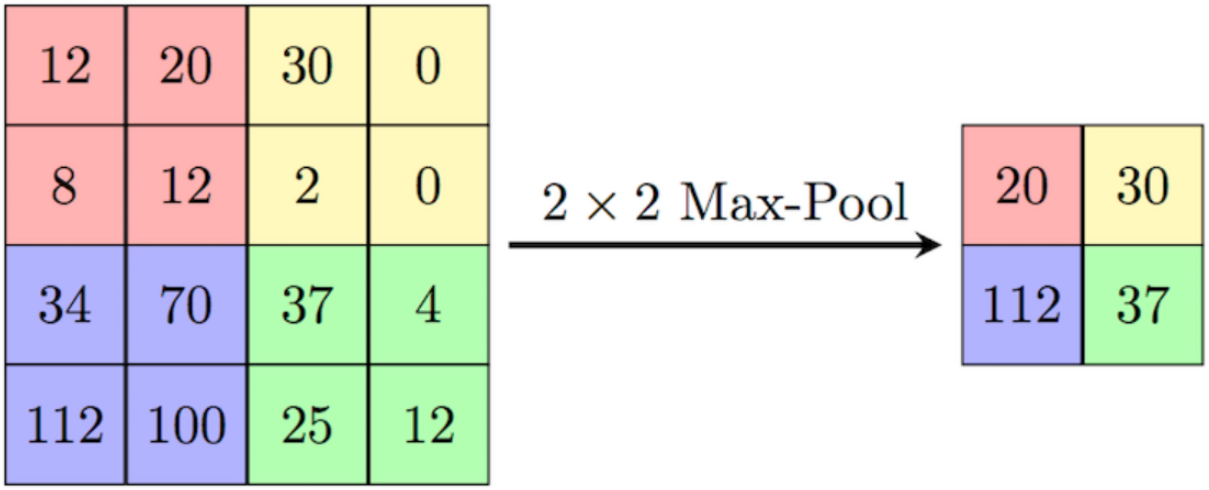


Fig 25. Max pooling of stride and size 2

Now let's move on to the activation function. While there are many activation functions, we used the Softmax Activation Function. Softmax is generally applied on the final layer in a multi classification problem. It helps in easily assigning probabilities to each class and predicting the outcome.

Consider an example where we have to predict the class of the image provided and the data points obtained in the last layer are 5, 2, -1, 3. To do so we first calculate the

exponential of each of the data points and divide them by the summation of all exponential values of the data points. This converts all the data points to probabilities (between 0 and 1) and the one which is closest to 1 is the class to which the provided image belongs.

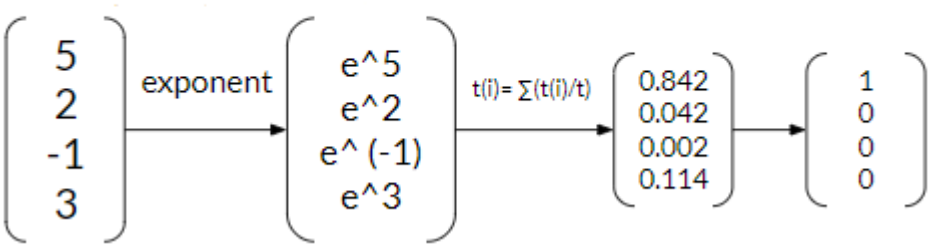


Fig 26. Softmax function

We have applied *Log Loss function* as the loss function to calculate the loss and minimize it for each iteration.

$$-\sum_{c=1}^{M} y_{o,c}\ log(p_{o,c})$$

Fig 27. Log Loss function

Fig. 27 shows the formula of Log Loss function where 'y' is either 0 or 1, 'p' is the probability assigned to that class and 'M' is the number of classes present.
For example, if in our case a certain image is categorized as Dalmatian then M will be 5, $y_1 = 1$ and the rest will be zero.

After dealing with both the activation function and the log loss function, we move on to the Stochastic Gradient Descent. To first understand **Stochastic Gradient Descent,** we have to look at **Gradient Descent**.

Gradient Descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).

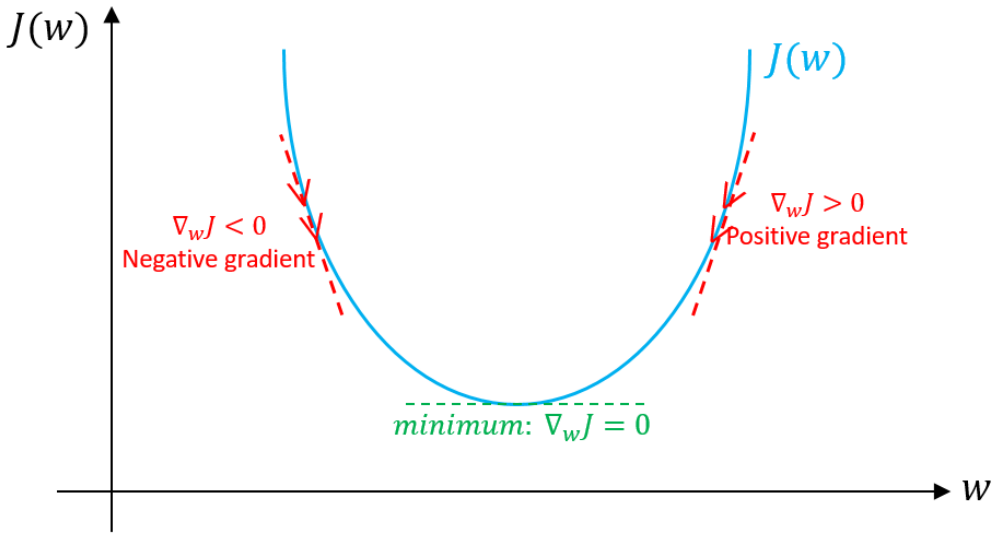


Fig 28. Gradient Descent function

Any point on the graph (Fig. 28) denotes the current value of the cost function. Our goal is to minimize this cost. In order to do so we keep on substituting new values of the coefficients such that the new cost is less than the previous one. We stop when we reach to bottom most point on the graph where the cost function is minimum. The values of coefficients at that point are then consider instead of the original ones.

However, if this process is repeated for each data point, we end up taking 3x the computational effort than it takes to compute loss. Also, computing the loss function takes into account every single data point. Since this process is iterative, we will go through the data point hundreds or thousands of times.

So instead, we pick a small number of random samples each time. This might seem counterintuitive at first, but it actually ends up giving the right results, although requiring us to do it many times but saving up on actual data point considerations. This method is nothing but **Stochastic Gradient Descent.**

Following are the results we have obtained after applying CNN.

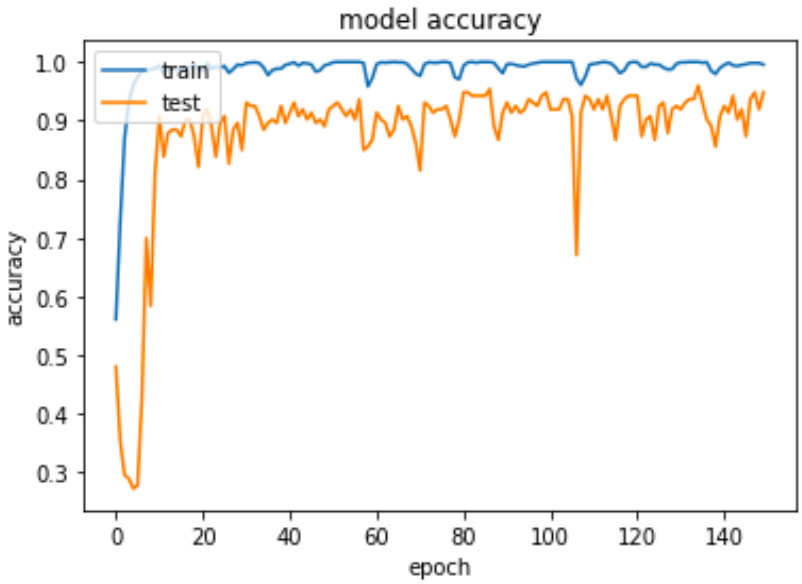


Fig. 29 (a): Graph of Variations in Accuracy with Epoch

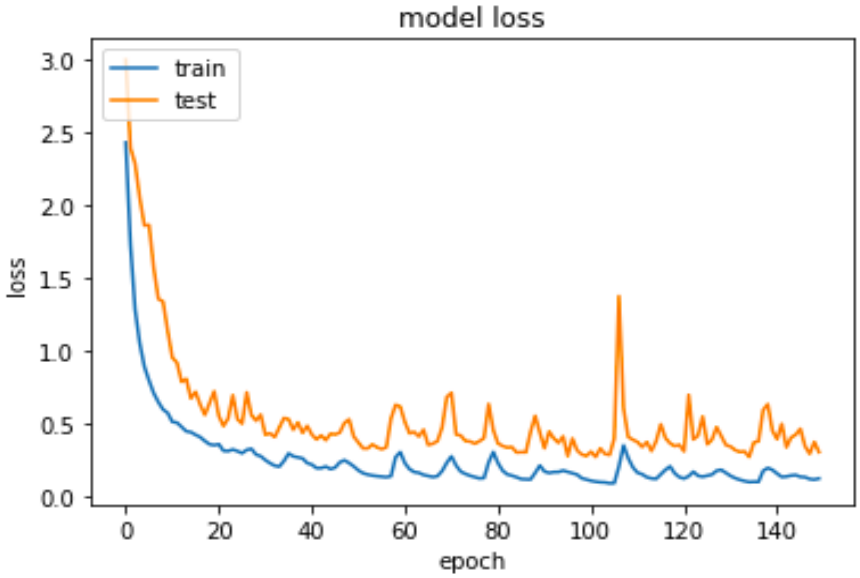


Fig. 29 (b): Graph of Variations in Loss with Epoch

From Fig. 29 (a) it can be inferred that the accuracy of the model has been greatly increased to 9 and from Fig. 29 (b) the losses has been greatly reduced to less than 1. We can notice that the graph of the test and train in both the graphs [Fig. 29(a) and Fig. 29(b)] are almost similar and very near to each other which in-turn suggests that the model has been trained well and is working as per our expectations.

…….	Dalmatian	Dollar bill	Pizza	Soccer Ball	Sunflower
Dalmatian	130	0	12	3	6
Dollar bill	0	125	3	1	3
Pizza	1	0	199	1	6
Soccer Ball	3	0	16	233	2
Sunflower	4	0	8	1	182

Table 3: Confusion Matrix

A row in the confusion matrix (Table 3) shows the number of images that were of some class and the predicted class. All the numbers in the main diagonal shows the number of correct predictions by the model.

……..	Dalmatian	Dollar bill	Pizza	Soccer Ball	Sunflower
Precision	0.94	1	0.83	0.97	0.91
Recall	0.86	0.94	0.96	0.91	0.93
F1-score	0.89	0.97	0.89	0.94	0.92

Table 4: CNN Result

The above table (Table 4) clearly shows that the accuracy of CNN model has been drastically increased from that of SVM model.

Conclusion and Future Work

In conclusion, we implemented SVM using a very small dataset to achieve an accuracy of **93%** and although SVM is a very strong technique, achieving such a high accuracy is still an anomaly. Eventually, we realized that our results boasted such a high accuracy due to the lack of a large enough dataset. So, using data augmentation, we more than tripled the size of our dataset and on performing SVM again, we achieved an accuracy of **82%**, a significant decrease. Unsatisfied with the results, we decided to move to other deep learning techniques. This quest led us to Neural Networks and in particular, CNN. On successfully implementing CNN, we achieved an accuracy of a staggering **93.57%** on the very same dataset. This stands as a testimony to the increased potential of deep learning techniques over the more traditional machine learning techniques.

Convolutional Neural Networks (CNNs) are similar to the more ordinary neural networks in that they are made up of hidden layers consisting of neurons with *learnable* parameters and conventionally, the Softmax function is the classifier used at the last layer of this network. However, our research noted that there have been many studies that say otherwise. These studies introduce the usage of Support Vector Machine (SVM) in an artificial neural network architecture. Now that we have the knowhow to implement both CNN and SVM, we think that we will be able to support or challenge these studies and if we do decide to take this project further, this is the direction we will be heading towards.

References

[1] D. Lu & Q. Weng, A survey of image classification methods and techniques for improving classification performance, International Journal of Remote Sensing, Vol. 28, No. 5, 10 March 2007, 823–870.

[2] Pooja Kamavisdar, Sonam Saluja, Sonu Agrawal A Survey on Image Classification Approaches and Techniques, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 1, January 2013.

[3] R. R Darlin Nisha, V.Gowri, A Survey on Image Classification Methods and Techniques for Improving Accuracy, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Volume 3, Issue 1, January 2014.

[4] S.V.S.Prasad , Dr. T. Satya Savithri & Dr. Iyyanki V. Murali Krishna, Techniques in Image Classification, Volume 15 Issue 6 Version 1.0 Year 2015, Global Journal of Researches in Engineering, Type: Double Blind Peer Reviewed International Research Journal.

[5] Maneela Jain Pushpendra Singh Tomar, Review of Image Classification Methods and Techniques, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181 Vol. 2 Issue 8, August - 2013.

[6] Nupur Thakur, Deepa Maheshwari, International Research Journal of Engineering and Technology (IRJET), Electronics and Telecommunication, Pune Institute of Computer Technology, Volume: 04 Issue: 11 | Nov -2017.

[7] Priya Pradip Naswale & P. E. Ajmire, Image Classification Techniques - A Survey, International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), Volume 5, Issue 2, March-April 2016

[8] Jipsa Kurian, V. Karunakaran, A Survey on Image Classification Methods, International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 1, Issue 4, October 2012.

[9] Chaitali Dhaware, Mrs. K. H. Wanjale Department of Computer Engineering, Vishwakarma Institute of Information Technology, Survey On Image Classification Methods In Image Processing, International Journal of Computer Science Trends and Technology (IJCS T) – Volume 4 Issue 3, May - Jun 2016.

[10] BangLiu, YanLiu, Kai Zhou, Image Classification for Dogs and Cats, Department of Electrical and Computer Engineering, Kaggle Dog Vs Cat Competition: http://www.kaggle.com/c/dogs-vs-cats., International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 3, Issue 5, December 2013.

[11] S. R. Suralkar, A. H. Karode, Priti W. Pawade, Texture Image Classification Using Support Vector Machine, Int. J. Comp. Tech. Appl, Vol 3 (1), 71-75 IJCTA | JAN-FEB 2012,ISSN:2229-6093.

[12] Le Hoang Thai, Tran Son Hai, Nguyen Thanh Thuy, Image Classification using Support Vector Machine and Artificial Neural Network, I.J. Information Technology and Computer Science, 2012, 5, 32-38 Published Online May 2012 in MEC, DOI: 10.5815/ijitcs.2012.05.05 .

[13] Foody, M. G., and Mathur, A. 2004b. Toward Intelligent Training of Supervised Image Classifications: Directing Training Data Acquisition for SVM Classification. Remote Sensing of Environment, 93, 107 - 117.

[14] L. Sunitha, M. BalRaju, J. Sasikiran, Department of computer Science and Engineering, Classification of Images using Support Vector Machine,International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064.

[15] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25 (pp. 1106-1114).

[16] Zeiler, M. D., & Fergus, R. (2013). Visualizing and Understanding Convolutional Neural Networks. arXiv preprint arXiv:1311.2901.

[17] Nadia Jmour, Sehla Zayen, Afef Abdelkrim, Convolutional neural networks for image classification,Published in 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET),14 June 2018.

[18] Muthukrishnan Ramprasath,M.Vijay Anand, Shanmugasundaram Hariharan, Image Classification using Convolutional Neural Networks, International Journal of Pure and Applied Mathematics Volume 119 No. 17 2018, 1307-1319 ISSN: 1314-3395.

[19] M. Manoj krishna, M. Neelima, M. Harshali,M. Venu Gopala Rao, Image classification using Deep learning, International Journal of Engineering & Technology, 7 (2.7) (2018) 614-617.

[20] Deepika Jaswal, Sowmya V., K.P.Soman, Image Classification Using Convolutional Neural Networks, June 2014, International Journal of Scientific and Engineering Research 5(6):1661-1668,DOI: 10.14299/ijser.2014.06.002.

[21] AbienFredM.Agarap,An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification, December 2017 License, CC BY-NC-SA 4.0.