12-21-2020

Image Spam Classification with Deep Neural Networks

Ajay Pal Singh
San Jose State University

Katerina Potika
San Jose State University, katerina.potika@sjsu.edu

Recommended Citation

Ajay Pal Singh and Katerina Potika. "Image Spam Classification with Deep Neural Networks" *Malware Analysis Using Artificial Intelligence and Deep Learning* (2020): 605-631. https://doi.org/10.1007/978-3-030-62582-5_24

Image Spam Classification with Deep Neural Networks

Ajay Pal Singh and Katerina Potika

Abstract Image classification is a fundamental problem of computer vision and pattern recognition. We focus on images that contain spam. Spam is unwanted bulk content, and image spam is unwanted content embedded inside the images. Image spam potentially creates a threat to the credibility of any email based communication system. While a lot of machine learning techniques are successful in detecting textual based spam, this is not the case for image spams, which can easily evade these textual-spam detection systems. In our work, we explore and evaluate four deep learning techniques that detect image spams. First, we train deep neural networks using various image features. We explore their robustness on an improved dataset, which was especially build in order to outsmart current image spam detection techniques. Finally, we design two convolution neural network architectures and provide experimental results for these, alongside the existing VGG19 transfer learning model, for detecting image spams. Our work offers a new tool for detecting image spams, usage of a bigger dataset, and is compared against recent related tools.

1 Introduction

Over the last decade, email and internet is flooded with spam content. A spam can be defined as unwanted content, distributed mostly via emails. Due to the effluence of spam emails over the Internet a lot of techniques have surfaced which classify the spam from the valid content. Reports from Symantec [18] indicated that 90.4% of the emails includes spam content. These spam emails can include phishing links,

Ajay Pal Singh
San Jose State University, One Washington Square, San Jose, CA 95192, e-mail: ajay.id.4.ms@gmail.com

Katerina Potika
San Jose State University, One Washington Square, San Jose, CA 95192, e-mail: katerina.potika@sjsu.edu

malware, advertisements, adult content, and other, which may impose a significant threat to the security of the user's privacy.

Spam initial was only in the form of texts. With the advent of machine learning, many classifiers were developed to filter such spam based on email content. Lai and Tsai [11] used four different machine learning techniques, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM) and Naïve Bayes, that used email messages to filter spam emails. These classifiers were able to classify text based spam with approximately 95% accuracy. Hence, over the years detecting content based spam emails became very easy. Google, Microsoft, Yahoo use techniques that perform very accurately to classify spam emails from the authentic emails.

However, over time, spammers came up with novel ideas to fool these content based classification techniques. Thus, image spam was developed, where unwanted textual information was delivered in the form of images. To detect these type of image text, optical character recognition (OCR) techniques [4] were developed which were able to extract the text from the images. It involves segmentation of the textual region within the images and using techniques to extract text from these regions. However, these text based classifiers were not always successful in detecting image spam. One reason was that segmentation of textual area within these images in itself is a difficult task [17]. Also, spammers started using obfuscation techniques, which made the OCR techniques less effective.

A more direct approach was used by Annadatha et al. [1] and Aneri Chavda et al. [2], where they consider properties of the image itself to classify spam images. They used image processing techniques in conjunction with various machine learning models. We use different deep learning techniques on various image properties as compared to these previous work [1] and [2]. We train neural networks and deep neural networks on these image properties, instead of using the machine learning techniques that were used previously in [2]. We then divulge deeper and experiment with other deep learning techniques, such as convolutional neural networks (CNN) based on raw images. Finally, we discuss the use of transfer learning and train a VGG19 model on our dataset. The main focus of this work is to quantify the robustness of these techniques on an improved spam dataset created by Aneri et. al [2].

The remainder of the report is organized as follows. In Section 2, we discussed the problem statement and the motivation behind it. Section 3 focuses on the related work done so far in this domain. Section 4 describes the essential background, topics and terminologies needed to understand this project. Section 5 discusses the various datasets used in this work, the steps involved in pre-processing these datasets and the architecture used to train the deep learning models. Section 6 presents the experimental results. Finally, Section 7 concludes and provides scope for future work.

2 Problem Statement and Motivation

This section defines the problem statement and scope for this work. It also focuses on the motivation and purpose of solving this problem.

We focus on binary image classification. Anything that contains the marketing, sexual or other unwanted content embedded within the images is called a SPAM image, whereas anything other than that is considered a HAM image. HAM is a keyword specified in the and used in previous papers [2], [1], so in order to maintain the consistency we will also use the same terminology.

The goal is to use specifically Neural Networks and Deep Neural Networks on the problem of spam image classification in order to obtain better results as obtained in the previous papers. There are two main parts towards that direction:

- Classification based on image features: extraction of 38 features from the images as described in [2] and then use the different Neural Networks and Deep Neural Network architectures with the motive to improve the results.
- Use of Raw Images: usage of deep learning techniques, such as Convolution Neural Networks, with different architectures and then with pre-built models based on transfer learning on the spam image classification problem.

Using different approaches as mentioned above the results are presented in the form of tables, graphs and other metrics to give a quantification of this work.

The internet is flooded with spam content, whether it is in the form of text or unwanted text in the images. Previous techniques are good to detect textual spam but the spammers are coming up with new ways to fool such techniques. We try to solve this hard problem of image spams. We discuss the results obtained to classify spam images by leveraging the power of neural networks and deep learning. Since the advent of deep learning, there is not much research done on this domain. By using our approach potentially administrators of email systems or other systems can minimize the spam content that is even embedded in images.

3 Background

Let us present the essential background and terminology that we need.

3.1 Spam Categories

In general spam detection techniques are partitioned into the following two categories:

1. Content based spam: spam in emails that are in textual form; classifiers in this case deal with the actual content of the email extracted from email headers, keywords,

body of email, etc. Wide variety of machine learning techniques are available for such spam classification [3].

2. Non - content based spam: spam that use advanced forms; such one is an image spam. For image spam classification we can look at the properties of an image, or with the advent of deep learning we can use images in their raw byte form. There are different generation of image spams ranging from first generation to third generation. Images in the first generation contain plain spam images, but in the second and third generation images are obfuscated using noise, by overlaying background of images to make them more resistant to OCR techniques. OCR techniques are capable of segmenting the part of the images that contain specific object for further purposes, for example extraction of text or object detection.

3.2 Classification Techniques

Here we mention the techniques that we use and are implemented for the experiments described in Section 6.

3.2.1 Neural Networks

Artificial neural networks (ANNs) are algorithms that are modeled after the neuronal structure of cerebral cortex of the human brain, but on much smaller scales. A neural network (NN) structure is divided into different layers: the input layer, one or many hidden layers, and the output layer. Each layer comprises of nodes or neurons. A basic unit of computation in a neural network is a neuron, which receives an input from previous layer nodes along with their specific weights, and performs a function on them. This function is also called the activation function. These neurons are activated by using different activation functions. Some examples of activation functions: the sigmoid, the tanh and the Relu function. A bias is usually added with each layer to provide regularization, and move the function graph by some constant from the center. The goal of the ANN's is to decrease the loss function, which is derived from the dataset.

As compared to support vector machines (SVM), which use only one function, neural networks provide non-linearity due to the structure of its layers. There are different types of neural networks, for example feedforward NN, single layer perceptron, multi-layer perceptron, and so on. The output layer in case of a classification problem usually consists of a sigmoidal activation function to provide probabilities for different classes, or labels for the dataset.

3.2.2 Deep Neural Networks

They are differentiated from basic NN by their depth; that is, the number of node layers through which data passes in a multi-step process of pattern recognition. In this each layer of a network is supposed to learn specific features as received from the previous layer. The further the layer is, nodes are able to understand more complex features, since they aggregate and recombine those features from previous layers. Deep-learning networks perform automatic feature extraction without human intervention, unlike most traditional machine-learning algorithms.

3.2.3 Convolution Neural Networks

The idea behind Convolution Neural Networks (CNN) was derived from NNs with neurons that learn from weights and biases. Also, each layer calculates a non-linear function using teh dot product and some activation functions. CNN's work on the images itself in the input layer. The normal NN's does not scale very well on the raw images, because they are not able to learn enough features from them. The architecture of CNN's introduce different types of layers, which each of which learn specific features from the previous layer. Thus, the general idea is that the starting layers are able to learn more generic features, such as the curves and edges from the images, then as the architecture grows these layers becomes more specific, for example detecting the ears of an animal. Unlike a NN a CNN have neurons arranged in three dimensions namely width, height and depth. The depth here refers to the channels in an image, for a color image the depth is 3, whereas for a grayscale image it is 1. The CNN architecture is built from different type of layers, which are repeated as necessary to build the deep CNN. These layers are:

1. Convolutional Layer: Tries to learn the features from the images by preserving the spatial relationship among pixels. It uses the concept of filters. The images are divided into small squares on which these filters are projected. These filters contain different values of pixels, for example a filter can be used to find edges of text in a spam image. So, if there exist edges in the speculated square, those pixels are activated. The pixels on the filter are multiplied on the input image area under consideration and a sum is performed over those activated pixels within the filter to check the intensity of the filter. These filters work in sync with the depth of the images, so if the image is RGB then there are 3 filters used for each depth of given sizes. There are three parameters that control the size of the Convolution layers: stride, depth and padding. The depth is mostly determined by the depth of the raw images or based on the previous layer input. The stride defines the number of pixels the filters are slided from left to right. Sometimes the input layer features are padded with zeros to maintain proportion with the size of the filter and to control the size of the output layer. After sliding the filter over all locations of the input array we get an activation map or feature map.
2. RELU (Rectified Linear Units) Layer: To provide non-linearity after each Convolution layer it is suggested to apply a RELU layer. A RELU layer works far better

in terms of performance as compared to the sigmoid or tanh function without compromising the accuracy. This layer also overcomes the problem of vanishing gradient. In vanishing gradient the lower layers train much slowly, because the gradient due to back-propagation decreases exponentially through the layers. The RELU function is given as

$$F(x) = \max(0,x)$$

This function changes all negative values to 0 and increases the non-linearity of the model without affecting the output of the Convolution layer.

3. Pooling (or down-sampling) Layer. It uses a filter of a given size, moves across the input from previous layer and applies a given function. For example, in a max pooling layer, a max of all the filter values is given as output. There are other types of pooling layers as well such as average pooling and L_2-norm pooling. This layer serves two purposes: it decreases the amount of computation by decreasing the amount of parameters of weights, and it overcomes overfitting in the model.
4. Dropout Layer: This layer helps in overcoming the problem of overfitting. Overfitting means the weights and parameters are so tuned to the training examples after training the network, that they perform very bad on the new examples. So this layer drops out a random set of activation in a given layer by setting there values to 0 [16].
5. Fully Connected Layer: it takes as input from any of the Convolution layer, Pooling or RELU layer and outputs a N dimensional vector. This N depends on the number of classes you want to classify. In case of the image spam classification problem the value of $N = 2$, i.e., whether the image is a SPAM or a HAM.

A classic CNN architecture is composed of the above layers repeated in some fashion as necessary. A simple example of such architecture is given in Figure 1.

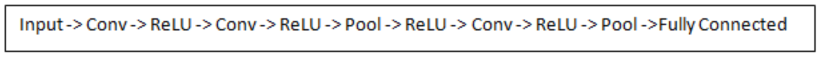

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool ->Fully Connected

Fig. 1 A simple CNN architecture composed of different layers

3.2.4 Transfer Learning

Data is an essential part of deep learning community. As you train your networks on large amount of data the network becomes more redundant and efficient in generalizing the results to new datasets. Thus in case you have a small amount of dataset to actually work on, transfer learning overcomes this caveat. It is a process of using pre-trained models, which are trained on millions or billions of samples of generalized datasets and then fine-tune these models on your own datasets. Rather than training the whole network we use a pre-trained model weights and freeze them and focus on training only specific lower level of layers which are more specific to our dataset. If your dataset is different from the pre-trained model dataset then in that case training more layers of the model is preferred. We focus on using two such pre-trained models VGG16 and VGG19.

A technique called data augmentation is widely used in overcome the problem of having less samples of dataset. It performs different image transformation on the images to produce new images and hence augment the dataset. Some transformations may include scaling the image by some ratio, rotating them, skewing, flipping and cropping the images.

3.3 Quality Metrics

The following terms will be used to quantify the results:

- True Positive (TP): an image is a SPAM image and classifier marks it is as a SPAM
- True Negative (TN): an image is a HAM image and classifier marks it is as a HAM
- False Positive (FP): an image is a HAM image and classifier marks it is as a SPAM
- False Negative (FN): an image is a SPAM image and classifier marks it is as a HAM
- Confusion Matrix: This is a matrix between TP, TN, FP and FN. A Figure 2 taken from [2] is shown below.

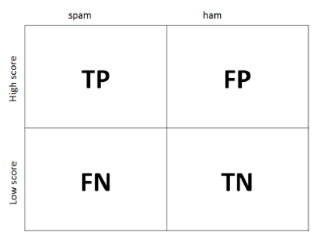


Fig. 2 A Confusion Matrix [2]

- Accuracy: It is a metric used to determine how well a classifier works. It is defined as:
$$Accuracy = \frac{TP + TN}{P + N} \tag{1}$$
where P (Positive) = $TP + FN$ and N (Negative) = $TN + FP$.
- ROC and AUC: Receiver Operating characteristics (ROC) and Area under the Curve (AUC) can be obtained from a trained classifier. A ROC curve is plotted against TPR (True positive rate) and FPR (False Positive rate) for varied threshold

values. An area under this ROC curve is known as AUC value. TPR and FPR are determined as:

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{2}$$

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN} \tag{3}$$

An AUC close to 1 is considered as a good classifier.

- K-fold cross validation: In this technique the classifier is trained K times. The dataset is divided into K subsets. Each time a classifier is trained, one of the K subsets is used as the validation or test and the remaining $K - 1$ subsets are used as the training set. The accuracy over all these K classifiers is averaged to provide the average accuracy. Cross validation techniques are generally used to overcome over-fitting within the dataset.
- Stratified K-fold cross validation: It is a slight variation in the K-fold cross validation technique. In this each fold is created in such a fashion that each subset contains approximately the same percentage of each target class. This is used in cases where the dataset classes are skewed i.e one class predominates the other.

4 Related Work

Image spam detection can be done using various techniques. One such approach is to use content based detection, i.e., to segment the content using OCR techniques and then classify it. Other approaches include to detect image spams based on the properties or features extracted from these images. Different machine learning algorithms are also used in conjunction with image processing to generate strong classifiers. Additionally, deep learning techniques, such as CNN, can also be used on the raw images to detect image spams.

A content based image spam detection technique is discussed in [1], which uses OCR techniques on spam images. These techniques extract the text from the segmented image, and perform textual analysis on the extracted text to determine whether the image is a SPAM or a HAM.

Another paper by Gevaryahu, Elias-Bachrach et al. [4] use the image meta data properties, such as file size and other properties to detect SPAM images. The work presented in [5] use probabilistic boosting trees for the classification by working on the color and gradient histogram features and achieve an accuracy of 94%.

Sanjushree, Suhasini et al. [10] use SVMs and particle swarm optimizations (PSO) on ten metadata features and three textual features for SPAM image classification. Using the combination of these techniques they were able to achieve a 90% accuracy on the Dredze dataset (discussed below) using 380 test and 300 training images. The authors of [5] use cluster based filtering techniques with client and server based models and claim to achieve a 99% accuracy.

A fuzzy inference system was used to analyze multiple features in [8], claiming to have achieved an accuracy of 85%. Annadatha et al. [1] used a linear SVM classifier on 21 image properties and each property was associate with a weight for the classification. The author performed feature reduction and selection based on these weights. Two datasets [5] and Dredze [4] is used by them and they achieved an accuracy of 97% and 99% respectively. Moreover, they also developed a new challenging dataset for their classifier.

Aneri et al. [2] used 38 features extracted for the Dredze and the Image Spam hunter (ISH) datasets (discussed below). In their approach they use SVM kernels and achieved an accuracy of 97% with linear, 96% with Radial Basis Function (RBF), and 95% with a polynomial kernel on the ISH dataset. Respectively, the results are 98% using linear, 98% using RBF and 95% using a polynomial kernel on the Dredze dataset. Additionally, feature reduction techniques are used, like univariate feature selection (UFS) and recursive feature elimination (RFE), to reduce the initial 38 features and provide better results. Expectation Maximization (EM) clustering techniques are some of their other approaches, but these did not performed very well and achieved an accuracy of 87% on the ISH and 70% on the Dredze dataset. One of the main contribution of this work is the creation of a new challenging dataset, that we also use in this work. The accuracy was not very good for it being 52%. The number of samples that we use are relatively large from these datasets. A big part of our work also discusses various data processing techniques used to extract images from the spam archive, which was an unprocessed dataset as provided by Dredze. Hence, the amount of data the experiments and results are based on are large as compared to all previous papers.

In a recent work, Sharma et al. [14] use CNNs along with other machine learning techniques to solve the SPAM image classification problem. They consider similar to our real-world image spam and challenging datasets. The best results they get are for their CNNs with an accuracy of 99.02% for the ISH dataset, 83.13% for the improved dataset of [2], and 71.83% for the challenging dataset of [1] based on the Dredze dataset.

Soranamageswari et al [15] use back-propagation neural network based on only color features and was able to achieve a 92.82% accuracy. An interesting aspect used was that of splitting the image into different blocks and using them as features, but this approach only achieves a 89.32% accuracy.

Hong-Gang Zhang and Er-Xin Shang [13] use CNNs to classify into 7 categories of unwanted content embedded in images. The worked on a dataset containing around $52K$ images. The 7 categories targeted by them include: commodities, spam images, political content images, adult content, recipe images, scenes, and everything else. They used five convolution alongside max pooling layers. The output of these layers is given to the fully connected layer and a SVM classifier is used on this N sized feature vector. They resized all the images to a $256x256$ size and were able to achieve an average accuracy of 75.1%. Following this approach we use a custom CNN architecture, but only on a binary classification, namely SPAM and HAM.

5 Framework

5.1 Datasets

We experiment with the three different datasets that are used by Aneri et al. [2]. However, the number of images used is much larger compared to the images used in that approach. We focus on making use of different formats of images such as gif, jpeg, jpg, png, tif, and bmp. The gif images were processed and the first frame were extracted from the gif images and converted to png files.

5.1.1 Dredze Dataset

Dredze et al. [4] created a SPAM image archive dataset as well as a personalized SPAM image archive. The personalized SPAM image archive contained a lot of unprocessed files in different formats such as gif, txt, jpg, etc. We preprocessed this archive as well to augment our dataset. Then, the experiments were performed on the combination of the Dredze SPAM archive and the Dredze personalized achive. Earlier papers [1] and [2] focused only on a subset of the personalized SPAM images archive. After the preprocessing step, the personalized SPAM images were 3165 with 1760 HAM images, and the SPAM images obtained were 10937. Totally, $14,103$ SPAM images and $12,565$ HAM images.

5.1.2 Image Spam Hunter (ISH)

The image spam hunter dataset contained both HAM and SPAM images [5]. We extracted and processed 922 SPAM and 810 HAM images.

5.1.3 Improved Dataset

This dataset was created by Aneri et al. [2]. This dataset was created by performing transformation on the HAM images to make them SPAM. The HAM images were resized to the size of SPAM images to align there metadata features. Noise was introduced in the SPAM images to make there edge detection difficult, since SPAM images generally have less noise as compared to HAM images. These noise induced SPAM images were overlayed on top of the HAM images to generate the improved dataset. We experimented with the additional $1,030$ improved SPAM images.

5.1.4 Combined dataset

In general CNN requires large amount of datasets to converge and perform better. So instead of experimenting with individual datasets mentioned above we combined together all these datasets to augment the number of SPAM image samples. In order to account for the HAM images we downloaded images belonging to different categories, that are not SPAM, to make our dataset balanced.

5.2 Data Pre-Processing

The Dredze dataset archive contained a lot of unwanted files and corrupted images from which features could not be extracted. There were a lot of images in different formats and almost $60 - 70\%$ of the images were in gif formats. These gif images were processed to extract the first frame and then saved in a png format which helped in augmenting the dataset. The steps in order to achieve this objective are descripted next:

1. All unwanted formats, such as .txt, were removed.
2. All .gif images were converted to .png files. All frames of the .gif images were extracted and the first frame was saved as a SPAM image. The rest of the frames did not contain a SPAM image and were discarded.
3. All corrupted files were removed.

In order to achieve the above objectives, different proof of concepts were tried. We created bash scripts to perform all the above steps and keep track of the results after each step to get a clean augmented SPAM images archive.

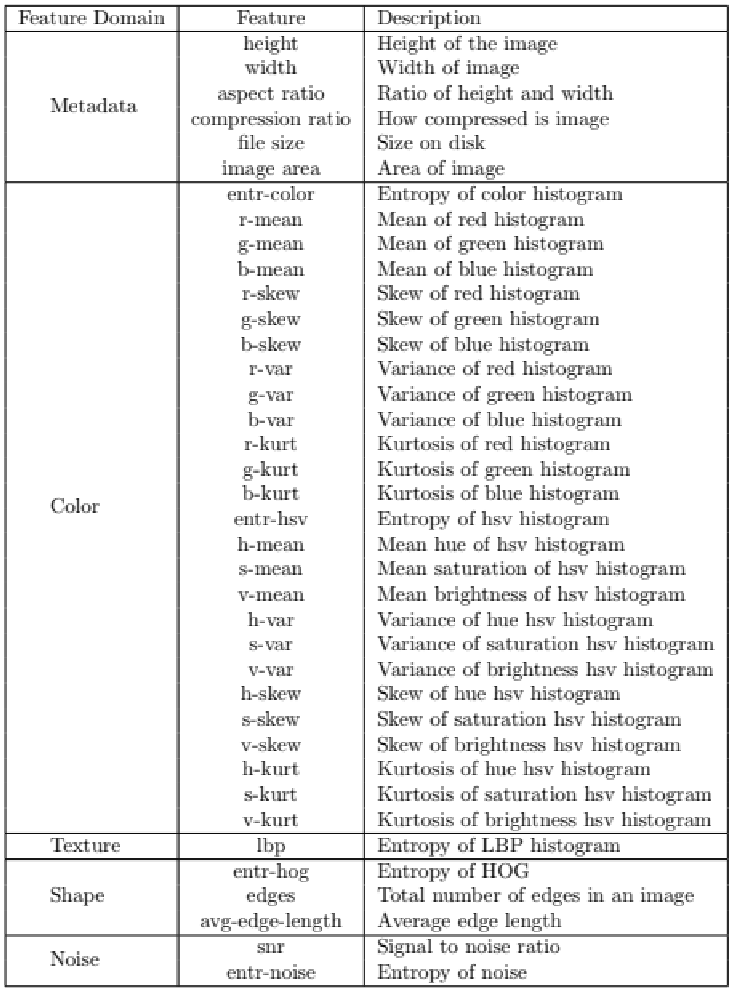
5.3 Image Features

The first part of the experiments used NNs and DNNs on features extracted from the images from the different datasets. We use the 38 features as mentioned in [2]. The features are classified in five big categories: metadata, color, texture, shape, and noise features. Figure 3 below describes the different features belonging to each category. The different categories of features is discussed below.

5.3.1 Metadata Properties

These properties contain the image height, width, aspect ratio, bit depth and compression ration of the image files. Compression of an image is defined as

$$\text{Compression Ratio} = \frac{\text{height} * \text{width} * \text{channels}}{\text{file size}} \tag{4}$$



Feature Domain	Feature	Description
Metadata	height	Height of the image
	width	Width of image
	aspect ratio	Ratio of height and width
	compression ratio	How compressed is image
	file size	Size on disk
	image area	Area of image
Color	entr-color	Entropy of color histogram
	r-mean	Mean of red histogram
	g-mean	Mean of green histogram
	b-mean	Mean of blue histogram
	r-skew	Skew of red histogram
	g-skew	Skew of green histogram
	b-skew	Skew of blue histogram
	r-var	Variance of red histogram
	g-var	Variance of green histogram
	b-var	Variance of blue histogram
	r-kurt	Kurtosis of red histogram
	g-kurt	Kurtosis of green histogram
	b-kurt	Kurtosis of blue histogram
	entr-hsv	Entropy of hsv histogram
	h-mean	Mean hue of hsv histogram
	s-mean	Mean saturation of hsv histogram
	v-mean	Mean brightness of hsv histogram
	h-var	Variance of hue hsv histogram
	s-var	Variance of saturation hsv histogram
	v-var	Variance of brightness hsv histogram
	h-skew	Skew of hue hsv histogram
	s-skew	Skew of saturation hsv histogram
	v-skew	Skew of brightness hsv histogram
	h-kurt	Kurtosis of hue hsv histogram
	s-kurt	Kurtosis of saturation hsv histogram
	v-kurt	Kurtosis of brightness hsv histogram
Texture	lbp	Entropy of LBP histogram
Shape	entr-hog	Entropy of HOG
	edges	Total number of edges in an image
	avg-edge-length	Average edge length
Noise	snr	Signal to noise ratio
	entr-noise	Entropy of noise

Fig. 3 Different Image Features [2]

5.3.2 Color Properties

These properties include mean, skew, variance, and entropy values of different properties of an image such as RGB colors, kurtosis, hue, brightness, and saturation. Mean can be a basic color feature that represents the average pixel value of the image. That is, it is useful for determining an image background. A SPAM compared to a HAM image has different histogram properties for these features as depicted in the examples of Figure 4. These histograms show the reasoning behind selecting these properties of color for the classification task. Similarly, in the examples of Figure 5 one can see the histogram for HSV's values of a SPAM and a HAM image. In images, a glossier surface has more positive skew values as compared to a lighter and matte surface. Hence, we can use skewness in making judgments about image surfaces. Kurtosis values are interpreted in combination with noise and resolution measurement. High kurtosis values goes hand in hand with low noise and low resolution. SPAM images usually have high kurtosis values.

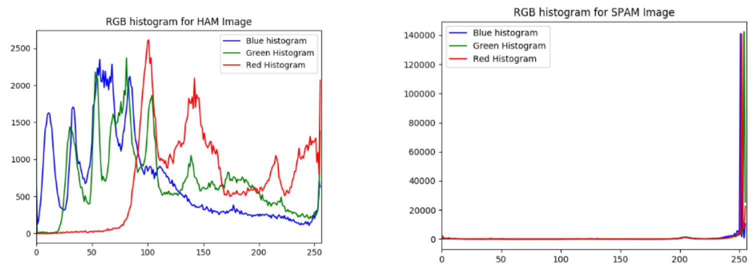


Fig. 4 SPAM vs HAM RGB's histogram

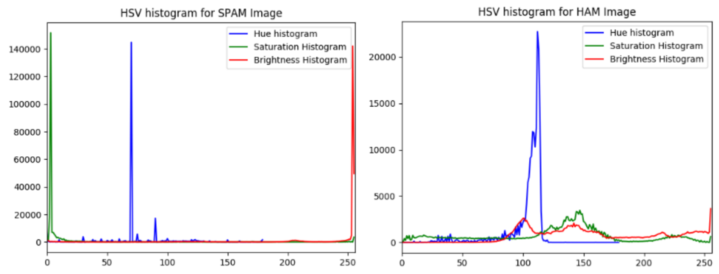


Fig. 5 SPAM vs HAM HSV's histogram

5.3.3 Texture Properties

The Local Binary Pattern (LBP) is used to determine similarity and information of adjacent pixels. The LBP would appear to be a strong feature for detecting an SPAM image that is simply text set on a white background. In the case of SPAM images these histograms will have high intensity for specific values rather than being scattered.

5.3.4 Shape Properties

Histogram of Oriented Gradients (HOG) determines how intensity gradient changes in an image. HOG descriptors are mainly used to describe the structural shape and appearance of an object in an image, making them excellent descriptors for object classification. Edges are one of the most important features to detect SPAM images. They serve as to highlight the boundaries in an image. Canny edge filters are used to find the edges. Figure 8 below shows the contrast in canny edges for SPAM and HAM images. Figure 6 and Figure 7 shows the hog features for a HAM and a SPAM image.

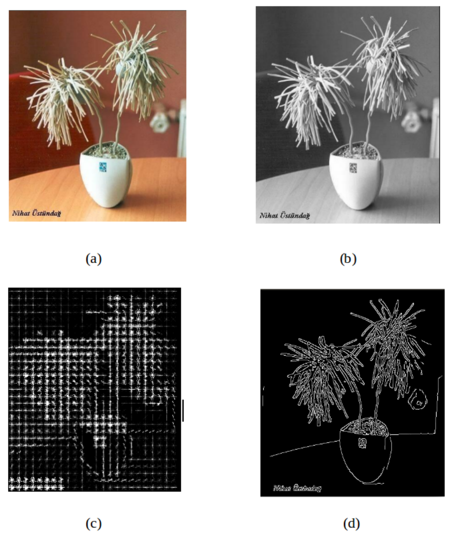


Fig. 6 a) HAM original Image b) HAM Grayscale Image c) HAM HOG d) HAM Canny edges

5.3.5 Noise Properties

These features include signal to noise ratio (SNR) and entropy of noise. SPAM images tend to have lesser noise as compared to HAM images. SNR is defined as the ratio of mean to standard deviation of an image.

5.4 Techniques Used

Let us briefly describe the various techniques and architectures that we are going to use.

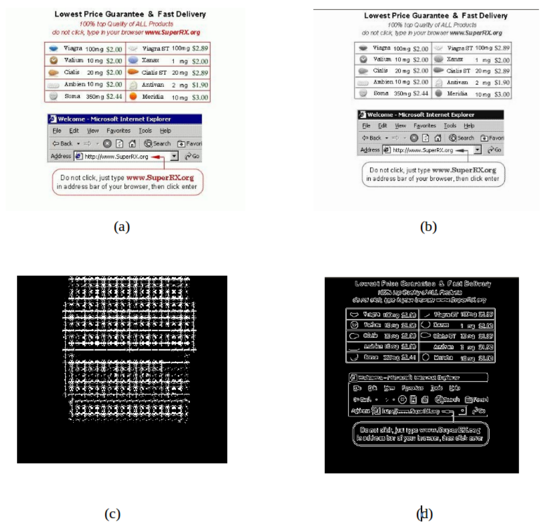


Fig. 7 a) SPAM original Image b) SPAM Grayscale c) SPAM HOG d) SPAM Canny edges

5.4.1 Neural Networks

A back-propagation neural network with 1 hidden layer with 20 neurons was used. The input layer consisted of the 38 features. The hidden layer used the RELU activation function and the output layer consists of the sigmoid activation function with one neuron. A K-fold stratified cross validation, with $K = 10$, was used with this. An architecture of the model is shown in Figure 8.

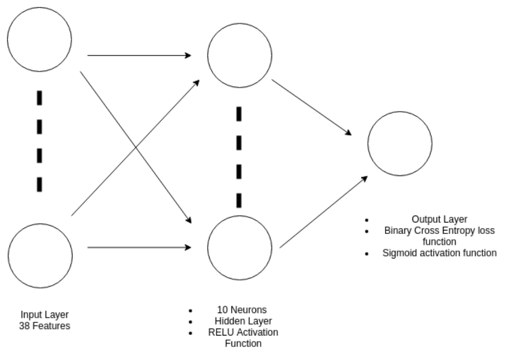


Fig. 8 Neural Network Architecture

5.5 Deep Neural Networks

The previous neural network was extended to introduce another hidden layer with 10 neurons and with the RELU activation function. Binary crossentropy was used as the loss function and again with K-fold stratified cross validation. We make use of two CNN architectures. We name them as CNN1 and CNN2. CNN1 was trained for 30 iterations, whereas CNN2 was run with 25 iterations. Both of them was trained with a batch size of 64 images. The training set contained 19924 images, whereas the validation set contained 2681 images.

5.5.1 CNN1 architecture

The CNN1 architecture as defined below was used as the third model to draw results. The images were first rescaled to 128x128x3 and then fed to the network.

1. 96 filters of size 3x3x3 were used to the input layer with a stride of 1 followed by the RELU function.
2. On the output of 96x126x126 from the previous layer a max-pool layer is used taking the maximum value from a 3x3 area with stride of 2x2.
3. On the input of 96x62x62 from previous layer another convolution layer with 128 filters, with 3x3 filter size and stride of 1, and no padding is used followed by a RELU activation function.
4. On 128x60x60 input from previous layer another maxpooling layer with a 3x3 area and stride of 2 is used on the input of previous layers to produce an output of size 128x29x29.
5. The input of the previous layer is flattened and given to a fully connected layer. The N vector obtained from the input layer is of size 107648. On this N vector a dense layer of size 256 is used with RELU as activation function and a dropout layer with probability of 0.1. Another dense layer of size 1, which acts as the output layer is added to the end of this with sigmoid activation function.

5.5.2 CNN2 architecture

The CNN2 architecture is defined below, and was used as the fourth model to draw results. The images were again first rescaled to 128x128x3 and then fed to the network.

1. 128 filters of size 3x6x6 were used to the input layer with a stride of 2 followed by the rectilinear linear operator (RELU) function.
2. On the output of 128x62x62 from previous layer a max-pool layer is used taking the maximum value from a 4x4 area with stride of 1.
3. On the input of 128x59x59 from previous layer another convolution layer with 128 filters with 4x4 filter size and stride of 1 and no padding is used followed by a RELU activation function.

4. On 128x56x56 input from previous layer another maxpooling layer with a 3x3 area and stride of 1 is used on the input of previous layers to produce an output of size 128x54x54.
5. On the previous layer input another convolution layer with 256 filters with 3x3 filter size and stride of 2 is used followed by a RELU activation function.
6. On the 256x26x2 input from the previous layer another maxpooling layer with a 5x5 area and stride of 2 is used on the input of previous layers to produce an output of size 256x12x12.
7. The input of the previous layer is flattened and given to a fully connected layer. The *N* vector obtained from the input layer is of size 36864. On this *N* vector a dense layer of size 1024 is used with RELU as activation function and a dropout layer with probability of 0.2. Another dense layer of size 128 is added after that with a dropout layer with probability of 0.1 and RELU activation function. The final layer is of 1 neuron, which acts as the output layer with sigmoid activation function.

5.6 Transfer Learning

There are pre-trained models that are open sourced. These models are trained on billions of images such as the ImageNet database [9]. Transfer learning is used to decrease the computation time to train your own network and to make you make use of these pre-trained networks. We can freeze some layers based on our requirements and only train a subset of those layers on our own dataset. There are two such pre-trained models available as open source, VGG16 and VGG19 [7, 12]. However, here we only discuss the VGG19 model and the assumption is that VGG16 would have performed similar to VGG19 with some minor difference in accuracy.

5.6.1 VGG19

In the VGG19 architecture we added 3 fully connected layers at the bottom with 1024, 512 and 1 layer respectively, and added a dropout layer with probability of 0.3 with 1024 neurons layer. We freeze all the layers of the network and just trained this fully connected layer added to the end in 50 iterations.

6 Experimental Results

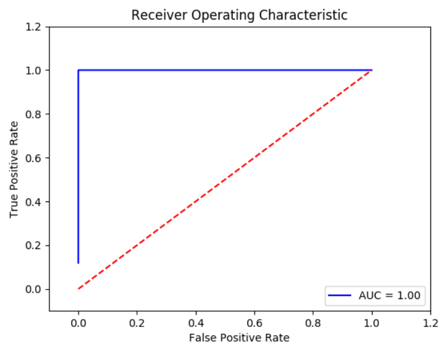
We first discuss the NNs results, then we go deeper and show our results for DNNs. After that, we will show an alternative approach, of using raw images from our dataset, and explain the results obtained for the CNN1 and the CNN2 architecture. Finally, we conclude with the results obtained from the VGG19 model, which uses the transfer learning approach. All the results were obtained by using the datasets discussed in Section 5. Specifically, NNs and DNNs were trained and tested on the Dredze, the ISH dataset and the improved dataset. Whereas CNN1, CNN2 and VGG19 was run on the combined dataset.

6.1 Neural Network results

We created a neural network with the architecture discussed in Section 5 and ran it for the ISH, Dredze, and Improved Dataset.

6.1.1 ISH Dataset

The NN was run with 100 mini batch size and for 500 iterations with 10-fold stratified cross validation. The mean accuracy obtained after training the model was 99.07%. Figure 9 shows the AUC achieved by the best classifier over the whole ISH dataset and the confusion matrix obtained with a 0.7 threshold value is shown in Figure 9. The FP rate obtained was 0.12%.



	SPAM	HAM
SPAM	921	1
HAM	0	810

Fig. 9 ROC & Confusion Matrix for ISH dataset trained on NN

When the above trained model was tested on the improved dataset, it gave a very low accuracy of 5.5%, which was expected as the improved dataset was meant to fool such classifiers. So, in the next experiment the ISH dataset and with the improved dataset and then trained on the NN, which gave an accuracy of 98.29% and an area under curve of 0.99. The ROC curve regarding the same is given in Figure 10.

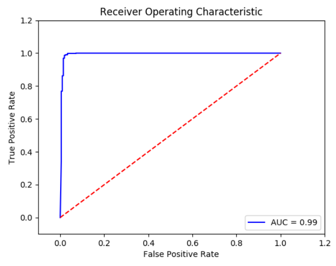
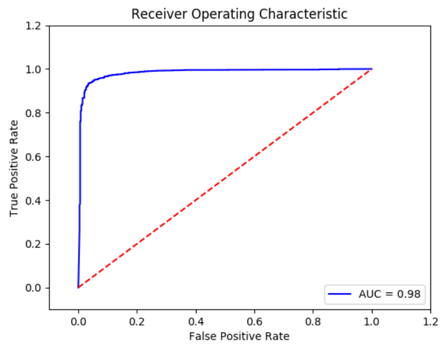


Fig. 10 ROC curve for NN when trained on Improved dataset and with ISH dataset

6.1.2 Dredze Dataset

The same NN was run on the Dredze personalized dataset and on the Dredze spam archive combined with 10-fold stratified cross validation. We got mean accuracy of 98.9% and 96.71%, respectively. The ROC curve when the NN was run on the Dredze spam archive and with the personalized dataset is shown in Figure 11 alongside its confusion matrix. The FPR achieved in this case was 0.8%. When this model was tested on the improved dataset we achieved an accuracy of 4.2%.



	SPAM	HAM
SPAM	13283	102
HAM	820	12463

Fig. 11 ROC & Confusion Matrix for Dredze dataset trained on NN

When the whole Dredze dataset was combined with the improved dataset and then trained on the NN we achieved an accuracy of 94.42%. Figure 12 below shows the ROC curve obtained for the same experiment.

The summary of all the results when trained with different combination of datasets on the NN is given in Figure 13.

As shown above, NN gave best results for the ISH dataset. It performed worse when trained on the ISH or the Dredze dataset, and then tested on the improved dataset. However, when the two datasets were combined with the improved dataset,

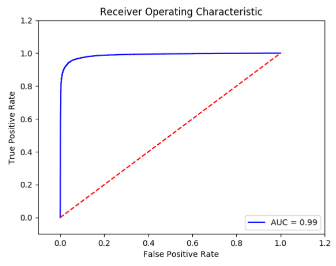
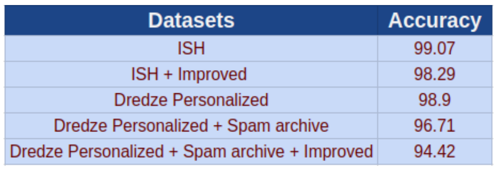


Fig. 12 ROC curve for Dredze dataset combined with improved dataset on NN



Datasets	Accuracy
ISH	99.07
ISH + Improved	98.29
Dredze Personalized	98.9
Dredze Personalized + Spam archive	96.71
Dredze Personalized + Spam archive + Improved	94.42

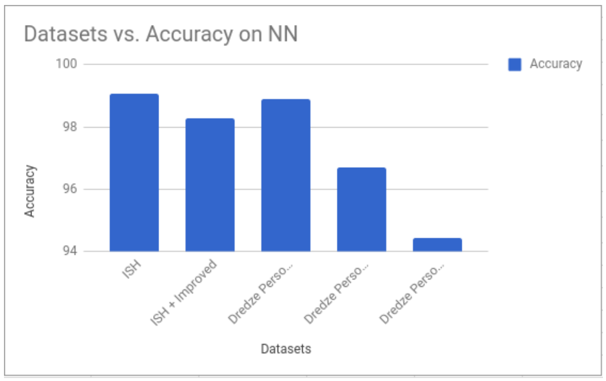


Fig. 13 Summarized Result of NN trained on different datasets

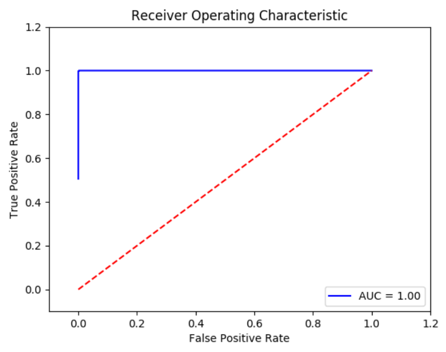
the NN was still able to perform better, however decreased the overall accuracy of the other datasets as it acted as noise for them.

6.2 Deep Neural Network results

The purpose of using a DNN was to compare the results obtained from the NN and see if the introduction of extra hidden layers actually affect the results or not. The experiments are performed on the same datasets and their combination with the improved dataset as done in the NN approach.

6.3 Image Spam Hunter

Two experiments were performed on the ISH with the same configuration as discussed in Section 5. When the DNN was trained on the ISH dataset alone we achieved a mean accuracy of 98.78%, the ROC curve and the confusion matrix are shown in Figure 14. The FPR in this case was 0% and when this model was tested on the improved dataset we achieved an accuracy of 5.24%. When the DNN was trained on the improved dataset and with the ISH dataset we achieve a mean accuracy of 98.13% and an ROC curve as shown in Figure 15.



	SPAM	HAM
SPAM	920	0
HAM	2	810

Fig. 14 ROC & Confusion Matrix for ISH dataset trained on DNN

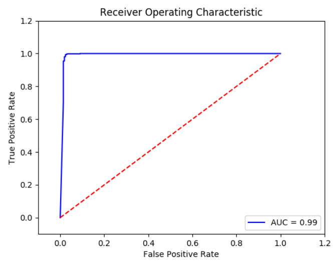
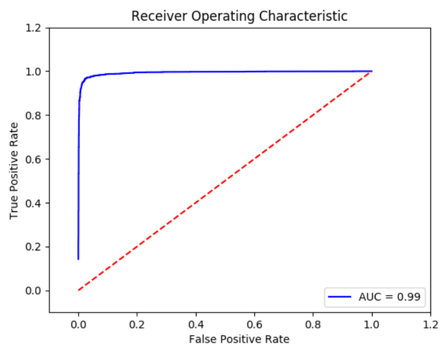


Fig. 15 ROC curve for ISH dataset and with improved dataset and trained on DNN

6.4 Dredze Dataset

After training it on the personalized dataset we achieved an accuracy of 98.95%. When the same model was trained on the personalized combined with the SPAM archive we obtained an accuracy of 96.82% and a FPR of 1%. When this model was tested on the improved dataset we achieved an accuracy of 5.9%. The ROC curve and confusion matrix obtained for the latter case is shown in Figure 16.



	SPAM	HAM
SPAM	13520	130
HAM	583	12435

Fig. 16 ROC & Confusion Matrix for Dredze dataset trained on DNN

When the Dredze dataset was combined with the improved dataset we achieved the following ROC curve of Figure 17 and a mean accuracy of 95.63%.

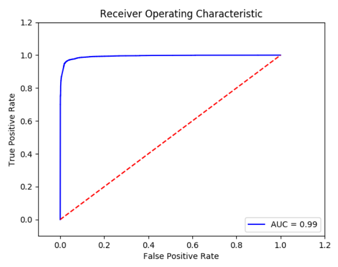
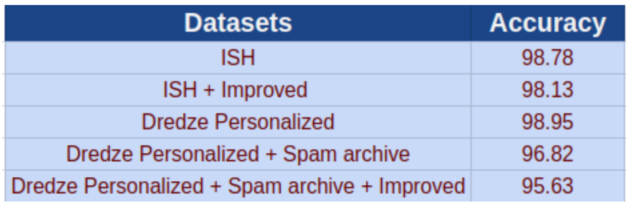


Fig. 17 ROC curve for Dredze dataset combined with improved dataset on DNN

The summary of all the results when trained with different combination of dataset on the DNN is shown in Figure 18 below.

After comparing the results we obtain from NNs and DNNs, we can conclude that the introduction of an extra layer indeed increase the accuracies for Dredze dataset with more samples but decrease the accuracy of the ISH dataset with comparable lesser samples. It also became more robust with the improved dataset.



Datasets	Accuracy
ISH	98.78
ISH + Improved	98.13
Dredze Personalized	98.95
Dredze Personalized + Spam archive	96.82
Dredze Personalized + Spam archive + Improved	95.63

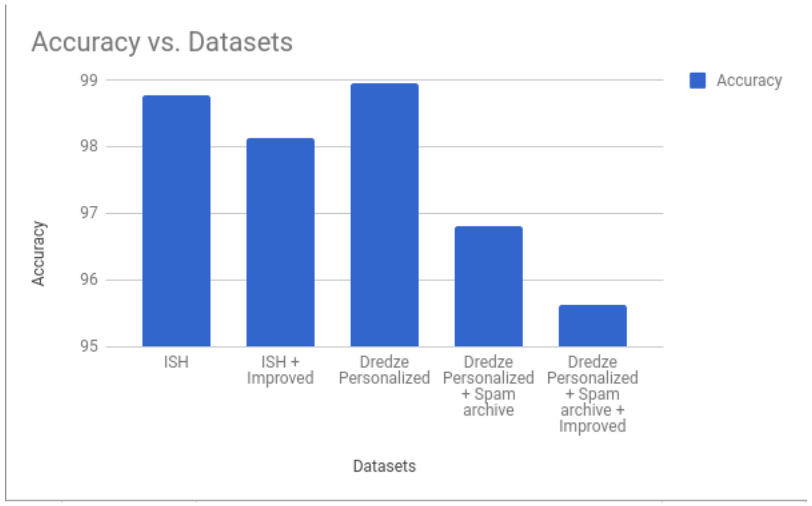


Fig. 18 Summarized Result of DNN trained on different datasets

6.5 Convolution Neural Networks & Transfer Learning results

We trained the CNN1 and the CNN2 architectures on 19924 images of SPAM and HAM, and test on 2681 images. The CNN1, CNN2, and VGG19 are trained on a GPU machine with GeForce GTX 960M, Cuda Version 8.0 and compute capability - 5.0 configuration, and each model took an average of 4 days to train. CNN1 was trained for 30 iterations, CNN2 for 25 iterations, and VGG19 for 50 iterations. For all of these models Adam optimizer was used along with binary cross-entropy. Figure 19 shows the training accuracy, training loss, validation accuracy and validation loss obtained over the 40 epochs the CNN1 model was trained on.

Then, Figure 20 shows the accuracy results for the three models when trained on the combined dataset, and when tested on the improved dataset.

From the above table it can be concluded that as the CNN2 performed little bit better as compared to CNN1 as it had more layers. Also VGG19 performed better than the other two because it was a pre-trained model on a much larger dataset. Transfer learning hence is preferable in such scenarios when there is lesser time and resources available to train your own model.

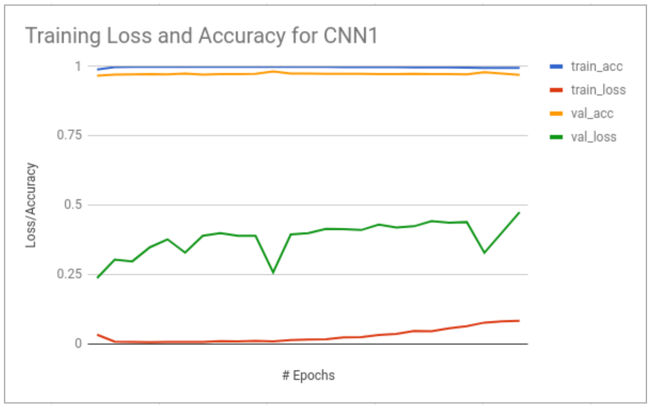
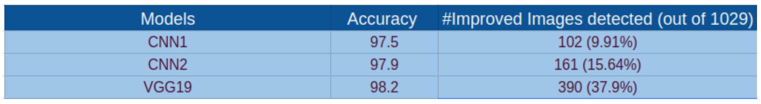


Fig. 19 Accuracy vs Loss for the CNN1 model



Models	Accuracy	#Improved Images detected (out of 1029)
CNN1	97.5	102 (9.91%)
CNN2	97.9	161 (15.64%)
VGG19	98.2	390 (37.9%)

Fig. 20 CNNs and Transfer Learning Accuracies

7 Conclusion & Future Work

In this work we make use of different real world image spam datasets and provide strong classifiers based on neural networks, deep neural networks, and convolution neural networks. We compare our results to the ones presented by Aneri et al. [2]. These techniques were able to learn even from the improved dataset provided by them. We performed different experiments with different combinations of datasets which was derived from Dredze (image archive and personalized), the image spam hunter and the improved dataset. In the CNN experiments, we matched and kept the datasets for SPAM and HAM balanced by randomly sampling HAM files from different categories over the internet. All the experiments, especially the ones with the convolution neural networks, showed really promising results because the size of the used dataset was comparatively much larger to the previous experiments performed in the past and the diversity of the HAM files.

With the advent of deep learning which make use of big data available across the Internet, even more strong classifiers are feasible. Techniques like generative adversarial networks (GAN's), introduced in year 2012 by Ian Goodfellow [6] can be used for this purpose. Using GAN's which is based on Nash equilibrium, more stronger and robust classifiers can be built. Also, object segmentation using CNN and RNN (Recurrent Neural Networks) [19] can be used to detect the segmented region of SPAMS and remove them from the images by extrapolating a background from ham images. Using such techniques, SPAM images can be converted to ham dynamically. Also, different experiments with different architectures within the CNN can be used to quantify different results. We can also use RFE (Recursive Feature Elimination) and UFS (Univariate Feature Selection), as done in [2] on the image features, when trained to neural networks and deep neural network to decrease the number of features under consideration.

References

1. A. Annadatha and M. Stamp. Image spam analysis and detection. *Journal of Computer Virology and Hacking Techniques*, 2016.
2. Aneri Chavda, Katerina Potika, Fabio Di Troia, and Mark Stamp. Support vector machines for image spam analysis. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 1: DCNET, ICE-B, OPTICS, SIGMAP and WINSYS, Porto, Portugal, July 26-28, 2018*, pages 597–607, 2018.
3. S Dhanaraj and V Karthikeyani. A study on e-mail image spam filtering techniques. In *Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on*, pages 49–55. IEEE, 2013.
4. Mark Dredze, Reuven Gevaryahu, and Ari Elias-Bachrach. Learning fast classifiers for image spam. In *CEAS*, pages 2007–487, 2007.
5. Yan Gao, Ming Yang, Xiaonan Zhao, Bryan Pardo, Ying Wu, Thrasyvoulos N Pappas, and Alok Choudhary. Image spam hunter. In *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pages 1765–1768. IEEE, 2008.
6. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
7. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
8. U Jain and S Dhavale. Image spam detection technique based on fuzzy inference system. *Master's Report, Department of Computer Engineering, Defense Institute of Advanced Technology*, 2015.
9. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
10. T Kumaresan, S Sanjushree, K Suhasini, and C Palanisamy. Image spam filtering using support vector machine and particle swarm optimization. *Int. J. Comput. Appl*, 1:17–21, 2015.
11. Chih-Chin Lai and Ming-Chi Tsai. An empirical performance comparison of machine learning methods for spam e-mail categorization. In *Hybrid Intelligent Systems, 2004. HIS'04. Fourth International Conference on*, pages 44–48. IEEE, 2004.
12. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
13. Er-Xin Shang and Hong-Gang Zhang. Image spam classification based on convolutional neural network. In *Machine Learning and Cybernetics (ICMLC), 2016 International Conference on*, volume 1, pages 398–403. IEEE, 2016.
14. Tazmina Sharmin, Fabio Di Troia, Katerina Potika, and Mark Stamp. Convolutional neural networks for image spam detection. *Inf. Secur. J. A Glob. Perspect.*, 29(3):103–117, 2020.
15. M Soranamageswari and C Meena. Statistical feature extraction for classification of image spam using artificial neural networks. In *Machine Learning and Computing (ICMLC), Second International Conference on*, pages 101–105. IEEE, 2010.
16. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

17. Mark Stamp. *Information security: principles and practice*. John Wiley & Sons, 2011.
18. Lance Whitney. Report: Spam now 90 percent of all e-mail. *CNET News*, 26, 2009.
19. Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.