DEEP LEARNING BASED PHISHING WEB PAGE DETECTION

TEVFİK UĞUR BASTEM

JANUARY 2022

ÇANKAYA UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER'S THESIS IN

COMPUTER ENGINEERING

DEEP LEARNING BASED PHISHING WEB PAGE DETECTION

TEVFİK UĞUR BASTEM

JANUARY 2022

ABSTRACT

DEEP LEARNING BASED PHISHING WEB PAGE DETECTION

BASTEM, Tevfik Uğur

Master of Science in Computer Engineering

Supervisor: Assist. Prof. Dr. Abdül Kadir GÖRÜR

Co-Advisor: Assoc. Prof. Dr. Ali Seydi KEÇELİ

January 2022, 53 pages.

With the increase in the usage of e-commerce, social media and digital entertainment services, there is a tremendous increase in phishing activities. In this study, based on the observation of phishing activities, a study has been carried out to detect phishing websites with deep models and transfer learning. Within the scope of the study, a data set containing a total of 2852 screenshots, consisting of real and fake screenshots of websites such as adobe, amazon, apple and microsoft etc. was used. The results obtained by using transfer learning from AlexNet, VGG16 and RESNET50 models as well as the proposed multi- input CNN model were analyzed. Promising results are obtained from the experiments. The effects of the obtained findings on other future studies were discussed.

Keywords: AlexNet, VGG16, RESNET50, Transfer learning, Multi-input CNN, Phishing

ÖZ

DERİN ÖĞRENME TABANLI KİMLİK AVI WEB SAYFASI TESPİTİ

BASTEM, Tevfik Uğur
Bilgisayar Mühendisliği Yüksek Lisans

Danışman: Dr. Öğr. Üyesi Abdül Kadir GÖRÜR
Ortak Danışman: Doç. Dr. Ali Seydi KEÇELİ
Ocak 2022, 53 sayfa

E-ticaret, sosyal medya ve dijital hizmetlerin kullanımının artmasıyla birlikte oltalama faaliyetlerinde muazzam bir artış yaşanmaktadır. Bu çalışmada, oltalama faaliyetlerinin gözlemlenmesinden yola çıkarak, transfer öğrenme yöntemleri ile sahte web sitelerinin tespitini yapacak bir çalışma gerçekleştirilmiştir. Çalışma kapsamında, adobe, amazon, apple, microsoft gibi web sitelerinin gerçek ve sahte ekran görüntülerinden oluşsan toplam 2852 ekran görüntüsü içeren bir veri setinden yararlanılmıştır. AlexNet, VGG16, RESNET50 transfer öğrenme yöntemleri yanı sıra kendi geliştirdiğimiz çok girişli CNN modelini kullanarak, elde edilen sonuçlar analiz edilmiştir. Elde edilen bulguların gelecekte yapılabilecek diğer çalışmalara etkisi tartışılmıştır.

Anahtar Kelimeler: AlexNet, VGG16, RESNET50, Transfer öğrenimi, Çok girdili sinir ağı, Oltalama

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to co-advisor, previously my advisor, Assoc. Prof Dr. Ali Seydi KEÇELİ, who provided me with all kinds of help, sacrifice during my studies, and expanded my horizons in academic studies for a long time.

I would like to thank my dear advisor, Assist. Prof Dr. Abdül Kadir GÖRÜR, who helped me to maintain my research for the rest of the study, cordially for his valuable comments, guidance, and counseling.

I wish to thank the examining committee for their kindness during the presentation of this thesis.

I must express my profound sincere and gratitude to my mother, Yıldızhan BASTEM, my father, Ahmet BASTEM, and my sisters, Pınar GENÇ and Zeynep YAVUZEKİNCİ who have brought me to where I am today and have always supported me without sparing their efforts.

Finally, my sincere acknowledgement also goes to my wife, Hatice Nazlı BASTEM, who stood by me unconditionally during my most stressful times and gave me support and strength with her presence.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CNN	**:**Convolusional Neural Network
APWG	**:**Anti Phishing Working Group
DOM	**:**Document Object Model
HTML	**:**HyperText Markup Language
BVM	**:**Ball Support Vector Machine
SVM	**:**Support Vector Machine
URL	**:**Uniform Resource Locator
IP	**:**Internet Protocol
ILSRVC	**:**ImageNet Large Visual Recognition Challenge
TN	**:**True Negative
TP	**:**True Positive
FN	**:**False Negative
FP	**:**False Positive
TPR	**:**True Positive Rate
FNR	**:**False Negative Rate
ReLu	**:**Rectified Linear Unit
etc	**:**Et Cetera, Other Similar Things

CHAPTER I

INTRODUCTION

In working against phishing attacks, phishing prediction heuristics are essential in developing solutions. However, phishing attacks continue to increase today and reflect the need for higher-precision solutions. It is getting more difficult to detect these phishing attacks with traditional methods. Deep learning approaches come to the fore in order to overcome this problem.

The most widely used of these methods are convolutional neural networks (CNN). According to the designed architecture, CNN takes the raw images as input and learns the attributes at various levels. Moreover, the CNN is classified using different machine learning methods and extracting the features obtained from end-to-end or after the CNN itself. For example, from the amazon screenshots in the dataset, the necessary attributes are automatically learned to distinguish which screenshots are fake and real. In this thesis, different deep learning approaches have been tested to determine the screenshots of fake websites.

Models using the AlexNet and VGG16 architectures directly, and a multi-input CNN model using with ResNet 50 model. In addition, within the scope of the thesis, the VGG16 network was used for transfer learning. The general flow diagram of the proposed method is given in Figure 1.1. Higher classification success rates are obtained by combining the features obtained from the raw images with the multi-input network and the features obtained by transfer learning on a single network.

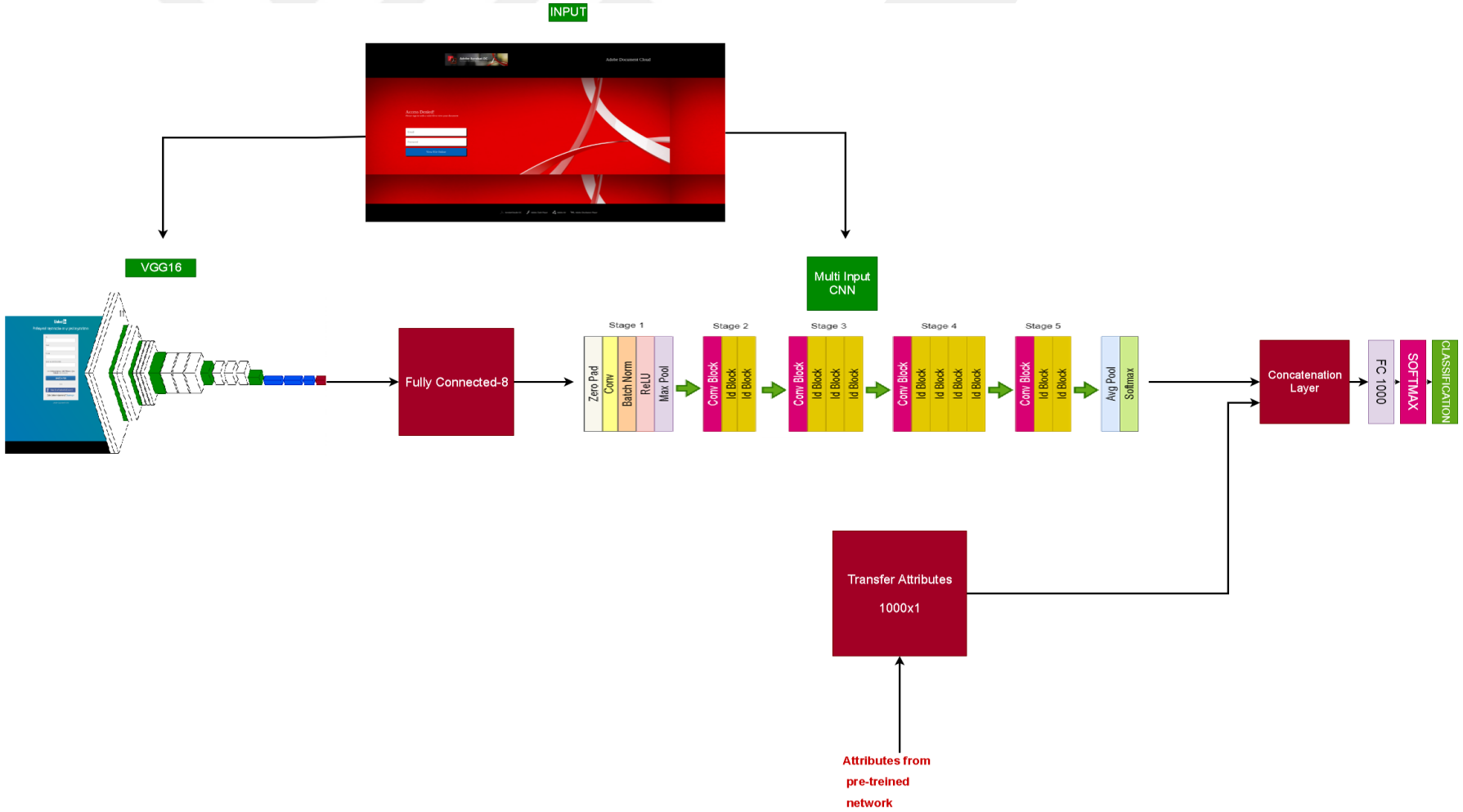

Figure 1.1: Multi-Input CNN Model

1.1 PROBLEM STATEMENT

The development of the information technologies has brought many conveniences to our lives and has also caused many problems. It also provides opportunities for individuals who want to commit crimes. In the World millions of credit card information are stolen, are made by software that comes with simple fishing e-mails. Malware can be written in many programming or scripting languages and can be transported in files. Viruses, Trojans, malicious emails, keyboard listening systems, url injection are the most common malwares. In addition to these, there are phishing attacks aimed at increasing deception and fraud [1]. According to Jakobsson and Meyers (2007), phishing can be used to prevent user-sensitive information through the illegitimate website that is entirely similar to the target an action aimed at obtaining information (such as personal identification number, password, credit card number) defines as [9].

1.2 AIM OF THE THESIS

Thesis proposal, develop a deep learning-based phishing web page recognition approach. In addition, the classical visual descriptors will be tested for this purpose and their performance will be matched with the results obtained from the deep-based methods. Deep Learning methods will be tested for web page similarity. A CNN by using publicly available datasets will be designed and implemented first [2]. Then, transfer learning by using well-known pre-trained models like VGG16 and AlexNet [3, 47] are tested. In transfer learning the pre-trained models will be used as feature extractor and classical machine learning methods will be employed on deep features for recognition. These methods will be applied on both whole image and image patches. In addition to classical CNN models, multi-input CNN model is developed, and raw images and pre-trained features are taken as inputs by this model. Transferred and learned features are combined to detect phishing web pages.

1.3 APWG PHISHING REPORTS

APWG publishes reports every year to draw attention to phishing activities. The main purpose of the group is to analyze phishing attacks and to report phishing attacks. In the report for the 4th quarter of 2020, it is observed that increase in phishing activities continues throughout the year as shown in Figure 1.2. In the world, the

number of phishing attacks web sites between January and December has increased from 50,000 to 250,000[10].

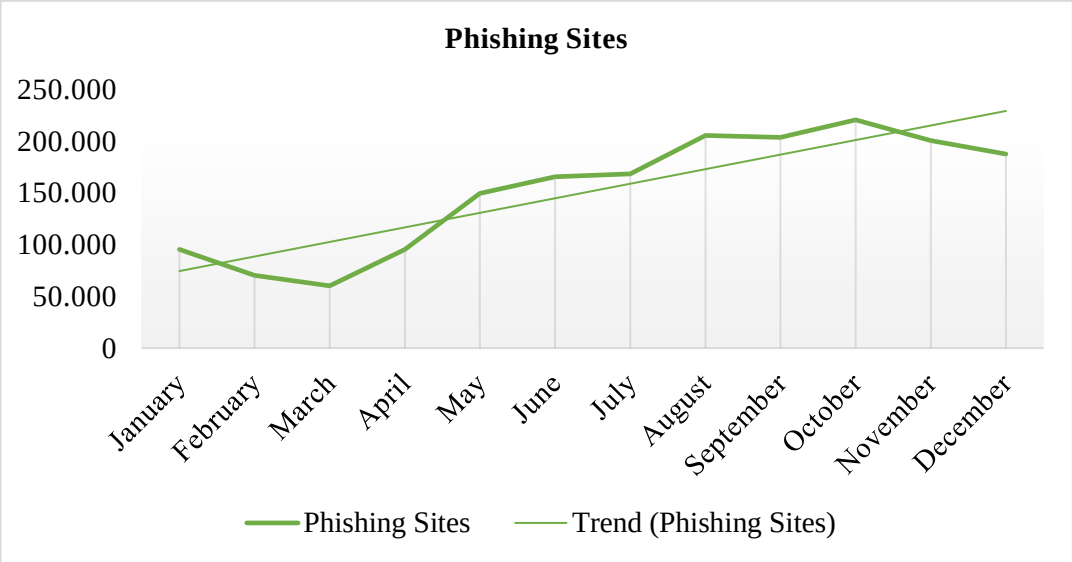

Figure 1.2: Phishing Activity, 2020 [10]

1.4 GOOGLE PHISHING REPORT

According to another study, in 2020, it was reported by Google that Two Million phishing websites were created. It was determined that Figure 1.3 showed an increase of 19.91% when compared to 2019. This result showed the effect of the increase in cyber-attacks with the onset of the coronavirus epidemic. According to research from Google, an average of 46,000 new phishing websites were created every week in 2020. According to another result obtained in the study, it was observed that there was a very rapid increase in cyber-attacks between February and May.

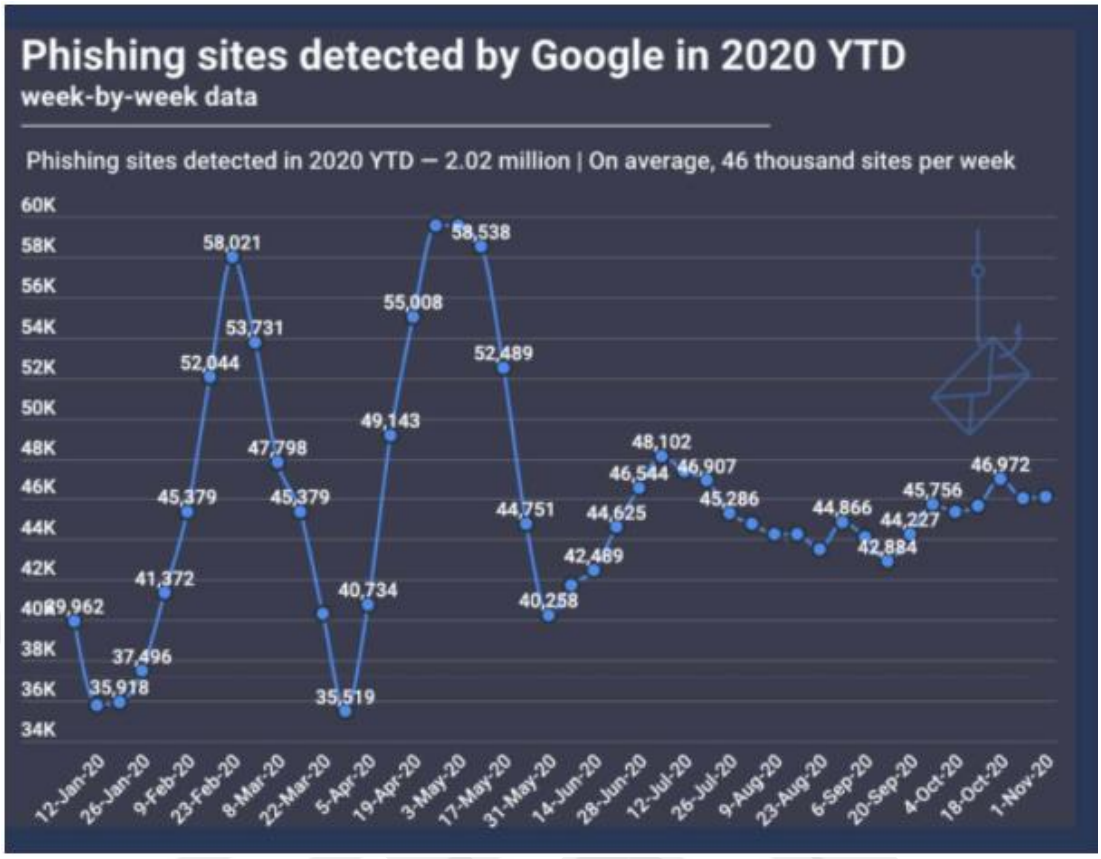

Figure 1.3: Two Million Phishing Websites Created in 2020- Phishing sites detected by Google [44]

1.5 THESIS ORGANIZATION

In the first part of the thesis, basic information, problem situation and the purpose of the thesis are given in order to gain a general point of view to the thesis. The organization of other departments is presented below:

In Chapter 2, the concepts of transfer learning and CNN algorithms are defined, and each component is explained.

In Chapter 3, the literature related to the thesis has been examined.

In Chapter 4, the dataset has been defined.

In Chapter 5, the tools and libraries have been explained.

In Chapter 6, the experimental setup has been explained and experimental results have been discussed.

In Chapter 7, whether the research has achieved its purpose has been explained. In addition, forward-looking application areas and suggestions have been discussed.

CHAPTER II

BACKGROUND

2.1 TRANSFER LEARNING

Transfer learning approaches are influenced by the human learning model. For the solution of the problem, previously obtained solutions are used [24] [25]. In summary, it means using a network trained with one dataset for a different dataset. Transfer learning approaches gained momentum with the study of Prat (1993) [25]. In this study, different data wanted to be classified using the coefficients learned from a different dataset.

2.2 CNN

Convolutional Neural Network (CNN) is one of the popular deep neural networks. CNN models are often used in image processing and take images as input. This algorithm, which captures the features in the images with different operations and classifies them, consists of different layers [26]. It gets its name from the linear mathematical operation between matrices.

2.2.1 CNN Architecture

CNN architecture consists of Input Layer, Convolutional Layer, Activation Layer, Pooling Layer, Fully Connected Layer, DropOut Layer, Classification Layer, Softmax Layer and Normalization Layer. The CNN architecture is shown in Figure 2.1 [54].

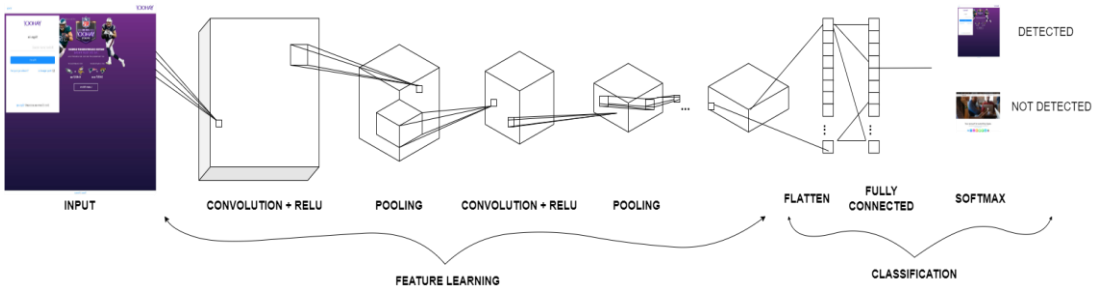

Figure 2.1: CNN Architecture [54]

2.2.2 Input Layer

It is the first layer of the convolutional neural network, also called the data input layer. The input data resolution and size should be determined according to the architecture of the model to be designed [28]. The input data should be chosen well. Because of this, data will affect the model's performance, training time, test time, and memory requirement. It is necessary to select a large number of input data to increase the success rate. It also causes a more extended training and testing time. For this, higher processor and memory capacity is needed.

2.2.3 Convolutional Layer

The primary layer of CNN is the convolution layer. Responsible for detecting input image properties. These layer parameters focus on using learnable kernels and apply filters to the image to extract features from the image [27]. Filters, which play an essential role at this stage, provide a new small matrix from the data [21]. The filter to be chosen will affect the success rate of the network and the training process. Therefore, an ideal filter should be chosen [29]. Convolution is performed by using filters in sizes such as 1x1, 2x2, 3x3, 5x5, 7x7, and at this stage, a feature map is created. In this thesis, as an example; the 7*7 filter is used for the AlexNet model. Figure 2.2 shows the convolution process.

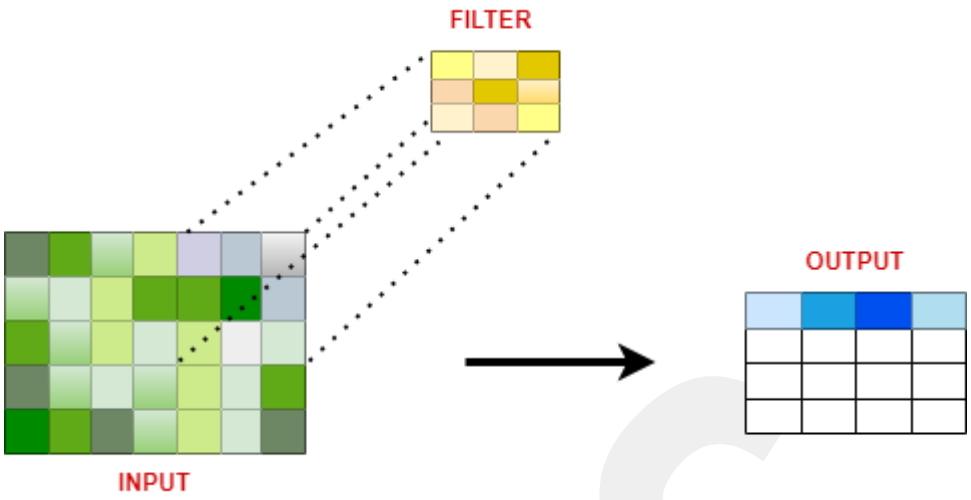

Figure 2.2: Convolution Operation [21]

Each filter contains its edge detection structure. Although the filters are usually square matrices, filters with different rows and columns are also used in some studies. A tensor is obtained by superimposing each filter on which the convolution process is applied to the input image. The number of filters used determines the depth of the tensor to be created. The convolution process is calculated according to the formula given below [31].

W_{out} :The size of the new image to be obtained after the convolution operation.
W_{in} :The size of the input image.
F : Filter size.
Stride-S: Convolution step size.
Padding-P: Add frame to image.

$$W_{out} = \frac{W_{in} - F + 2P}{S+1} \quad (2.1)$$

2.2.4 Activation Layer

Activation functions are used when transmitting the output value in neurons from one layer to another. The initial value is determined to be able to decide whether the output value should be transferred to other layers. Because the neuron may not know what the limits of the real value will be and the artificial neuron may be in the value range $(-\infty, +\infty)$. Therefore, activation functions are needed to decide the neuron's activation states. Thus, it will be able to control the output value produced by a neuron

and it will be possible to decide whether to see the neuron actively or not [32]. Activation functions can be linear or non-linear. While choosing the activation function, a nonlinear and differentiable function was preferred. Because our model can be learned better and the factors that give better performance are taken into consideration. For these reasons, the Relu(Rectifier linear unit) activation function is preferred in the thesis. Although the ReLu activation function looks linear, it is a non-linear function. Relu graph shows in Figure 2.3.

$$g(x)=\max(0,x)=\begin{matrix}x \text{ if } x\geq 0\\ 0 \text{ if } x<0\end{matrix} \tag{2.2}$$

The ReLu function is as shown above. It gives an output x if x is positive and 0 otherwise [32].

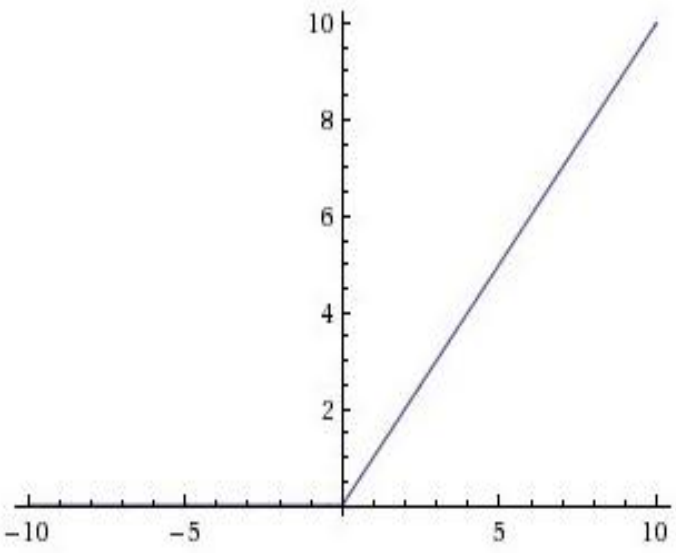

Figure 2.3: ReLu Graph [32]

The range of ReLu is [0, inf). This means it can blow up the activation.

2.2.5 Pooling Layer

After the convolution process (2015), reduces the parameter and data size in the network. As a result of the pooling process, makes the neural network faster. A certain filtering process carries out data size reduction. Pooling Layer operates

independently on each channel of the input. These layers are maximum (maximum pooling), minimum (minimum pooling) or average (average pooling) pooling [34]. In our study, max pooling, known to give the best results before, was applied for AlexNet and VGG16 learning models. Figure 2.4 shows an example of max pooling operation. Pooling layer down samples the volume spatially, independently in each depth slice of the input volume. Equation [51] shows the extraction pooling feature map formula.

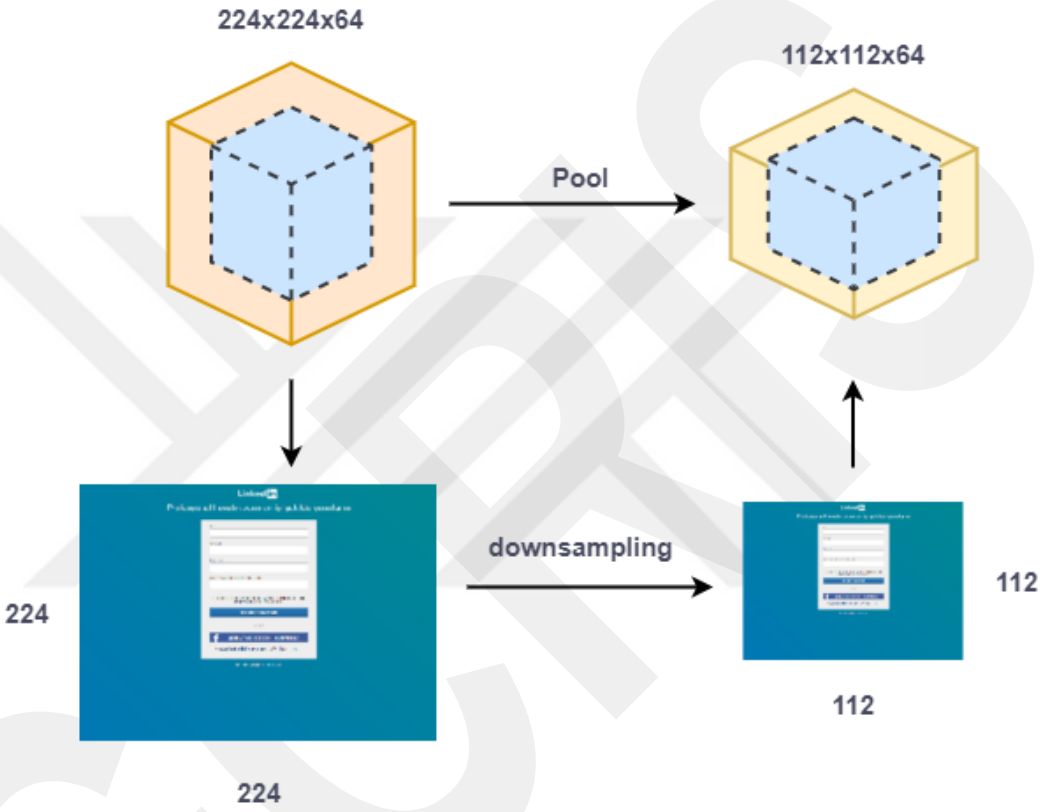

Figure 2.4: Example of Max Pooling [52]

I_x= Input shape

P= Pooling window size

S= Stride

$$\text{Max Pool Output} = \left[\frac{I_x - P}{S}\right] + 1 \quad (2.3)$$

A new output was obtained in the process by choosing the highest value from the sections with the same color in the max pooling process using the filter and stride value.

2.2.6 Fully Connected Layer

This layer takes the feature maps as input and prepares them for the classification [35]. The fully connected layer is the layer where learning with artificial neural networks. Therefore, as in logistic regression, forward propagation and backward propagation are performed in this layer. In the convolutional neural network architecture, the fully connected layer comes after the consecutive convolution, ReLu and pooling layers. This layer is dependent on all fields of the previous layer. The number of layers can vary depending on the architecture. One fully connected layer looks for high-level features that have a high degree of association with a class, by looking at the neurons with weights indicating these properties, which belongs to the class [37]. In convolutional neural network architecture, if the last layer matrix size A(x) and the fully connected layer B(x) matrix are selected, the total weight matrix AxB is formed. For this reason, this layer is called the fully connected layer. Figure 2.5 shows fully connected layer structure.

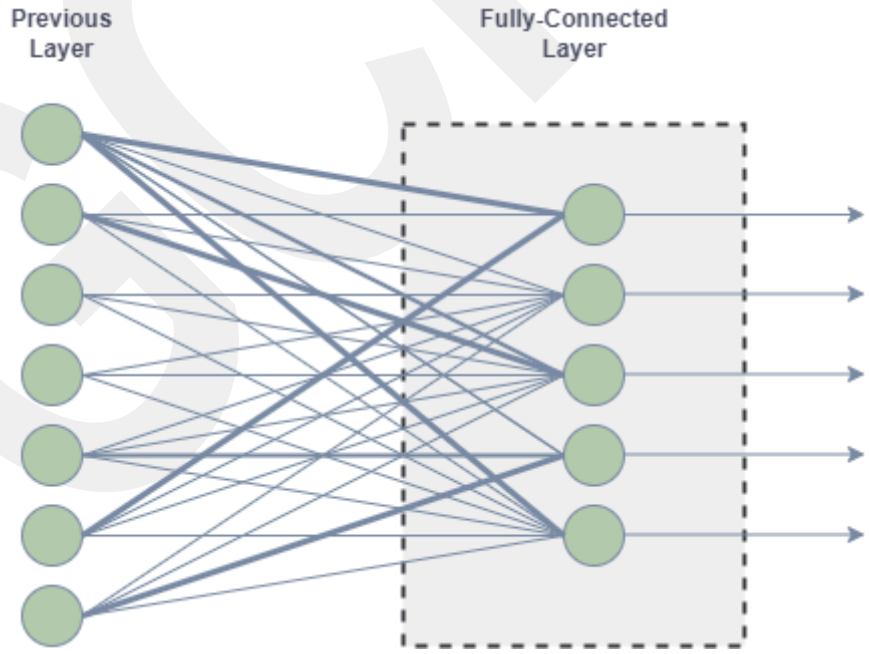

Figure 2.5: Fully connected layer [37]

2.2.7 DropOut Layer

In multi-layer artificial neural networks, while the neural network is being trained, the network called overfitting memorization takes place. This is undesirable situation. To prevent the network from being memorized, some nodes in the network that memorize are eliminated. Thus, network attempts are made to eliminate memorization [38]. Hinton et al. suggested, the Dropout layer as an editing layer for fully connected layers. In Figure 2.6, the artificial neural network and after the dropout structure is shown.

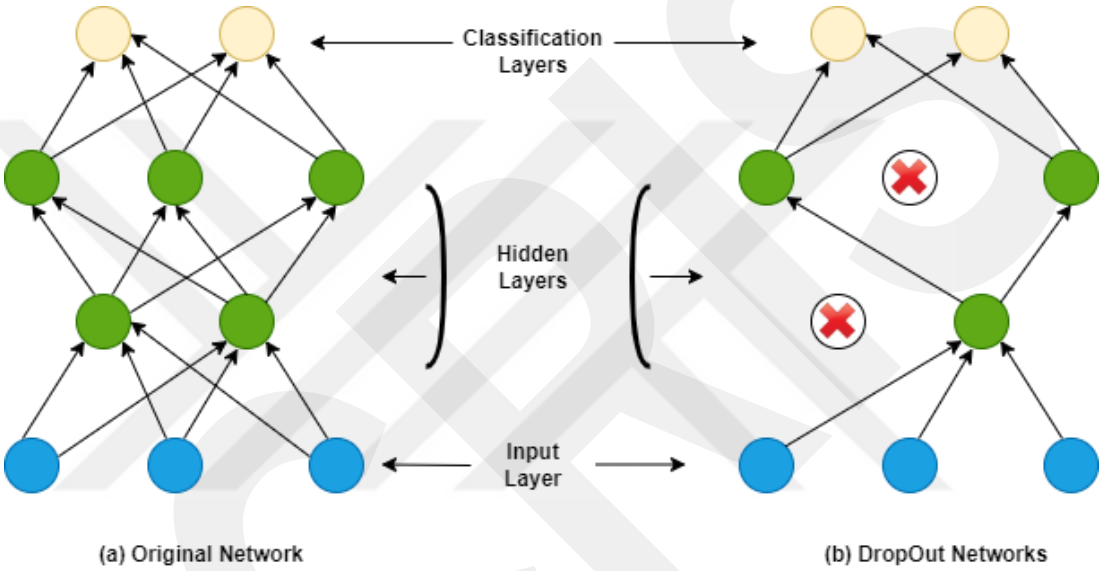

Figure 2.6: (a) Artificial neural network (b) Dropout applied neural network (Crossed neurons dropped from the network) [38]

2.2.8 Classification Layer

Classification after fully connected layer, produces as many results as the number of items to be classified. Each of these results represents a class. For classification, the softmax classifier is generally used, although it is known that there are different types of classifiers [39].

2.2.9 Softmax Layer

Softmax layer receives the input and performs the classification. Softmax indicates which class the probabilistic input data belongs to. In the deep learning network, for each class, outputs the probability value. Cross-entropy is used for these operations [40].

2.2.10 Normalization Layer

Training of deep convolutional neural networks takes a serious process computationally. Activation of neurons can be normalized to reduce the training time. The normalization layer is effective to stabilize the hidden layers. Usually, normalization is performed after Relu layer [41].

2.3 EVALUATION METRICS

2.3.1 Confusion Matrix

In order to evaluate the performance of classification models used in machine learning, the confusion matrix is often used, in which the predictions of the target attribute and the actual values are compared.

Table 2.1: Confusion Matrix

Confusion Matrix		**Actual**	
		Image Detected Positive (1)	**Not Detected Negative (0)**
Predicted	**Image Exist Positive (1)**	TP[1,1] True Positive	FP[1,0] False Positive
	Image Not Exist Negative (0)	FN[0,1] False Negative	TN[0,0] True Negative

The Table 2.1 is a confusion matrix of the output of a model set up for binary classification. Positive and Negative terms in this matrix represent the classes to be separated.

2.3.1.1 True Positive

True positives are values where the true value is 1 and the predicted value is 1.

2.3.1.2 True Negative

True Negatives are instances where the true value is 0 and the predicted value is 0.

2.3.1.3 False Positive

False Positives are instances where the true value is 0 but the predicted value is 1.

2.3.1.4 False Negative

False Negatives are instances where the true value is 1 but the predicted value is 0.

2.3.2 Accuracy Rate

Generally, it is a measure of how often the classifier guesses correctly. Accuracy value is a value between 0 and 1.

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+TN+FN} \tag{2.4}$$

2.3.3 Precision

It is a measure of how accurately predicted from all classes.

$$\text{Precision}=\frac{TP}{TP+FP} \tag{2.5}$$

2.3.4 Recall

It is a metric that shows how many of the predicted transactions is positively predicted.

$$\text{Recall}=\frac{TP}{TP+FN} \tag{2.6}$$

2.3.5 F1-Score

F1 Score value shows the harmonic mean of Precision and Recall values.

$$\text{F1 Score}=2*\frac{\text{Precision}*\text{ Recall}}{\text{Precision + Recall}} \tag{2.7}$$

CHAPTER III

LITERATURE REVIEW

Jain and Gupta [5] estimated based on screenshots of suspicious web pages. They stated that using image contrast and clustering of key points with the k-means algorithm (k-means) makes it possible to predict image similarities. They mentioned that visual similarities could also be determined through optical character recognition by converting them to text and comparing the results. This method better addresses the zero-day phishing issue. However, the similarity classifier must be continuously trained for the determined approach to achieve the desired result.

In another study, Rosiello et al. [6] DOM tree by comparing the HTML tags and the fake web sites are aimed to be detected. This method is directly effective against phishing websites. For a web page to give the desired result, view uniquely with the DOM tree structure should be defined. However, there is an obvious downside of this method. Phishing attackers use different HTML tags in case it gets the look of the same website, and the fake website will not be detected.

Bohunsky et al. [7] to detect fake web pages to determine "visually" method display fake and original web pages. This method is defined to make screenshots small rectangular cut them into pieces. In this way, the visualization of two web pages by comparing the box structure, determines its correlation.

In the study by Gowtham et al. [11], legitimate and phishing web pages were researched in depth. Based on the analysis, heuristics are proposed to extract 15 features from web pages of similar type. In the system created, before applying heuristics to the web pages, two pre-scan modules were used. With the help of the modules used, unnecessary calculation is reduced. Also, false positive rate has been reduced without compromising on false negative. As a result of the study, an accuracy rate of over 90% was obtained. The experimental results and the proposed method effective for protecting users from online identity attacks.

In the study by Li et al [12], the minimum requirement for phishing website detection enclosing BVM (Ball Support Vector Machine) is recommended. This approach aimed to provide high speed and high accuracy for phishing website detection. In order to increase the integrity of the vectors, studies have been carried out. First, according to the DOM tree, the topology of the website structure analysis was made. Then, the topological features of the website are extracted. Then, feature vectors were determined by the classifier. The proposed method is compared with SVM. It has been observed that the proposed method has higher detection sensitivity. The accuracy of the proposed system and its validity were evaluated.

Nguyen and Nguyen [13] detect phishing websites using machine learning methods by using URL and page content. In the study, obtained from URL and content decision tree, random forest, support vector machine, naïve bayes and neural network methods have been compared. According to the test results, random forest method is the most successful predictor.

Kazemian and Ahmed [14] for the detection of malicious websites, compared k-NN, support vector machine, naïve bayes classifier and kmeans methods. For supervised methods, accuracy of over 89% was obtained.

Mohammad et al. [15] proposed a self-constructed neural network method for detecting phishing websites. The proposed automatically creates the network and shows a high classification success.

Abdelhamid et al. [16], [17] proposed a method for relational classification data mining to divide websites into three different classes. According to experiments, relational classifier method has higher accuracy than other methods.

Moghimi et al. [18] proposed a method for detection of phishing websites that were designed to steal banking information. This method ensures that suspicious websites of IP addresses and by detecting URL addresses, in the study, 3066 websites were examined, and the recommended method has been successful at a rate of 99.14%.

Using a heuristic approach is an effective method for website-based phishing detection of suspicious websites. Ludl et al. [19] in their study, suggested a method that used a heuristic-based approach to detect and classify website-based phishing sites specifically targeting HTML and URL. In this study, a data set of 18 different phishing websites was used. As a result, it was seen that 16.9% of HTML content and 0.4% of URL content were misperceptions.

In another study, Yi et al. [42] proposed a method for detecting phishing websites that steal identity and credit card information. The study, using the deep learning method, aimed to block IP numbers of suspicious websites and these websites by detecting the URL address. In the study, as a data set, real data is used. As a result, the proposed method has reached 90% accuracy in detecting phishing websites.

In another study, Pan et al. [43] for phishing detection, proposed to analyze websites identity. They hypothesized that websites' title, description, copyright, etc. features could change on phishing websites. Contrary to what was predicted in the study, 29% recommended method was found to be unsuccessful.

CHAPTER IV

EVALUATION

4.1 DATASET

In the thesis, from 2782 screenshots of 15 websites the resulting dataset was used [8]. The distribution of the dataset is as follows in Table 4.1

Table 4.1: Distribution Of Dataset

NAME OF DATASET	NUMBER OF DATA
Adobe	70
Amazon	29
Alibaba	76
Apple	64
Boa	116
Chase	111
Dhl	109
Dropbox	115
Facebook	144
Linkedin	38
Microsoft	118
Paypal	214
Wellsfargo	134
Yahoo	114
Other	1400

The dataset weight is illustrated in Figure 4.1.

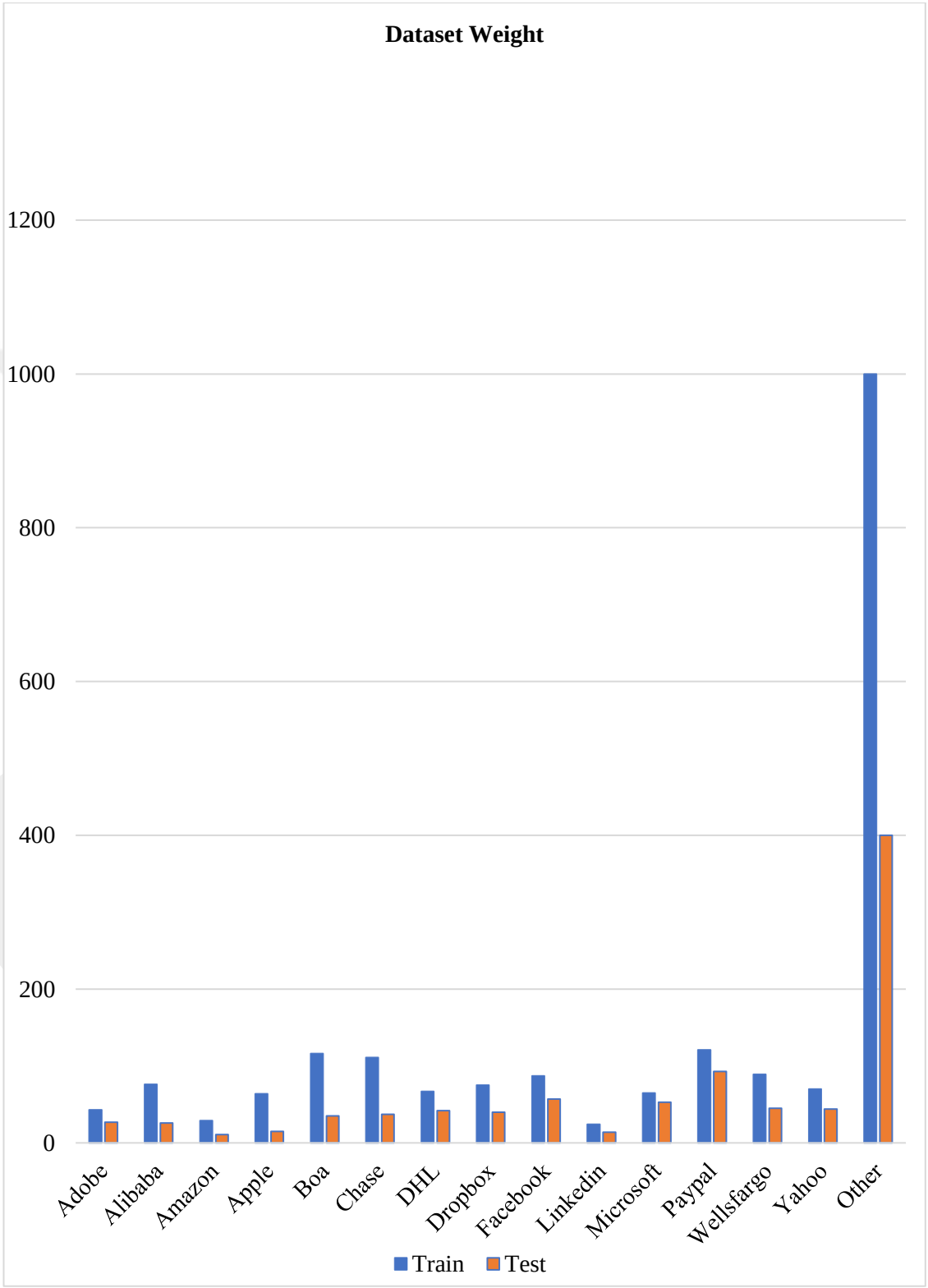

Figure 4.1: Dataset Weight

CHAPTER V

TOOLS AND LIBRARIES

In this section, tools, Multi-Input CNN algorithm (our own work) and some CNN algorithms are mentioned. These algorithms are AlexNet, VGG16 and ResNet50. The reasons why we choose these algorithms and use them in our study are that these algorithms get high scores in ILSRVC competitions and bring new features to the CNN architecture.

5.1 AlexNet

AlexNet [20] architecture at the 2012 ImageNet competition educated with about a million images that made their name known feature, extremely successful in classifying images. It is a CNN model. Figure 5.1 [20] illustrate architecture of AlexNet, which consists of eight layers; the first five are convolutional layers, some of them maxpooling and the last three layers are fully connected layers. There are also input and output layers. AlexNet architecture, 1000 objects designed to classify. In Alexnet architecture approximately 60 million parameters were used and the parallel pair. It is also the first model to run on a GPU. Table 5.1 shows structural details of AlexNet [46].

Table 5.1: AlexNet Parameter Count [46]

AlexNet Network - Structural Details													
Input			**Output**			**Layer**	**Stride**	**Pad**	**Kernel Size**		**in**	**out**	**# of Param**
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
Total													**62378344**

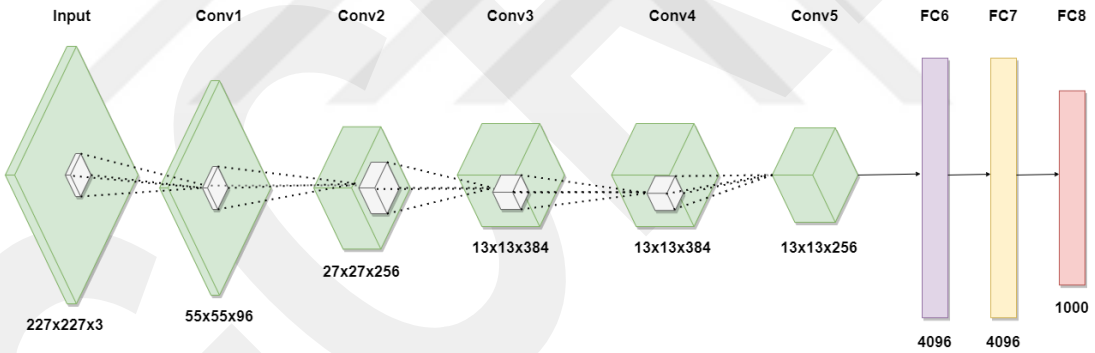

Figure 5.1: An illustration of AlexNet layers [20]

AlexNet uses ReLu as activation in non-linear parts. In previous standard neural networks, tanh or sigmoid was used. These function types are shown in Figure 5.2 [45].

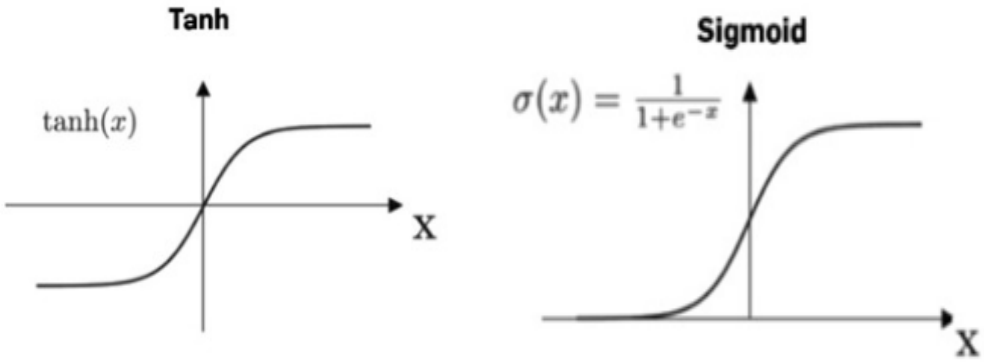

Figure 5.2: Activation Function Types [45]

5.2 VGG16

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition" [22]. Model, from 14 million in 1000 classes in ImageNet, a dataset of multiple images 92.7% reaches the first 5 test accuracy. VGG16 improves AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. The architecture of VGG16 is shown in Figure 5.3 [23]. If we examine the architecture; The VGG-16 architecture consists of convolutional, pool and fully connected layers. A total of 21 main layers occurs [22]. This architecture has an increasing network structure. The image input resolution is 224 × 224 pixels. Convolutional layer filter size is 3 × 3 pixels. In this architecture, the last layers which are fully connected are used for feature extraction. Table 5.2 shows structural details of VGG16 [46].

Table 5.2: VGG16 Parameter Count [46]

VGG16 - Structural Details													
#	Input Image			Output			Layer	Stride	Kernel Size		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	256	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
Total													**138423208**

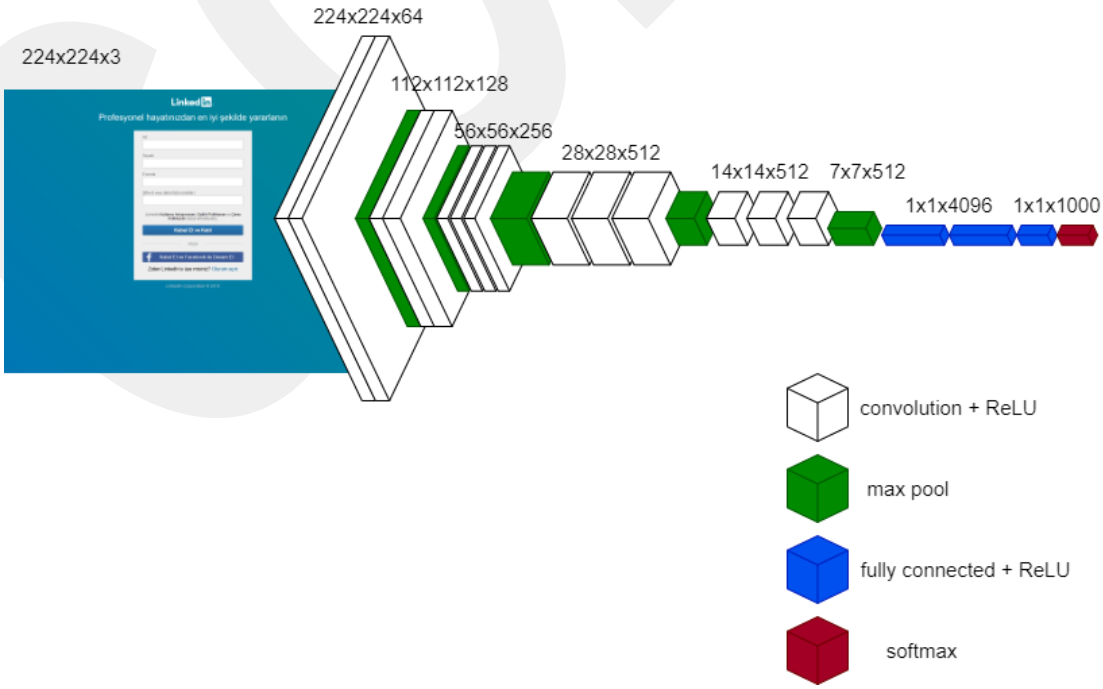

Figure 5.3: An illustration of VGG16 Architecture [23]

5.3 ResNet 50

ResNet is a network structure proposed by He Kaiming et al. [48] from Microsoft Research Asia in 2015, and was the winner in the ILSVRC-2015 classification task. The ResNet-50 model consists of 5 stages. Each stage has a convolution and identity block. Each convolution block and each identity block has 3 convolution layers [49]. The ResNet-50 has over 23 million trainable parameters. Figure 5.4 shows ResNet 50 architecture.

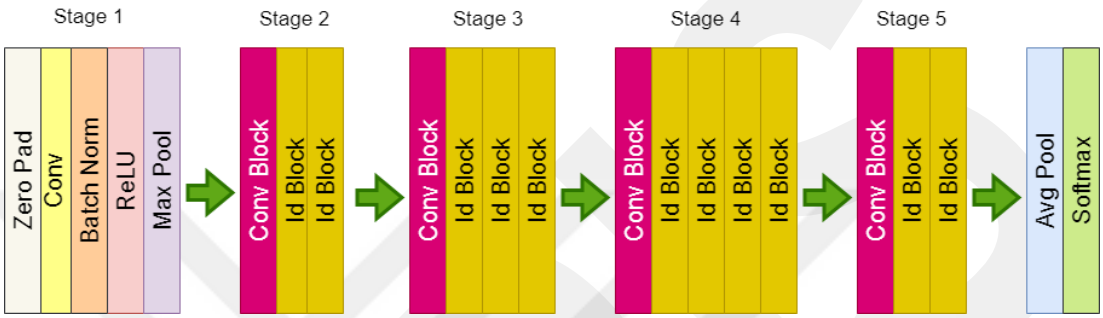

Figure 5.4: RESNET 50 Architecture [50]

ResNet has emerged to reduce the number of errors that occur as the network gets deeper and the number of trains increases.

5.3.1 Vanishing Gradient Problem

Due to Vanishing/Exploding Gradient problem, training of deep neural networks is difficult. In a Convolution Neural network, when we stack multiple layers, the training error should decrease in theory, but in practice or reality adding more layers to the CNN (making the CNN deeper) causes the training error to increase rather than decrease.

5.3.2 Residual Block

Residual learning framework is used to reduce training and testing errors in a deep network [48]. Residual Blocks as part of the ResNet architecture. In a residual block, input x is added directly to the output of the network. Figure 5.5 shows a building block of resudial learning.

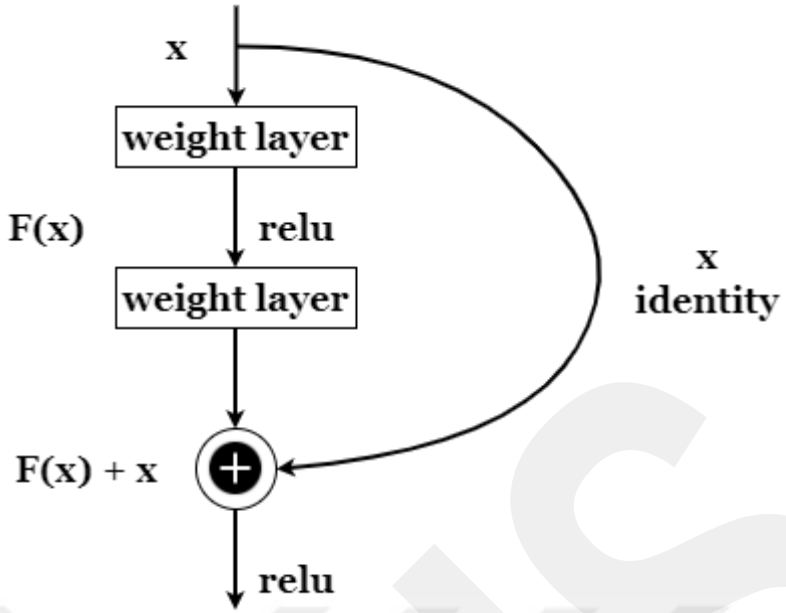

Figure 4.5: Residual learning: a building block [48]

According to the formula, the input x is added directly to the output of the network. This $F_{(x)}$+ x path is known as shortcut connection.

$H_{(x)}$:Initial mapping.

$F_{(x)}$:Residual function.

x : Residual block.

$$H_{(x)}=F_{(x)}+x \tag{5.1}$$

5.4 MULTI INPUT CNN

In this section, the CNN model is developed, and the features of the model are explained. Multi Input CNN derived from VGG16. A multi-input CNN classifier is trained, where features are learned from images activation values of fully connected layer of VGG16.

Network consists of two branches. The first branch has an ordinary CNN structure. Raw website entered screenshots and generated deep feature attributes. The second entry is already an image given as input to a trained network that takes the activation values it creates. The network is illustrated in Figure 5.6. When the architecture of the designed multi-input CNN model is examined; There is RESNET50 model. The part that receives the raw images as input in the design, the first layer is the image input layer. 224*224*3 size images are provided as input. In the network, ResNet 50 has 5 stages and each stage with a convolutional and identity block. Stage

1, the feature map convolutional layer with 2 strides which is followed by Batch and Relu. Convolution layers are the main layers and convolutional block has one extra layer to match the input and output dimension. Each stage has the same number of filters. There are filters for learning the different types of features in these layers. Each filter overwrites the input images and convolution is applied. Lastly, two non-residual blocks are added to the end.

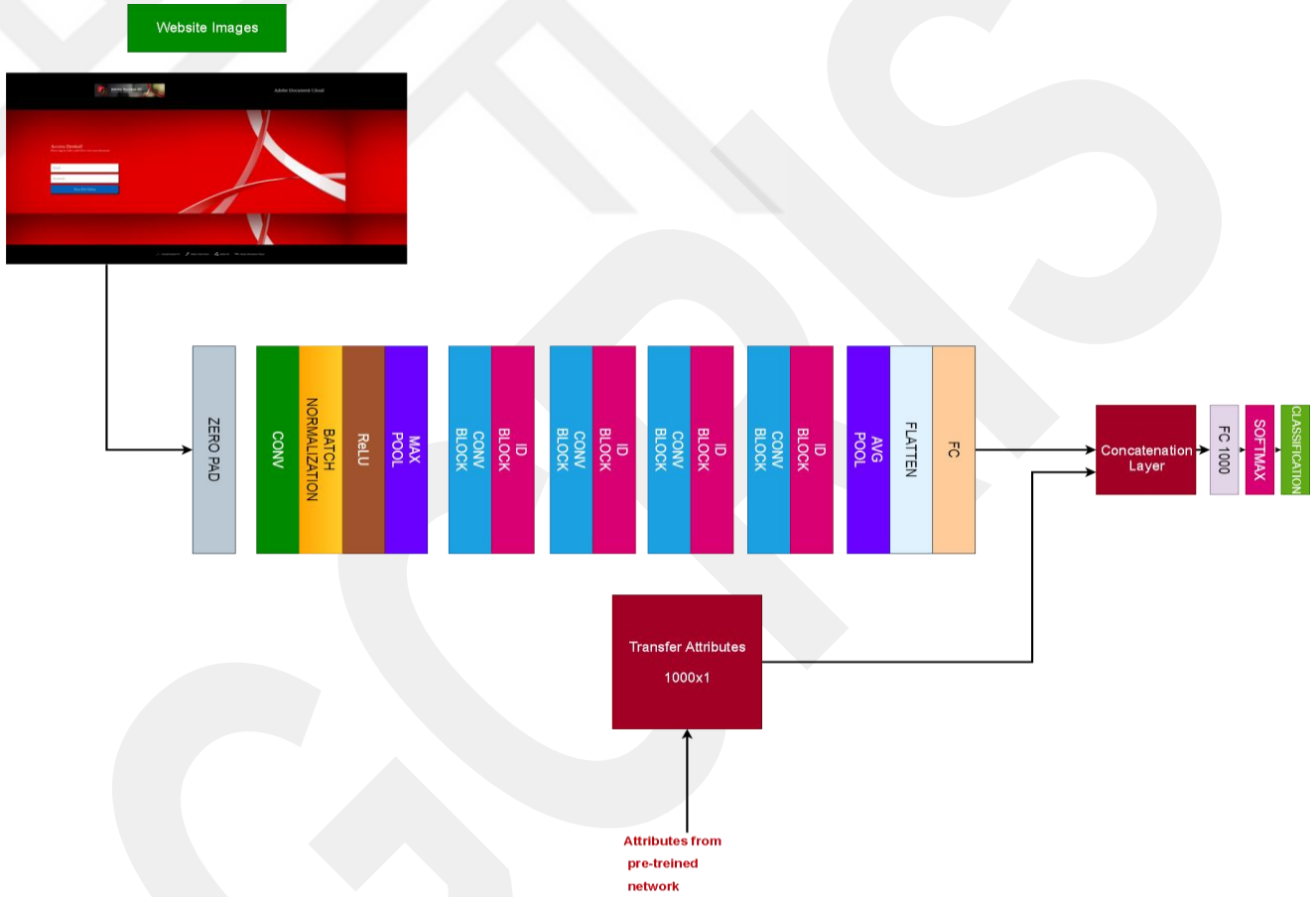

Figure 5.6: Multi-Input CNN Architecture

CHAPTER VI

EXPERIMENTS

6.1 EXPERIMENTAL SETUP

In this thesis, the experiments were compiled using the R2019a MATLAB program installed on a 64-bit Windows 10 operating system. Properties of the computer used; NVIDIA GeForce GTX950M has a 4 GB graphics card, Intel © i7-Core 2.7 GHz processor and 16 GB RAM capacity.

6.2 EXPERIMENTAL STUDIES

In this part, a study is carried out to obtain the results of AlexNet, VGG16 and multilayer CNN models separately. The same dataset was applied for all models and the results were compared. In order to carry out the experiments, models were created, algorithms were developed, and models were simulated by MATLAB.

The image dataset for AlexNet and all other models has been added to MATLAB as a script. As stated in the pseudocode, for all models, the dataset is defined as 70% train and 30% test data. Pseudocode of input datasets are shown in Figure 6.1 and random dataset example are shown in Figure 6.2.

```
Algorithm: Load Dataset
Initialize Dataset;
imds = imageDatastore();
[imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'randomized');
numTrainImages = numel(imdsTrain.Labels);
idx = randperm(numTrainImages,2);
while index = 1:2 do
    subplot(2,2,index);
```

Figure 6.1: Pseudocode of input dataset

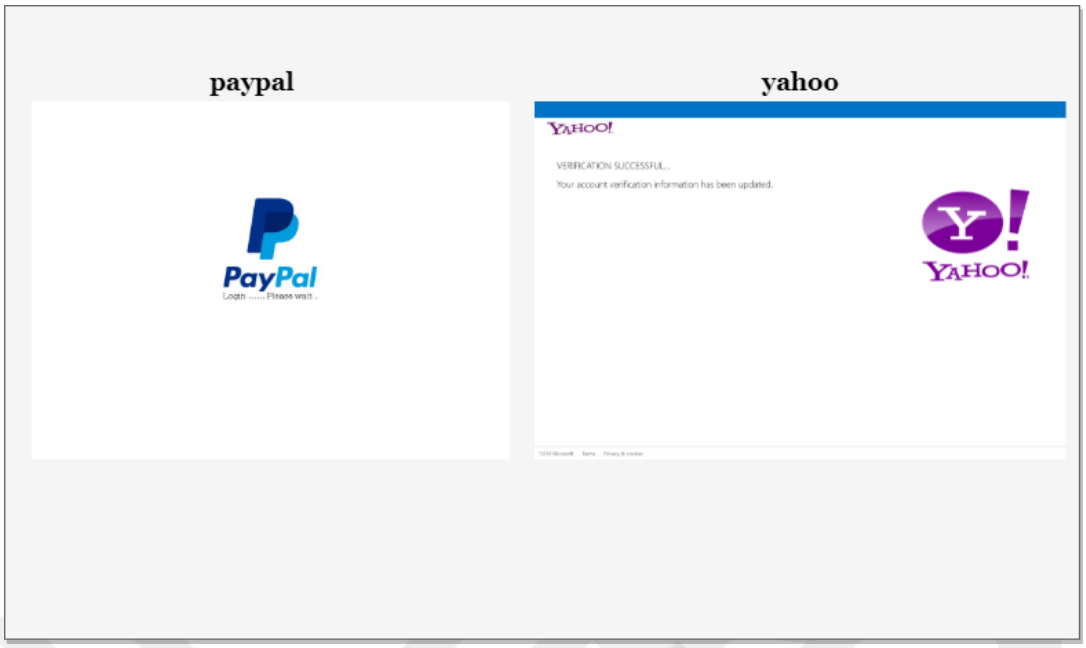

Figure 6.2: Random dataset example

6.2.1 AlexNet Experimental Studies

In this part, the AlexNet pre-trained model to the existing dataset is applied. Also, it explains how the result was obtained. How the algorithm is implemented. AlexNet network trained on the dataset. This syntax is equivalent to net = alexnet. The AlexNet network has been analyzed and shown in Table 6.1.

Table 6.1: AlexNet Network Analyze

#	Name	Type	Activation and Lernables
1	'data'	Image Input	227x227x3 images with 'zerocenter' normalization
2	'conv1'	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	'relu1'	ReLU	ReLU
4	'norm1'	Cross Channel Normalization	cross channel normalization with 5 channels per element
5	'pool1'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
6	'conv2'	Grouped Convolution	2 groups of 128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]
7	'relu2'	ReLU	ReLU
8	'norm2'	Cross Channel Normalization	cross channel normalization with 5 channels per element
9	'pool2'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
10	'conv3'	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
11	'relu3'	ReLU	ReLU

Table 6.1: Continuation

12	'conv4'	Grouped Convolution	2 groups of 192 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu4'	ReLU	ReLU
14	'conv5'	Grouped Convolution	2 groups of 128 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu5'	ReLU	ReLU
16	'pool5'	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
17	'fc6'	Fully Connected	4096 fully connected layer
18	'relu6'	ReLU	ReLU
19	'drop6'	Dropout	50% dropout
20	'fc7'	Fully Connected	4096 fully connected layer
21	'relu7'	ReLU	ReLU
22	'drop7'	Dropout	50% dropout
23	'fc8'	Fully Connected	1000 fully connected layer
24	'prob'	Softmax	softmax
25	'output'	Classification Output	crossentropyex with 'tench' and 999 other classes

In total, AlexNet network has five convolutional layers and three fully connected layers. AlexNet algorithm pseudocode shows in Figure 6.3.

```
Algorithm : AlexNet
Initialize AlexNet Networks;
AlexNet      load();
features,labels = getbatch(dataset);
model       createmodel();
analyzeNetwork(AlexNet);
inputSize = net.Layers().InputSize;
layerTransfer = net Layers();
numClasses = numel(categories(imdsTrain.Labels));
layerTransfer       set();
fullyConnectedLayer();
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter();
augimdsTrain = augmentedImageDatastore();
options = trainingOptions();
netTransfer = trainNetwork();
idx = randperm();
while index = 1:2 do
      subplot(2,2,index);
      readimage(imdsValidation,idx(index));
      imshow(index);
      YValidation = imdsValidation.Labels;
      Accuracy = mean(YPred == YValidation);
End
[cm,order] = confusionmat(YValidation, YPred);
tp = sum((YPred == 1) & (YValidation == 1))
fp = sum((YPred == 1) & (YValidation == 0))
tn = sum((YPred == 0) & (YValidation == 1))
fn = sum((YPred == 0) & (YValidation == 0))
sensitivity = tp/(tp + fn)  %TPR
specificity = tn/(tn + fp)  %TNR
precision = tp / (tp + fp)
FPR = fp/(tn+fp);
Accuracy = (TP+TN)./(TP+FP+TN+FN);
recall = tp / (tp + fn)
F1 = (2 * precision * recall) / (precision + recall);
```

Figure 6.3: Pseudocode of Future Extraction AlexNet

The dimensions of all inputs are adjusted to be 227*227*3. All layers are extracted, except the last three, from the pre-trained network. The layers are transferred to the new classification task by replacing the last three layers with a fully connected layer, a softmax layer, and a classification output layer.

After all the layers are set, the network train process is started. Training progress is shown in Figure 6.4.

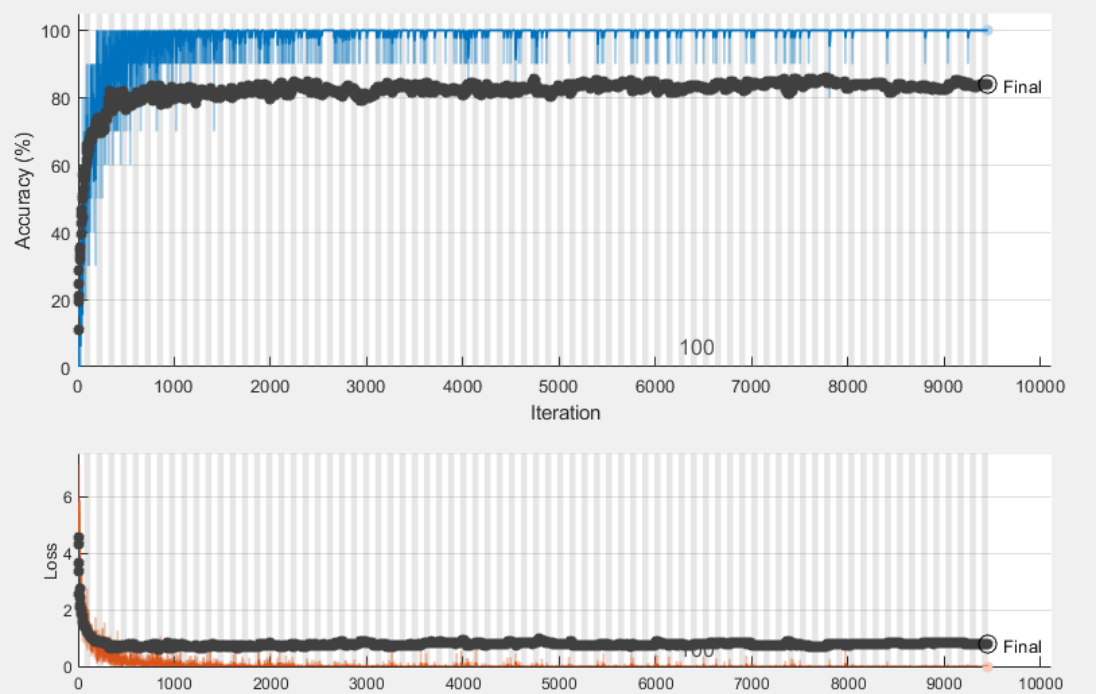

Figure 6.4: AlexNet Train Progress

After the AlexNet train network progress is complete, used the fine-tuned network to classify the validation images. Figure 6.5 and figure 6.6 show the result of the image classification process.

Figure 6.5: AlexNet Image Classification Result – 1

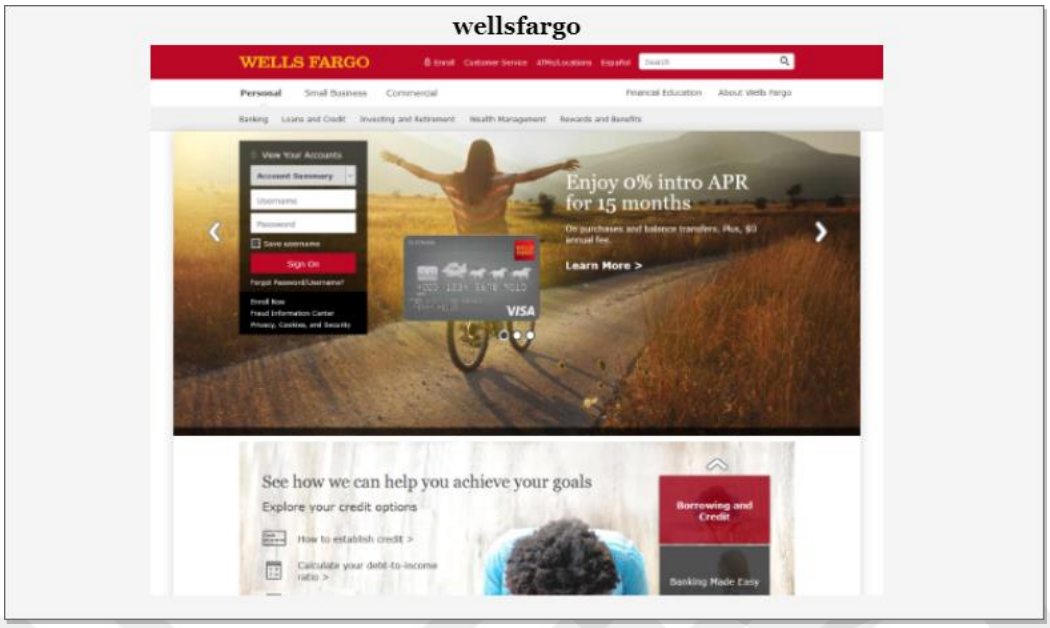

Figure 6.6: AlexNet Image Classification Result - 2

6.2.2 VGG16 Experimental Studies

In this part, applied the VGG16 pretrained model to the existing dataset and explain how the classification result is obtained. How the algorithm is implemented. Pretrained VGG16 CNN trained on the dataset. This syntax is equivalent to net = vgg16. The VGG16 network has been analyzed and shown in Table 6.2.

Table 6.2: VGG16 Network Analyze

#	Name	Type	Activation and Learnable
1	'input'	Image Input	224x224x3 images with 'zerocenter' normalization
2	'conv1_1'	Convolution	64 3x3x3 convolutions with stride [1 1] and padding [1 1 1 1]
3	'relu1_1'	ReLU	ReLU
4	'conv1_2'	Convolution	64 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]
5	'relu1_2'	ReLU	ReLU
6	'pool1'	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
7	'conv2_1'	Convolution	128 3x3x64 convolutions with stride [1 1] and padding [1 1 1 1]
8	'relu2_1'	ReLU	ReLU
9	'conv2_2'	Convolution	128 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]
10	'relu2_2'	ReLU	ReLU
11	'pool2'	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
12	'conv3_1'	Convolution	256 3x3x128 convolutions with stride [1 1] and padding [1 1 1 1]
13	'relu3_1'	ReLU	ReLU
14	'conv3_2'	Convolution	256 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
15	'relu3_2'	ReLU	ReLU
16	'conv3_3'	Convolution	256 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
17	'relu3_3'	ReLU	ReLU
18	'pool3'	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
19	'conv4_1'	Convolution	512 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
20	'relu4_1'	ReLU	ReLU
21	'conv4_2'	Convolution	512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]
22	'relu4_2'	ReLU	ReLU
23	'conv4_3'	Convolution	512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]
24	'relu4_3'	ReLU	ReLU
25	'pool4'	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
26	'conv5_1'	Convolution	512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]
27	'relu5_1'	ReLU	ReLU
28	'conv5_2'	Convolution	512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]
29	'relu5_2'	ReLU	ReLU
30	'conv5_3'	Convolution	512 3x3x512 convolutions with stride [1 1] and padding [1 1 1 1]

Table 6.2: Continuation

31	'relu5_3'	ReLU	ReLU
32	'pool5'	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
33	'fc6'	Fully Connected	4096 fully connected layer
34	'relu6'	ReLU	ReLU
35	'drop6'	Dropout	50% dropout
36	'fc7'	Fully Connected	4096 fully connected layer
37	'relu7'	ReLU	ReLU
38	'drop7'	Dropout	50% dropout
39	'fc8'	Fully Connected	1000 fully connected layer
40	'prob'	Softmax	softmax
41	'output'	Classification Output	crossentropyex with 'tench' and 999 other classes

In total, VGG16 CNN has 41 layers. 16 layers learnable weight, 13 layers convolutional and 3 fully connected layers. VGG16 experimental future extraction pseudocode shows in Figure 6.7.

Algorithm : VGG16

```
Initialize VGG16 Networks;
AlexNet      load();
features,labels = getbatch(dataset);
model       createmodel();
analyzeNetwork(VGG16);
inputSize = net.Layers().InputSize;
layerTransfer = net Layers();
numClasses = numel(categories(imdsTrain.Labels));
layerTransfer        set();
fullyConnectedLayer();
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter();
augimdsTrain = augmentedImageDatastore();
options = trainingOptions();
netTransfer = trainNetwork();
idx = randperm();
while index = 1:2 do
      subplot(2,2,index);
      readimage(imdsValidation,idx(index));
      imshow(index);
      YValidation = imdsValidation.Labels;
      Accuracy = mean(YPred == YValidation);
End
[cm,order] = confusionmat(YValidation, YPred);
tp = sum((YPred == 1) & (YValidation == 1))
fp = sum((YPred == 1) & (YValidation == 0))
tn = sum((YPred == 0) & (YValidation == 1))
fn = sum((YPred == 0) & (YValidation == 0))
sensitivity = tp/(tp + fn)  %TPR
specificity = tn/(tn + fp)  %TNR
precision = tp / (tp + fp)
FPR = fp/(tn+fp);
Accuracy = (TP+TN)./(TP+FP+TN+FN);
recall = tp / (tp + fn)
F1 = (2 * precision * recall) / (precision + recall);
```

Figure 6.7: Pseudocode of VGG16 Future Extraction

The dimensions of all inputs are adjusted to be 224*224*3. All layers are extracted, except the last three, from the pre-trained network. The layers are transferred to the new classification task by replacing the last three layers with a fully connected layer, a softmax layer, and a classification output layer.

After all the layers are set, the network train process is started. Training progress is shown in Figure 6.8.

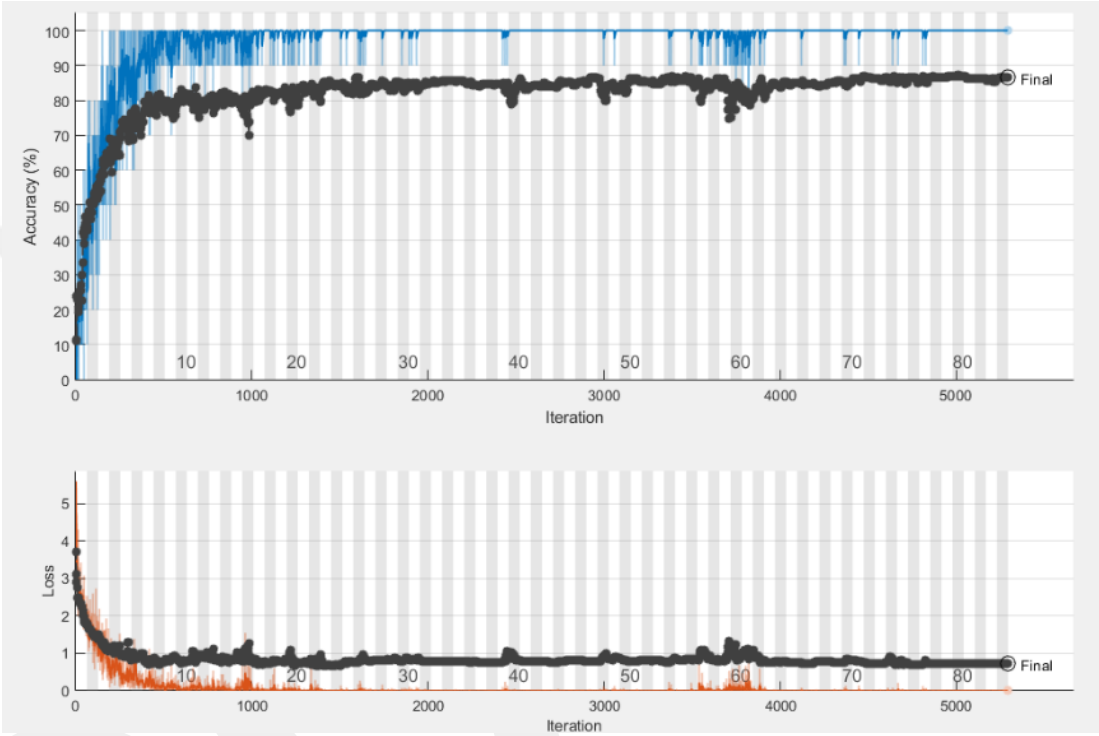

Figure 6.8: VGG16 Train Process

After the VGG16 train network progress is complete, used the fine-tuned network to classify the validation images. Figure 6.9 and Figure 6.10 show the result of the image classification process.

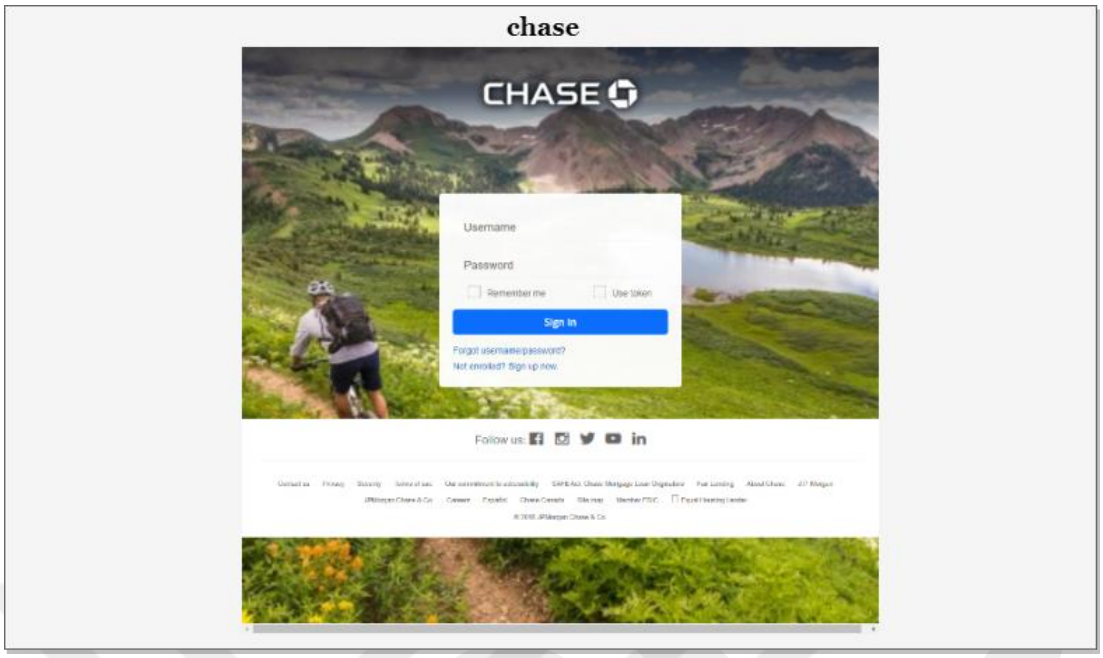

Figure 6.9: VGG16 Image Classification Result - 1

Figure 6.10: VGG16 Image Classification Result - 2

6.2.3 Multi-Input CNN Model Experimental Studies

In this part, applied the Multi-Input CNN pre-trained model to the existing dataset and explain how the classification result is obtained. How the algorithm is implemented. Multi-Input CNN network trained on the dataset.

The multi-input CNN model takes the input images data from the VGG16 pre-trained network FC-8 image classifies. Image classification processes are carried out

with the ResNet50 pre-trained model. First, input image is detected by convolution layer. After convolution layer is the Batch and ReLu layers. Batch layers by previous layers setting the calculated activation values and used to normalize. ReLu layers, eliminate the effect of dark areas and provide a nonlinear used to obtain a model. Pooling layers, decrease the input sizes to reduce the complexity of calculation. This is done by applying a mathematical MAX operation on the values corresponding to the filter.

In pooling operations, 3x3 filters are used. Fully connected layers are the last layers of the CNN model. These layers are the standard layers of neural networks. Class values for a specific input are calculated, in fully connected layers. The activation values of each layer correspond to a different abstraction.

The second branch of the model takes the features obtained by transfer learning. Machine learning models, which mostly need to be rebuilt when properties and data change, were created to work alone. Additionally, previously acquired knowledge in machine learning is often similar can be applied to tasks. Usually requires a lot of effort transfer learning model instead of rebuilding models, reuse the acquired knowledge and similarly model significantly reduce development time and isolated. It aims to develop the model efficiency of the learning model. Within the scope of the study, VGG16 network was used to extract attributes. By taking the activation values created in the Fully Connected-8 layer, it is added to the multi-input deep learning network as second input.

Fully Connected-8 layer generates a vector of 1000 as output. This vector is for every image as an input to the network that will receive and perform the actual classification. Multi-input CNN experimental future extraction pseudocode shows in Figure 6.11, pseudocode of Classification Layer shows in Figure 6.12 and training process is shown in Figure 6.13.

Algorithm: Multi-Input CNN

```
Input VGG16 Transfer Attributes;
Initialize RESNET50 Networks;
RESNET50      load();
Features,labels = getbatch(dataset);
model      createmodel();
stage – 1;
imageInputLayer();
convolution2dLayer();
batchNormalizationLayer();
reluLayer();
maxpooling2d();
stage – 2;
convolutionalUnit();
additionLayer();
reluLayer();
convolutionalUnit();
additionLayer();
reluLayer();
stage – 3;
convolutionalUnit();
additionLayer();
reluLayer();
convolutionalUnit();
additionLayer();
reluLayer();
stage – 4;
convolutionalUnit();
additionLayer();
reluLayer();
convolutionalUnit();
additionLayer();
reluLayer();
stage – 5;
averagePooling2dLayer();
fullyConnectedLayer();
softmaxLayer();
classificationLayer();
```

Figure 6.11: Pseudocode of Transfer Layer Multi-Input Model

```
concatenation;
concatenationLayer();
fullyConnectedLayer();
softmaxLayer();
while index 1:2 do
        subplot(2,2,index);
        readimage(imdsValidation,idx(index));
        imshow(index);
        YValidation = imdsValidation.Labels;
End
[cm,order] = confusionmat(YValidation, YPred);
tp = sum((YPred == 1) & (YValidation == 1))
fp = sum((YPred == 1) & (YValidation == 0))
tn = sum((YPred == 0) & (YValidation == 1))
fn = sum((YPred == 0) & (YValidation == 0))
sensitivity = tp/(tp + fn)  %TPR
specificity = tn/(tn + fp)  %TNR
precision = tp / (tp + fp)
FPR = fp/(tn+fp);
Accuracy = (TP+TN)./(TP+FP+TN+FN);
recall = tp / (tp + fn)
F1 = (2 * precision * recall) / (precision + recall);
```

Figure 6.12: Pseudocode of Classification Layer

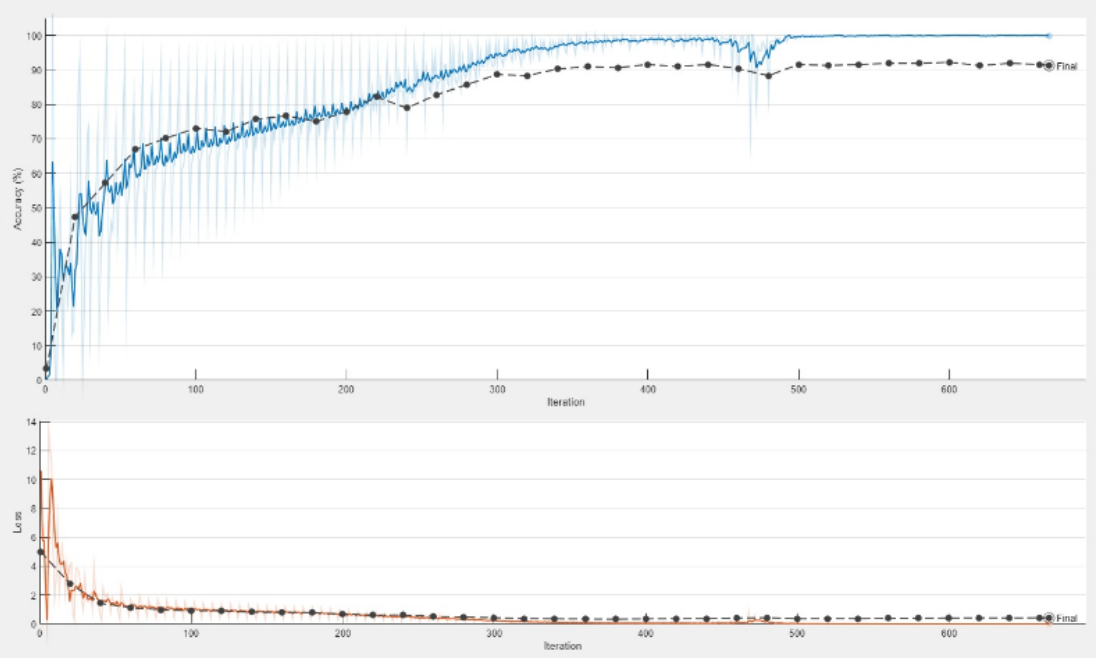

Figure 6.13: Multi-Input CNN Train Process

In the last parts of the network, attributes obtained by transfer learning and activation values from the other branch, are combined in the concatenation layer and transferred to the next fully connected layer. The last layers are softmax and are the

classification layers. Softmax layer converts the outputs to probability values and selects the maximum probability label as the output of the classification layer.

After the multi-input CNN train network progress is complete, used the fine-tuned network to classify the validation images. Figure 6.14 and figure 6.15 show the result of the image classification process.

Figure 6.14: Multi-Input CNN Image Classification Result - 1

Figure 6.15: Multi-Input CNN Image Classification Result - 2

6.3 EXPERIMENTAL RESULT

The parameters used in training the models are shown in Table 6.3. 1996 screenshots corresponding to 70% of the total 2852 screenshots that make up the data set were used for training the classifier, and the remaining 856 screenshots were used for sample testing. The comparison results obtained by using feature selection methods are shown in Table 6.4.

Table 6.3: CNN Training Parameters

Training Options	Training Properties
Optimizer	Sgdm
Initialize Rate	1e-4
Min Batch Size	60
Epoch Count	1000
Verbose	1
Execution Environment	auto
Learn Rate Drop Factor	0,2
Learn Rate Drop Period	5
Learn Rate Schedule	piecewise
Output Network	Last-iteration

Table 6.4: Rates of Feature Selection Methods

Architecture	Image Size	Class Set	Validation accuracy (%)	TPR (%)	FNR (%)	F1-Score (%)
AlexNet	227*227	Data Set	83.94	81.3	18.Tem	89.6
VGG 16	224*224	Data Set	86.50	86.4	13.May	92.8
Multi Input CNN	227*227	Data Set	91.23	91.2	8.Tem	95.4

The results we obtained when we compared the overall accuracy results of VGG16, AlexNet and multi-input CNN transfer learning models; we observed that the multi-input CNN model was better at predicting phishing websites than other models. It is known that the emergence of this result is due to the use of the ResNet50 transfer learning model as a classifier. The ILSVRC results, edited by the ImageNet team since 2010, are shown in the Figure 6.16 [53].

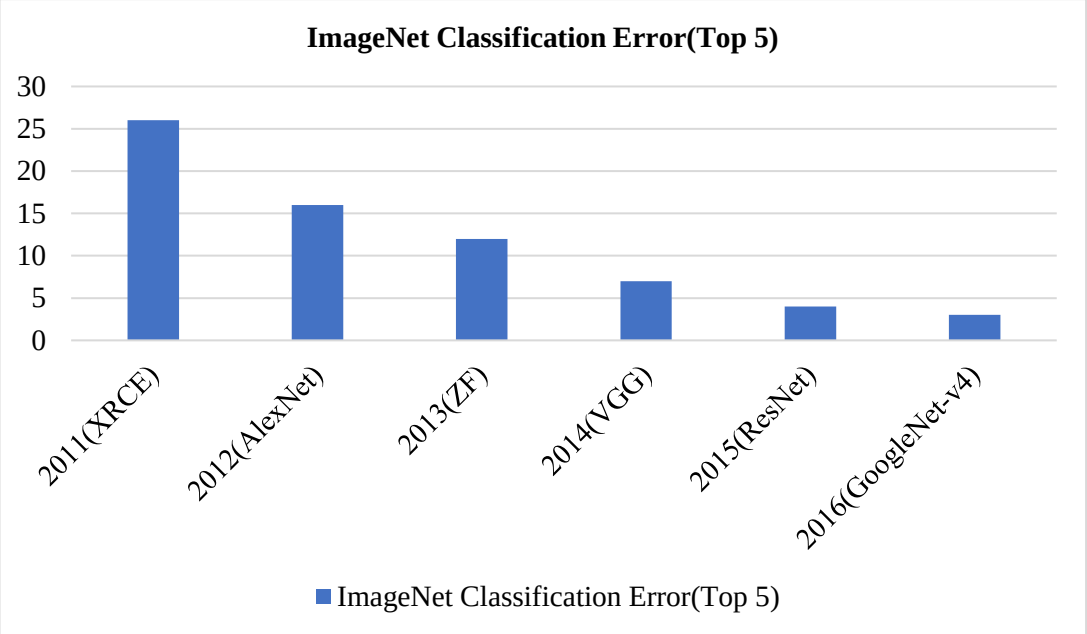

Figure 6.16: ImageNet Classification Error [53]

It is seen that these results and the results we obtained overlap with each other. The best accuracy rate was given by the multi-input CNN model (91.23%), which we created using the ResNet50 model, while VGG16 (86.50%) and AlexNet (83.94%) gave the lowest rates. In addition, confusion matrix algorithms were used to compare the obtained results. The results of some of these algorithms are shown in the Table 6.4. According to these results, the multi-Input CNN model (91.2%) gave the highest TPR rate, the multi-input CNN model (8.7%) gave the lowest FNR rate, and the multi-input CNN model (95.4%) gave the highest F1-Score rate.

6.4 COMPARISON WITH SIMILAR STUDIES

In this section, the performance results obtained from similar studies detecting phishing websites are compared with the performance results of our study.

Kaytan and Hanbay [55] used a model they developed to detect phishing websites. They named this model the ELM classifier. According to web sites features, sample data of websites were collected. 90% of this data is separated as train and 10% as test data. Prediction was made with the ELM classifier. According to the result obtained, the accuracy rate was 95.93%. When we compare our study with this study, similar prediction and performance tests were carried out in both studies. Accuracy rates of both studies were high.

In another similar study, J. Mao et al. [56] based on auto layout they did research on phishing detection techniques using machine learning techniques. They propose a learning-based aggregation analysis mechanism to decide page layout similarity, which is used to detect phishing pages. They compared the solution method with popular machine learning algorithms and achieved scores that were not bad at all. The solution method presented in this study, like the other study, was very similar to the result we wanted to achieve. In this study, a prediction rate of over 90% was obtained.

Comparison of methods are shown in Table 6.5.

Table 6.5: Comparison of Method Results

Method	Number Of Dataset	Accuracy Rate(%)
Multi-Input CNN	2782	91.23
ELM Classifier	11055	95.93
Similarity Of Page Layouts And Detect Phishing Pages	2923	<93

CHAPTER VII

CONCLUSIONS AND FUTURE WORKS

Within the scope of the study, an end-to-end deep learning network was designed in which the attributes learned with a CNN trained from scratch and attributes obtained by transfer learning are combined. Within the scope of the classification of phishing websites, it has been shown that better results can be obtained by combining the features learned from scratch and the features obtained by transfer learning. As detailed in the experimental result, multi-input CNN achieved higher rates on detection of phishing websites than AlexNet and VGG16 models. The best result was given by the multi-input CNN model (91.23%), which we created using the ResNet50 model, while VGG16 (86.50%) and AlexNet (83.94%) gave the lowest score. Moreover, the multi-Input CNN model (91.2%) gave the highest TPR rate, the multi-input CNN model (8.7%) gave the lowest FNR rate, and the multi-input CNN model (95.4%) gave the highest F1-Score rate.

Word2vec method can also be used as an additional input within the scope of future studies. Word2vec models proposed by Mikolov et al. [4] are often used to learn word placement. While the traditional word model represents every word in the corpus with large sparse vectors, the word embedding approach adopts the true valued dense vector representation where each vector represents a continuous vector space of the word. The position of each word in the learned vector is considered embedding. Word2Vec has two big advantages. First, contextual similarities between words could be inferred, and another is that semantic relationships could be extracted from learned vectors. In this context, in the future study, a 3-input network will be developed by using the Ocr and word2vec method together, tokenizing the texts to be obtained from the website image with ocr and transforming them into vectors with word2vec.

REFERENCES

[1] M. Jakobsson and S. Myers, *Phishing and countermeasures: Understanding the increasing problem of identity theft. Introduction to Phishing*. Hoboken, N.J.: Wiley-Interscience, 2007, pp. 1-2.

[2] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[3] H. Qassim, A. Verma and D. Feinzimer, *Compressed residual-VGG16 CNN model for big data places image recognition.* In 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), pp. 169-175, 2018.

[4] F. Zhang, "*A hybrid structured deep neural network with Word2Vec for construction accident causes classification*", International Journal of Construction Management, pp. 1-21, 2019. Available: 10.1080/15623599.2019.1683692.

[5] A. Jain and B. Gupta, "*Phishing Detection: Analysis of Visual Similarity Based Approaches*", Security and Communication Networks, vol. 2017, pp. 1-20, 2017. Available: 10.1155/2017/5421046.

[6] A. P. E. Rosiello, E. Kirda, C. Kruegel, and F. Ferrandi. *A layout-similarity-based approach for detecting phishing pages*. In SecureComm 2007, pp. 454–463, 2007.

[7] P. Bohunsky and W. Gatterbauer, *Visual structure-based web page clustering and retrieval*, In Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp. 1067–1068, New York, NY, USA, 2010. ACM.

[8] F. Dalgıç, A. Bozkır and M. Aydos, "Phish-IRIS: A New Approach for Vision Based Brand Prediction of Phishing Web Pages via Compact Visual Descriptors", *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2018. Available: 10.1109/ismsit.2018.8567299.

[9] M. Jakobsson and S. Myers, *Phishing and countermeasures: Understanding the increasing problem of identity theft. Introduction to Phishing*. Hoboken, N.J.: Wiley-Interscience, 2007, pp. 1-2.

[10] "Phishing activity trends report", *Docs.apwg.org*, 2020. [Online]. Available: https://docs.apwg.org/reports/apwg_trends_report_q4_2020.pdf. [Accessed: 15- Jan- 2021].

[11] R. Gowtham and I. Krishnamurthi, "A comprehensive and efficacious architecture for detecting phishing webpages", Computers & Security, 40, pp.23-37, 2014.

[12] Y. Li, L. Yang and J. Ding, "A minimum enclosing ball-based support vector machine approach for detection of phishing websites", Optik, 127(1), pp.345- 351, 2016.

[13] H. Nguyen and D. Nguyen, "Machine Learning Based Phishing Web Sites Detection", AETA 2015: *Recent Advances in Electrical Engineering and Related Sciences*, pp. 123-131, 2016. Available: 10.1007/978-3-319-27247-4_11.

[14] H. Kazemian and S. Ahmed, "Comparisons of machine learning techniques for detecting malicious webpages", *Expert Systems with Applications*, vol. 42, no. 3, pp. 1166-1177, 2015. Available: 10.1016/j.eswa.2014.08.046.

[15] R. Mohammad, F. Thabtah and L. McCluskey, "Predicting phishing websites based on self-structuring neural network", *Neural Computing and Applications*, vol. 25, no. 2, pp. 443-458, 2014. Available: 10.1007/s00521-013-1490-z.

[16] N. Abdelhamid, A. Ayesh and F. Thabtah, "Phishing detection based Associative Classification data mining", *Expert Systems with Applications*, vol. 41, no. 13, pp. 5948-5959, 2014. Available: 10.1016/j.eswa.2014.03.019.

[17] N. Abdelhamid, A. Ayesh, F. Thabtah, "*Associative classification mining for website phishing classification*". Proceedings of the International Conference on Artificial Intelligence, Las Vegas, USA, 22-25 July 2013.

[18] M. Moghimi, A. Y. Varjani, New rule-based phishing detection method. Expert Systems with Applications, 53, 231-242, 2016. doi:10.1016/j.eswa.2016.01.028.

[19] C. Ludl, S. Mcallister, E. Kirda, C. Kruegel. On the effectiveness of techniques to detect phishing sites. In DIMVA '07: Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, Springer, Berlin, Heidelberg, 4579, 20-39, 2007. doi:10.1007/978-3- 540-73614-1_2.

[20] Z. Zhao, Y. Zhang, Z. Comert, and Y. Deng, "Computer-aided diagnosis system of fetal hypoxia incorporating recurrence plot with convolutional neural network", Front. Physiol., vol. 10, pp. 255, 2019. DOI: 10.3389/fphys.2019.00255.

(PDF) Subclass Separation of White Blood Cell Images Using Convolutional Neural Network Models. Available from:

https://www.researchgate.net/publication/336389582_Subclass_Separation_of_White_Blood_Cell_Images_Using_Convolutional_Neural_Network_Models [accessed Apr 11 2021].

[21] A. Krizhevsky, I. Sutskever, and G.E. Hinton, 25th International Conference on Neural Information Processing Systems. ImageNet Classification with Deep Convolutional, pp. 1097–1105. Lake Tahoe, Nevada: NIPS'12 Proceedings, 2012.

[22] K.Simonyan, A. Zisserman, *Very deep convolutional networks for large-scale image recognition*. arXiv. 2014 arXiv.1409.1556

[23]"VGG16 - Convolutional Network for Classification and Detection", Neurohive.io, 2021. [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/. [Accessed: 24- Apr- 2021].

[24] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, 2010.

[25] B. Koçer, "Transfer Öğrenmede Yeni Yaklaşımlar", Selçuk Üniversitesi, Doktora Tezi, 2012.

[26] S. Albawi, T. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network", 2017 International Conference on Engineering and Technology (ICET), 2017. Available: 10.1109/icengtechnol.2017.8308186 [Accessed 13 May 2021].

[27] K. Shea, R. Nash, "*An introduction to convolutional neural networks*", ArXiv e-prints. Retrieved from https://www.researchgate.net/publication/285164623, 2015.

[28] X. Zhang, Y. Wang, N. Zhang, D. Xu and B. Chen, "Research on Scene Classification Method of High-Resolution Remote Sensing Images Based on RFPNet", Applied Sciences, vol. 9, no. 10, p. 2028, 2019. Available: 10.3390/app9102028.

[29] D. Cireşan, U. Meier, J. Masci, L. Gambardella and J. Schmidhuber, "Flexible, High Performance Convolutional Neural Networks for Image Classification", IDSIA, USI and SUPSI Galleria 2, 6928 Manno-Lugano, Switzerland, 2011.

[30] L. Hui-bin, W. Fei, C. Qiang and P. Yong, "Recognition of individual object in focus people group based on deep learning", 2016 International Conference on Audio, Language and Image Processing (ICALIP), 2016. Available: 10.1109/icalip.2016.7846607 [Accessed 4 June 2021].

[31] G. Altan, "DeepGraphNet: Grafiklerin Sınıflandırılmasında Derin Öğrenme Modelleri", European Journal of Science and Technology, pp. 319-327, 2019. Available: 10.31590/ejosat.638256.

[32] A. Sharma V, "Understanding Activation Functions in Neural Networks", Medium, 2017. [Online]. Available: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0. [Accessed: 30- Jun- 2017].

[33] M. Castelluccio, G. Poggi, C. Sansone, & L. Verdoliva, "*Land use classification in remote sensing images by convolutional neural networks*", arXiv preprint arXiv:1508.00092, 2015

[34] E. Karakullukçu, "Yanık Görüntülerinin Çok Değişkenli İstatistiksel Yöntemler Ve Derin Öğrenme Yaklaşimlari İle Analizi", 2020. [Accessed 27 June 2021].

[35] A. Adler, M. Elad, and M. Zibulevsky, "*Compressed Learning for Image Classification: A Deep Neural Network Approach*", Handbook of Numerical Analysis, vol. 19, pp. 3 - 17, 2018. Available: https://linkinghub.elsevier.com/retrieve/pii/S1570865918300024. [Accessed 27 June 2021].

[36] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "*Dropout: a simple way to prevent neural networks from overfitting*." Journal of machine learning research, 15(1), pp. 1929- 1958, 2014.

[37] Y. LeCun and Y. Bengio, "*Convolutional networks for images, speech, and time series."* The handbook of brain theory and neural networks, 3361(10), 1995.

[38] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "*Dropout: a simple way to prevent neural networks from overfitting*." Journal of machine learning research, 15(1), pp. 1929- 1958, 2014.

[39] D.C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, *"Flexible, high performance convolutional neural networks for image classification."* In IJCAI Proceedings-International Joint Conference on Artificial Intelligence, vol. 22, no. 1, p. 1237, 2011.

[40] Y. Tang, Deep learning using linear support vector machines. arXiv preprint arXiv:1306.0239, 2013.

[41] H. Li, Z. Lin, X. Shen, J. Brandt and G. Hua, "*A convolutional neural network cascade for face detection*", 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5325-5334, 2015. Available: 10.1109/cvpr.2015.7299170.

[42] P. Yi, Y. Guan, F. Zou, Y. Yao, W. Wang, T. Zhu, Web phishing detection using a deep learning framework. Wireless Communications and Mobile Computing, 9, 1-9, 2018. doi:10.1155/2018/4678746.

[43] Y. Pan, X. Ding, Anomaly based web phishing page detection. In ACSAC ’06: Proceedings of the 22nd Annual Computer Security Applications Conference, IEEE Computer Society. 1, 381-392, 2006. doi:10.101 9/ACSAC.2006.13.

[44] S. Chandler, Google Registers Record Two Million Phishing Websites In 2020. Retrieved 3 November 2021, from https://www.forbes.com/sites/simonchandler/2020/11/25/google-registers-record-two-million-phishing-websites-in-2020/?sh=4188f8101662.

[45] Activation Function - AI Wiki. (2021). Retrieved 4 November 2021, from https://docs.paperspace.com/machine-learning/wiki/activation-function

[46] AlexNet, VGGNet, Inception ve ResNet Nedir?. (2021). Retrieved 7 November 2021, from https://frightera.medium.com/alexnet-vggnet-inception-ve-resnet-nedir-bddc7482918b

[47] P. Ballester, and R. M. Araujo, On the performance of GoogLeNet and AlexNet applied to sketches. In Thirtieth AAAI Conference on Artificial Intelligence, 2016.

[48] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition. Microsoft Research, 2015

[49] MLK - Machine Learning Knowledge. 2021. Keras Implementation of ResNet-50 (Residual Networks) Architecture from Scratch - MLK - Machine Learning Knowledge. [online] Available at: <https://machinelearningknowledge.ai/keras-implementation-of-resnet-50-architecture-from-scratch/> [Accessed 12 December 2021].

[50] S. Poudel, Y. Kim, D. Vo, and S. Lee, Colorectal Disease Classification Using Efficiently Scaled Dilation in Convolutional Neural Network. IEEE, 2020, Access, 8, pp.99227-99238.

[51] Cs.colby.edu, 2021. [Online]. Available: https://cs.colby.edu/courses/F19/cs343/lectures/lecture11/Lecture11Slides.pdf. [Accessed: 14- Dec- 2021].

[52]"CS231n Convolutional Neural Networks for Visual Recognition", Cs231n.github.io, 2021. [Online]. Available: https://cs231n.github.io/convolutional-networks/. [Accessed: 14- Dec- 2021].

[53] videantis - processors for deep learning, computer vision and video coding. 2021. *Deep learning in five and a half minutes - videantis - processors for deep learning, computer vision and video coding.* [online] Available at: <http://www.videantis.com/deep-learning-in-five-and-a-half-minutes.html> [Accessed 18 December 2021].

[54] T. Ergin, "Convolutional Neural Network (ConvNet yada CNN) nedir, nasıl çalışır?", Medium, 2021. [Online]. Available: https://medium.com/@tuncerergin/convolutional-neural-network-convnet-yada-cnn-nedir-nasil-calisir-97a0f5d34cad.

[55] M. KAYTAN and D. HANBAY, "Effective Classification of Phishing Web Pages Based on New Rules by Using Extreme Learning Machines", vol. 2, no. 2548-1304, pp. 15-36, 2017.

[56] J. Mao et al., "Phishing page detection via learning classifiers from page layout feature", *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, 2019. Available: 10.1186/s13638-019-1361-0.