T.C.
İSTANBUL UNIVERSITY-CERRAHPASA
INSTITUTE OF GRADUATE STUDIES

M.Sc. THESIS

FACE RECOGNITION VIA UNSUPERVISED DEEP LEARNING AND AUTO-ENCODERS

Ahmet ŞİMŞEK

SUPERVISOR
Assist. Prof. Dr. Pelin GÖRGEL

Department of Computer Engineering

Computer Engineering Programme

ISTANBUL-2018

This study was accepted on 14/12/2018 as a M. Sc. thesis in Department of Computer Engineering, Computer Engineering Programme by the following Committee.

Examining Committee Members

Dr. Öğr. Üyesi Pelin GÖRGEL (Supervisor)
Istanbul University -Cerrahpasa
Engineering Faculty

[illegible]

Prof. Dr. Ahmet SERTBAŞ
Istanbul University -Cerrahpasa
Engineering Faculty

[illegible]

Dr. Öğr. Üyesi Yusuf YASLAN
İstanbul Technical University
Engineering Faculty

[illegible]

FOREWORD

I would like to sincerely thank Assist. Prof. Dr. Pelin GÖRGEL, a very valuable supervisor, for her support and assistance during my thesis studies.

I am grateful to the University of Istanbul for supporting my colleagues and practicing my work during this study.

December 2018 Ahmet ŞİMŞEK

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	**Explanation**
W	: Weights
b	: Bias
λ	: Lambda
θ	: Theta
β	: Beta
ρ	: Sparsity parameter

Abbreviation	**Explanation**
PCA	: Principal Component Analysis.
LDA	: Linear Discriminant Analysis.
DCT	: Discrete Cosine Transform.
AI	: Artificial Intelligence.
LLE	: Local Linear Embedding.
LBP	: Local Binary Patterns.
LLE	: Local Linear Embedding.
HOG	: Histogram of Oriented Gradients.
SIFT	: Scale Invariant Feature Transform.
SVM	: Support Vector Machine.
NN	: Nearest Neighbor.
NS	: Nearest Subspace.
CNN	: Convolutional Neural Networks.
VJ	: Viola–Jones Detector.
ICA	: Independent Component Analysis.
SURF	: Speeded Up Robust Features.
DBN	: Deep Belief Network.

ÖZET

YÜKSEK LİSANS TEZİ

GÖZETİMSİZ DERİN ÖĞRENME VE OTOMATİK KODLAYICILAR İLE YÜZ TANIMA

Ahmet ŞİMŞEK

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Dr. Öğr. Üyesi Pelin GÖRGEL

Yüz tanıma; yüzün poz açısı, ifadesi, aydınlatma, yüzün kapanması ve arka plan farklılıklardan kaynaklanan zorluklar nedeniyle çokça incelenen güncel bir konudur. Son zamanlarda, derin öğrenme yöntemleri, görüntünün tanımlanması ve görüntü tanıma alanlarında dikkate değer sonuçlar elde etmiştir.

Otomatik Kodlayıcı (Autoencoder), girişlerden faydalanarak bir kimlik fonksiyonunun yaklaşıklılığını öğrenmek, başka deyişle görüntünün faydalı tanımlamasını otomatik öznitelik çıkarımı ile öğrenmek için kullanılan gözetimsiz bir yapay ağdır. Bu tezde, yüz tanıma işlemi için Derin Yığılı Aralıklı Gürültü Temizleme Otomatik Kodlayıcıları (Deep Stacked Denoising Sparse Autoencoder-DSDSA) metodu önerilmiştir. Sınıflandırma aşamasında, en etkin sınıflandırıcılardan ikisi olan Destek Vektör Makineleri (SVM) ve Softmax sınıflandırıcısı kullanılmıştır. ORL, Yale, Caltech ve PubFig83'in bir alt kümesi gibi bilinen yüz görüntüleri veritabanları ile elde edilen deneysel sonuçlar, önerilen sistemin tatmin edici bir performans sergilediğini ve karşılaştırılabilir bir doğruluk oranı sağladığını göstermektedir.

Kasım 2018, 70 sayfa.

Anahtar kelimeler: Yüz Tanımı, Makine Öğrenmesi, Gözetimsiz Öğrenme, Derin Öğrenme, Sinir Ağı, Otomatik Kodlayıcılar

SUMMARY

FACE RECOGNITION VIA UNSUPERVISED DEEP LEARNING AND AUTOENCODERS

M.Sc. THESIS

Ahmet ŞİMŞEK

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Department of Computer Engineering

Supervisor : Assist. Prof. Dr. Pelin GÖRGEL

Face recognition is a hot topic under investigation due many challenges of variation includes the difference in poses, illumination, expression, occlusion and scenes. Recently, deep learning methods achieved remarkable results in image representation and recognition fields.

Autoencoder is an unsupervised artificial network used to learn an approximation of an identity function in other words to learn good representations of image, utilizing inputs by extracting features automatically. In this thesis, Deep Stacked Denoising Sparse Autoencoder (DSDSA) method is proposed for face recognition task. In classification phase, two of the most powerful classifiers were used, namely multi-class Support Vector Machines (SVM) and Softmax classifier. Experimental results on known face image databases, includes ORL, Yale, Caltech and a subset of PubFig83 show that the proposed system yields promising performance and achieves comparable accuracy.

November 2018, 70 pages.

Keywords: Face Recognition, Machine Learning, Unsupervised Learning, Deep Learning, Neural Network, Autoencoders

1. INTRODUCTION

Machine learning, where the computer is taught to simulate human learning, is considered a revolution in the world of technology. Many outstanding achievements have been made thanks to this strategy. Machine learning developed between supervised learning where target is clear to unsupervised learning which has taken more research and development recently where no labels are given to the learning algorithm, thus opened the way to benefit from large data without labels which are available widely and can be obtained much easier than the labeled data, also there is a semi-supervised method that take advantage of both previous methods. Deep learning has benefited from this revolution by moving from manual to automatic feature learning which learn data representations from raw data automatically, thus making us closer to reaching the goal, 'building the computer brain'.

Face recognition has received considerable attention in research in past decades, because of its wide using in various fields. Recently, for instance, it has been particularly competitive in its use in smart phone applications. Despite the good success achieved, face recognition is still under investigation due to the variation of constraints and determinants of which the system has been adapted, this includes the difference in poses, illumination, expression, occlusion and scene. Thus, we may see that some face recognition systems successfully applied to some databases with specific constraints, but do not achieve the same result when examined under different conditions.

Deep learning is a sophisticated area of machine learning, the use of deep neural networks led to promising results in image representation, especially in the nonlinear features extraction of high levels. Deep learning methods strive to learn complex functions mapping the input to the output directly from data. By learning a hierarchy where the upper levels features are constructed from the output of lower-level features [1].

Autoencoder is a special artificial neural network which has unique capabilities in dealing with unlabeled data through unsupervised learning, with a clear efficiency in extract nonlinear features from high-dimensional datasets. Autoencoders have multiple formats and approaches that have been used to solve different problems in many domains.

Proceeding from this advantage of deep learning and autoencoders, the use of deep stacked denoising sparse autoencoders was investigated to learn the feature representation of faces images by automatic faces and facial features extraction, to reach a robust face recognition system. The proposed face recognition system namely Deep Stacked Denoising Sparse Autoencoders (DSDSA) combines the deep neural network technology, sparse autoencoders and denoising task.

As for the rest of the thesis, Chapter 2 is customized to display the scientific background of the search, Chapter 3 will discuss the materials and methodology of the thesis; the proposed face recognition system will be presented with details in the first part of Chapter 3, while the deep learning model is released in the second part and the Autoencoders have been explained in the third part. Chapter 4 gives the experimental results using some chosen databases. Finally, discussion and conclusion are provided in Chapter 5.

2. BACKGROUND

2.1. FACE RECOGNITION

2.1.1. Definition

Face Recognition System is a biometric technology interested in analysing face images in order to do identification or verification tasks. System tries to extract robust features from the input face to learn face representation.

Face identification: Given unknown input face, the algorithm has to look for face identity against known faces.

2.1.2. Face Recognition Approaches

Face recognition is one of the topics that has been researched extensively over many years, there are many techniques that have been suggested; these approaches can be grouped as follows:

2.1.2.1.Appearance Based Approaches

In this approach different statics operations were applied to feature space where images represented as a high dimensional vector, the matching score is the differences between projection vectors. Eigenfaces, Fisherfaces, PCA-based and Linear Discriminant Analysis (LDA) are the most prominent methods in this approach.

However, methods of this approach are greatly affected by the conditions in which the images were taken like lighting, pose and resolution, making them ineffective in dealing with local variations in the faces.

2.1.2.2.Geometry Based Approaches

In this approach the face template is constructed by statistical analysis of a set of geometrical features. However, geometric properties do not give an enough distinctive representation of the faces.

2.1.2.3.Statistical Learning Methods

In the Statistical approach, features are learned from face images, these tools are as PCA, LDA, Discrete Cosine Transform (DCT) and Gabor Wavelet.

2.1.2.4. Deep Methods

In the deep learning methods, the representation of faces is learned during the training process, different approaches of artificial neural networks have proven their worth in face recognition.

2.1.3. Uses and Applications

The use of face recognition technology has spread widely in different fields due to its many advantages and ease of use, superior to other biometric systems such as fingerprint and eye recognition. These are examples of the most prominent of these applications and uses:

- **Security and law enforcement issues**: Face recognition contributed in the improvement of security systems significantly, it plays a role in systems such as policing, passport verification, driver's license and airports surveillance. Caterpillar[1] has applied facial recognition technology to wake-up a sleepy driver.
- **ID verification**: Face recognition helps businesses and institutions with prevent fraud and identity theft in addition to maintaining the time. It used widely in voting systems, attendance tracking, check in and credit cards verification.
- **Social media**: Many of the popular social networking platforms are now using face recognition technology to offer benefits to its users. Facebook introduced DeepFace [2] with accuracy up to a 97%, Social Mapper [3] by Trustwave designed to automatically locates profiles on many social networks based on a name and picture, PimEyes [4] and PicTriev[2] are examples of face finder search engine.
- **Mobile and smart devices applications**: Many face recognition based applications were developed to secure and facilitate mobile usage. You can unlock your mobile screen or specific apps, protect your own files, or search for contacts using face recognition applications. As of iPhone X Apple has added a facial recognition feature called FaceID[3] to unlock the phone replaced the fingerprint feature in the previous versions.
- **Marketing**: Face recognition and facial expressions helps brands and marketers to offer attractive advertisements for their customers based on identifying the detected face age,

[1] https://www.cat.com/en_US/support/safetyservices/products/dss.html.
[2] http://www.pictriev.com/.
[3] https://www.apple.com/lae/iphone-xr/face-id/.

gender and expression. Snapchat[1], Alibaba[2] and Amazon[3] use face recognition technology to provide better services to their customers.

- **Biomedical engineering**: Face recognition assisted in many medical uses, FaceMatch[4] uses "Imagus[5]" face-matching technology to perform a diagnostic test for people with undiagnosed moderate to severe intellectual disability.

2.2. MACHINE LEARNING AND DEEP LEARNING

2.2.1. Machine Learning

Machine learning is a branch of artificial intelligence (AI), which is to make the computer simulate the human think, while the machine learning principle is how to make the machine learn from the data to build analytical models automatically.

The heart of the machine learning is the data. Its work is centered around how to take advantage of the raw data by using powerful features that express the real character of the data without repetition or uselessness. Any deficiency or anomaly like noise and bias in the training data will lead to undesirable results.

The main challenge in machine learning also comes back to the data, getting enough data to solve a particular issue is the first challenge. Not far from this, the problem of overfitting, resulting from noisy presence in the training data or the inadequacy of data that were introduced in the training phase, which make the resulted model non-comprehensive and not useful to dealing with unknown data. Overfitting leads to very complex model with too many parameters. The curse of dimensionality is a challenge that must always be considered in matters of machine learning.

[1] https://www.snapchat.com/.
[2] https://www.alibaba.com/.
[3] https://www.amazon.com/.
[4] https://facematch.org.au/.
[5] https://www.imagus.com.au/.

2.2.2. Deep Learning

2.2.2.1. Definition

Deep learning is a machine learning branch, aims to discover the latent representation of data by learning from the data itself.

Deep learning methods, unlike classical machine learning where features are extracted manually, extract relevant features automatically from raw data. Deep learning methods learn by example to perform tasks like classification directly from data.

The relationship between deep learning and data is integrative, more data often improves the efficiency of results. However, getting the best results with minimal data usage is the best solution.

2.2.2.2. Limitations

Just as learning through data is one of the most important advantages of deep learning, it may be a cause of some drawbacks. The most important of these is that if it is trained through certain data it may not be able to circulate this solution to other data which has not seen before, what is known as the problem of overfitting and generalization.

The previous problem tends to appear when the data is fairly small. However, using large data may be costly in time and resources.

Also, when the data contains noise, the solution will be inaccurate when dealing with clean data. This may lead to a so-called bias problem.

One of the other drawbacks of deep learning is that latent representation may be ambiguous, especially when the network is too deep. Training a deep learning algorithm is not an easy task at all, as it is often prone to problems such as vanishing gradient and local minima.

It is worth mentioning here that many effective solutions have been proposed to overcome these problems, making deep learning methods increasingly important recently.

2.2.2.3. Applications

Deep learning methods have proved their worth by achieving state-of-the-art accuracy results in various fields, these are examples of the most prominent areas in which deep learning methods achieved remarkable success:

- Object detection and recognition
- Speech recognition and language translation.
- Self-driving cars.
- Natural language processing (NLP).
- Medical Research.
- And many more...

2.3. SUPERVISED, UNSUPERVISED AND SEMI-SUPERVISED LEARNING

The supervised and unsupervised are terms to describe the type of data used in machine learning model.

2.3.1. Supervised Learning

In supervised learning each example in dataset is labelled. Given a dataset $X \in \mathbb{R}^n$, n-dimensional real number vector, for each input variable $x^i \in X$ there is y^i label. The algorithm training to learn the mapping function from the input to the output is: Y = f(X).

The goal in the supervised learning is to approximate the mapping function which can predict the correct label y' for unknown input data x' .

Evaluating supervised learning algorithms is fairly easy, because the correct answers of the input data are available. Data is often divided into two sets: a training dataset and test dataset, other techniques such as cross-validation can also be used (Figure 2.1).

The performance measure is usually done by analysing some metrics like Accuracy, Precision, F-Measure, Error Rate or others.

In addition to the lack of availability, supervised methods can be very time-consuming. However, supervised models may need to be constantly rebuilt to get the correct answers to new data that was not covered by the training data.

The main two fields of supervised learning are: Classification and regression problems.

- Classification: The goal of the algorithm is to predict discrete values, mostly binary when the target is limited to only two classes e.g. {1,0} or multi-class where the label is a member of specific group like animals or cars.

- Regression: It is used to predict target with continuous or real values, e.g. car prices.

Following the most common supervised learning algorithms:

- Nearest Neighbor.
- Naive Bayes.
- Decision Trees.
- Linear Regression.
- Random Forest.
- Classical Neural networks.

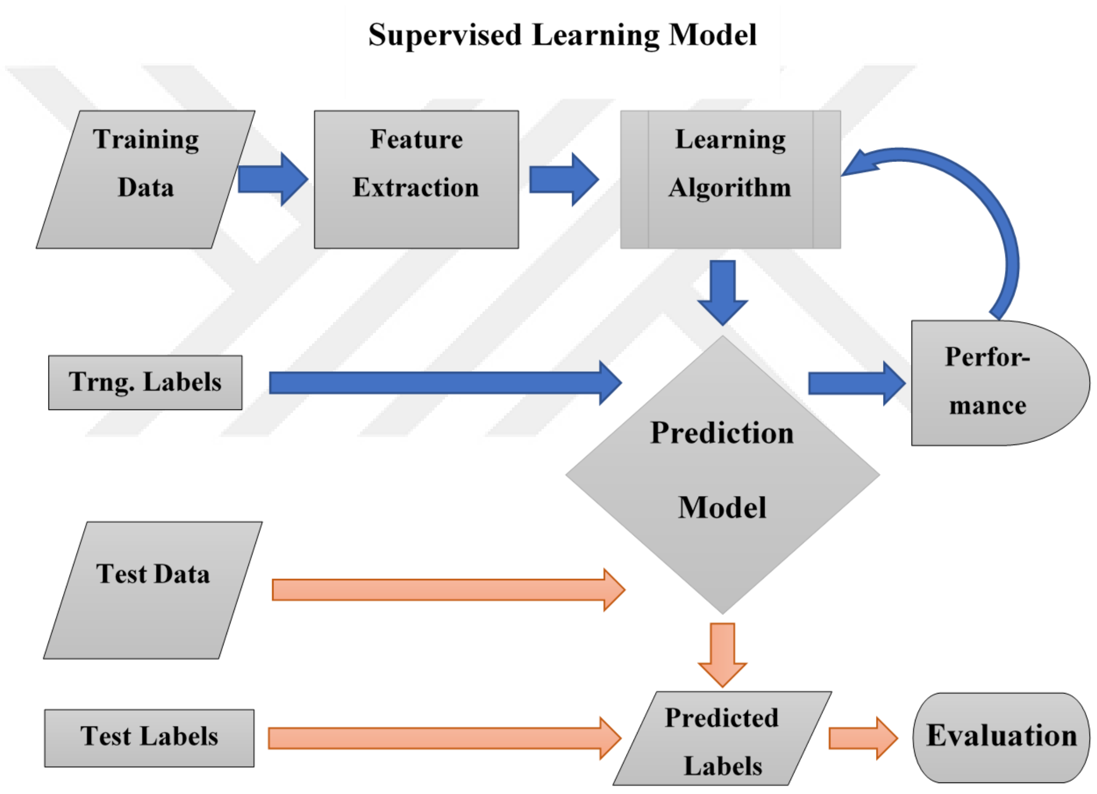


Figure 2.1: Supervised Learning Model.

2.3.2. Unsupervised Learning

On the contrary of supervised learning, each example in unsupervised learning dataset is unlabelled, only input data X is given. Due to the lack of labelled data, unsupervised learning provides a viable alternative through reliance on unlabelled data.

The goal of unsupervised learning algorithms is to learn the data structure or representation from the input data. The algorithm may cluster or group unlabelled data based on similarities or variation, it cannot be applied to a problem such as classification or regression directly because there are no target labels (Figure 2.2).

Evaluating unsupervised learning algorithms is not trivial, because there is no correct answer of the input data, labelling data impractical due to the cost of time and effort. Several alternative methods have been proposed to evaluate the results, by including some probability measure (e.g. log-likelihood) or by distinguishing between intra-class and inter-class for each member of cluster, some other method evaluate the resulted latent feature by using it as input for a supervised learner. Visualization is another way when dealing with an image problem.

However, unsupervised learning usually gives approximate and unpredictable results compared with the supervised learning. Also determining its accuracy is not evident, but it provides a reasonable solution to many of the problems that cannot be solved in supervised way.

The main fields of unsupervised learning are:

- Clustering: The idea of clustering is to automatically separate data to groups depends on the similarity and variety, such as grouping animals by species or cluster documents by topics.
- Association: Association looks for the interrelations between the elements of data, which is usually used in the recommendation field, such as recommend x item to people who bought y item, based on an analysis of buyers' behavior or on the correlation between the items themselves.
- Anomaly detection: To detect abnormal elements in the dataset, such as detect the rotten pieces of hardware. Also used to analyze behaviors, such as detecting fraudulent.

Following the most common unsupervised learning algorithms:

- k-means clustering.
- Association Rules.
- Autoencoders.
- Gaussian mixture models

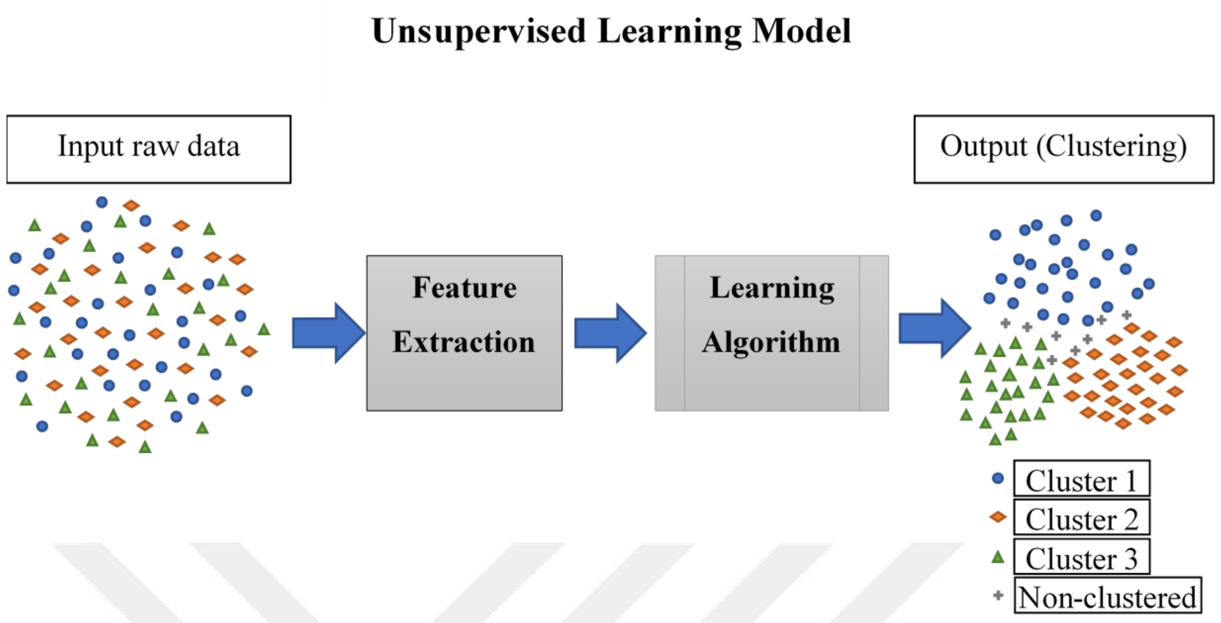


Figure 2.2: Unsupervised Learning Model.

2.3.3. Semi-Supervised Machine Learning

Semi-supervised learning is a method that combines supervised and unsupervised learning. Generally, the data is mostly unlabelled and some of data is labelled, it exploits the abundance of unlabelled data in addition to benefiting from the labelled data for fine-tuning and evaluation.

Semi-supervised learning is useful to solve such classification or regression problems when there is no enough labelled data, but the unlabelled data is available (Figure 2.3).

The advantage of semi-supervised learning is the ability to handle both supervised and unsupervised fields, it is used in classification and regression as well as clustering and association and so on.

Co-Training is one of the most commonly used semi-supervised learning technique.

Following the most common semi-supervised learning algorithms:

- Label Propagation.
- Transductive support vector machine.
- Laplacian regularized least squares.

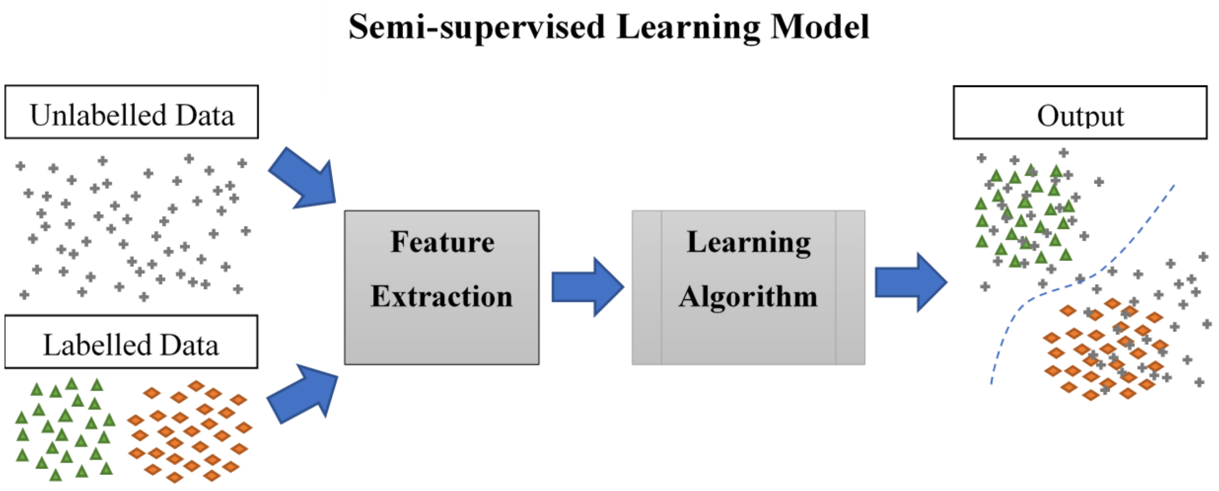


Figure 2.3: Semi-supervised Learning Model.

2.4. GOALS AND CHALLENGES

2.4.1. Goals

All research related to Machine Learning shares the goal of reaching high-accuracy results trying to reach the accuracy of the human and even surpass it, of course with trade-off, speed (time) and size. Face recognition technology aims to outdo other biometric techniques such as DNA matching, fingerprint recognition, iris and retina recognition. More accuracy improves the security and reduces error rate.

Full Automation, minimizing as much handcraft effort as possible, is a big and important goal as well. The face recognition technology does not need to seek or direct contact with the target. Thanks to advanced high-speed and resolution cameras, pictures can be taken and identified the faces without any effort from the person, even without attracting the attention of pedestrians, this also reduces overall process time and the cost as well. When incorporating the benefit of deep learning methods to automatically extract representative features and uses of unsupervised learning where no manual labels needed while training, the distance can be cut to get a fully automatic system. Fortunately, the Autoencoder can take advantage of these techniques.

2.4.2. Challenges

Despite the reach of advanced results and solutions in the face recognition and deep learning field, but there are important challenges facing the dissemination and generalization of these solutions. The large degrees of differences in face images include the variations in illumination, pose, noise and expression can significantly affect the performance.

Street and surveillance cameras and manual photography contain pictures with great difference in content in terms of angle rotation, scenery and resolution, this also applies to available face datasets. For example, existing solutions have been very successful in dealing with frontal and semi-frontal images, but they still behave poorly with images containing faces with high pose angle degrees, this is also true for images that have other issues like low resolution or lighting.

Another challenge when face contains a partial occlusion resulting of wearing sunglasses, veil and hats, even makeup, tattoos, beard and moustache represent the same challenge.

The Big Brother Watch, an organisation leading the protection of privacy and civil liberties in UK, claimed that the accuracy of the facial recognition system used by UK police forces, South Wales Police and the Metropolitan Police is “almost entirely inaccurate” with 91%-98% inaccurate [5].

2.5. CONTRIBUTIONS

The contributions of the proposed system can be outlined as follows:

- Complete and robust face recognition system, can deal with various faces datasets, including unconstrained faces with variability in pose, expression, illumination and scene.
- Exploit the sparsity and denoising terms to build deep unsupervised denoising sparse autoencoder that outperforms standard autoencoder by extracting more robust features from variances face images.
- Use two robust classifiers, multiclass-SVM and Softmax classifier to obtain more accurate recognition.
- System can be used for both faces identification and verification.

The experimental results of using the proposed (DSDSA) method achieved comparable accuracy results in face recognition when it was applied to known face databases like ORL, Yale and Caltech. Also, can be generalized to other recognition tasks thanks to its ability to extract robust features.

2.6. RELATED WORK

In recent years, face recognition has been studied extensively, many methods have been raised to achieve more accurate results, methods can be summarized into the ones that use deep learning and which don't use.

2.6.1. Face Recognition

Early studies are divided into template based and feature based methods. In template based methods the goal is to create a general template for each face, Eigenface and Fisherface are the most famous methods in this category, many methods of dimensionality reduction like Principal Component Analysis (PCA) [6], Linear Discriminate Analysis (LDA) [7] and Local Linear Embedding (LLE) [8] were proposed nonlinear extensions based on Kernel-PCA [9] and Kernel-LDA [10] were proposed in order to solve the problem of nonlinear distribution of face images. Feature based methods such as Local binary patterns (LBP) [11], Gabor [12], HOG [13], SIFT [14] and others were applied to extract robust features which were used to represent the different faces. In classification field, Support Vector Machine (SVM) [15] was proposed and used to classify the extracted features, various classifiers like Nearest Neighbor (NN) [16] and Nearest Subspace (NS) [17] were also proposed. Distance-based methods, which consist in computing a similarity metric proposed [18],[19],[20]. The sparse representation is also gradually utilized to solve face recognition [21],[22],[23],[24],[25]. Many other methods combined between some of previous methods [23],[24],[25],[26][27]. However, despite some of the success that these methods achieved, they stayed to suffer from their linear nature, generalizability problem or shallow structures.

2.6.2. Deep Learning Based Face Recognition

Lately, thanks to their great successes in image representation and recognition fields deep learning methods achieved great investigating on face recognition [2],[28],[29],[30],[31],[32]. Convolutional Neural Networks (CNNs) have taken the main interesting in many researches, DeepFace [2] trained a deep CNN to classify faces, Sun et al. has developed DeepFace work with DeepID [30],[31][32],[33]. Siamese network was suggested by Chopra et al. [34]. Huang et al. [35] learned a generative deep model without supervision. Sun et al [29] used supervised multiple deep models. Zhu et al. [36] used multiple deep Convolutional Networks. FaceNet [37] by Schroff et al. achieved state-of-the-art face recognition performance using deep convolutional network. Parkhi et al. used a deep CNN trained to classify faces using a dataset of 4 million examples [38].

2.6.3. Autoencoders for Face Recognition

Hinton and Salakhutdinov [39] proposed a deep autoencoder networks training algorithm. Accordingly, autoencoders have been researched extensively in the field of face recognition and related tasks. Kan et al. [40] used stacked progressive autoencoders for face recognition. Zhang et al [41] proposed Coarse to fine autoencoder for real time face alignment. Gao et al. [42] used stack supervised autoencoder for face recognition. Ding and Tao [43] used stacked autoencoder for learning face representations. Zhang et al [44] investigated sparse autoencoders with Softmax classifiers to build deep hierarchical for face recognition. Recently, Li et al. [45] proposed deep autoencoder with dropout.

3. MATERIALS AND METHODS

3.1. FACE RECOGNITION SYSTEM

3.1.1. Face Recognition Phases

Face recognition system follows procedures usually followed in biometric systems, which include:

- Image acquisition: usually obtained via different cameras or by using digital images.
- Face finding: detect all existing faces in the given image.
- Face normalization: do some pre-processing steps like pose correction, warping, noise reduction and possibly colour removing (grayscale).
- Feature extraction: get the face representation.
- Face matching (classification): do face authentication.

3.1.2. The Proposed Face Recognition System

The proposed face recognition system consists of five modules: Face detection, face localization, normalization, feature extraction and faces matching. Figure 3.1 shows the proposed face recognition system modules.

3.1.3. Face Detection

Face Detection represents the basic step in any face recognition system, the next steps will depend heavily on them. Therefore, it is important to choose a face detection algorithm that works effectively with different images.

Face detection is to determine the presence of the face in the given image. Given input images, face detection algorithm will decide if there is a face in the image or not, return the face's coordinates if detected.

There are many challenges start from the absence of face from the given image, multiple faces existing, pose, lighting, resolution, expression and skin color.

One of the salient problems is when the algorithm detects non-face item wrongly as face, what is called a false positive detection. One may have to deal with this problem manually.

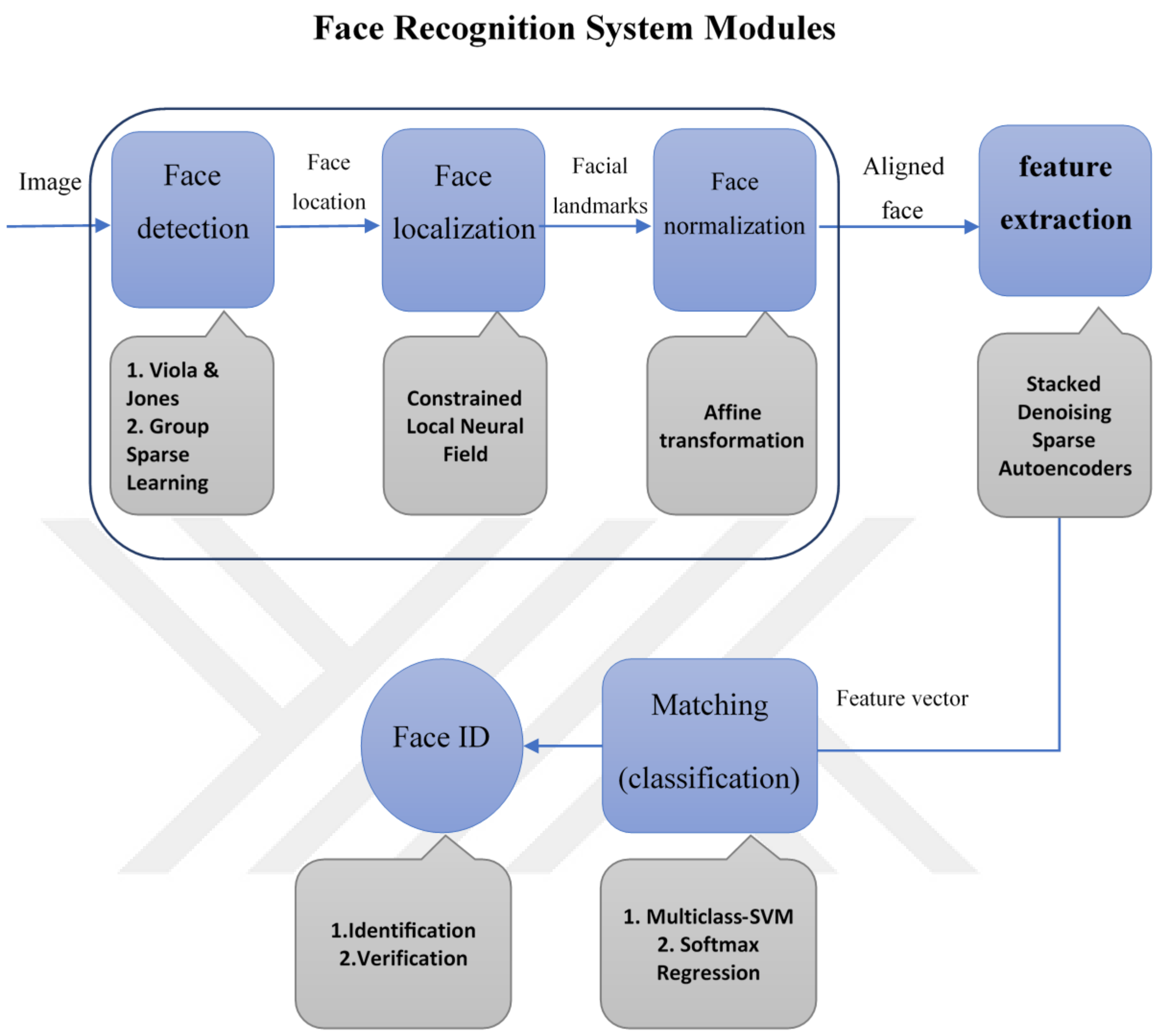


Figure 3.1: Face recognition system modules and the methods used for each one.

In this proposed system, integrated face detector between Viola and Jones; Haar feature-based cascade classifiers [46] and Group sparse learning produced by Xiang Yu et al. [47] were applied in order to detect and locate a bounding box around the face. First, the system tries to detect face using Viola and Jones, thanks to its high speed and accuracy in frontal faces. However, Viola and Jones has lack of performance when dealing with faces in complicated conditions, if Voila and Jones failed to detect any face the system will try to use the group sparse learning method.

Following an overview of these two methods:

3.1.3.1. The Viola–Jones (VJ) Haar Feature-based Cascade Classifier

The Viola–Jones detector [46] was proposed in 2001 by Paul Viola and Michael Jones, it is probably the most famous approach for frontal 2D detection, it involves exhaustively searching an entire image for faces, with multiple scales explored at each pixel using Haar-like rectangle features boosting classification.

VJ involves a very simple image representation based on Haar wavelets, an integral image is used for rapid feature detection, the AdaBoost machine-learning method is used for selecting a small number of important features and a cascade combination of weak learners are used for classification.

The Haar-like feature looking is done by scans the given image for a supposed face by using filters based on the difference in density (Figure 3.2), thus revealing the distinctive parts of the face, such as the eyes, nose and mouth.

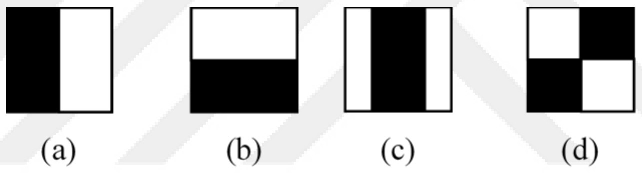


Figure 3.2: Haar-like filter samples: two-(a,b), three-(c), four-(d) rectangles.

Integral image representation is a very fast way to calculate rectangular features, where the summation of the pixel values in each rectangle is calculated to judge whether certain characteristics exist or not (Figure 3.3). This leads to too many features, where the AdaBoost is used to do feature selection process from what so called weak classifiers, AdaBoost then do a cascade training (Figure 3.4) to train a strong classifier.

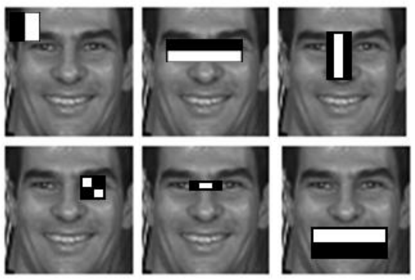


Figure 3.3: VJ framework scan an image to detect the features existence.

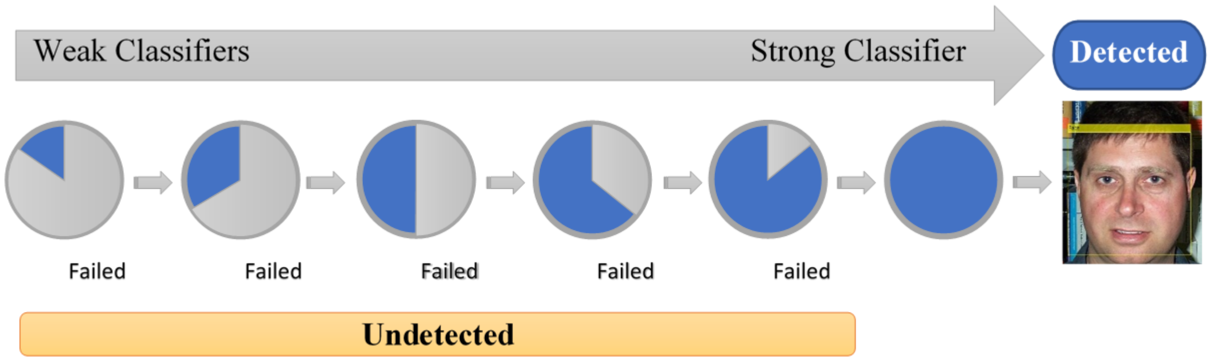


Figure 3.4: Cascade Classifier.

The original VJ implementation is designed for upright frontal image. VJ is characterized by a slow training, but very fast classification. Samples of successfully detected face and detection with false positive in the Figure 3.5 and Figure 3.6, respectively.

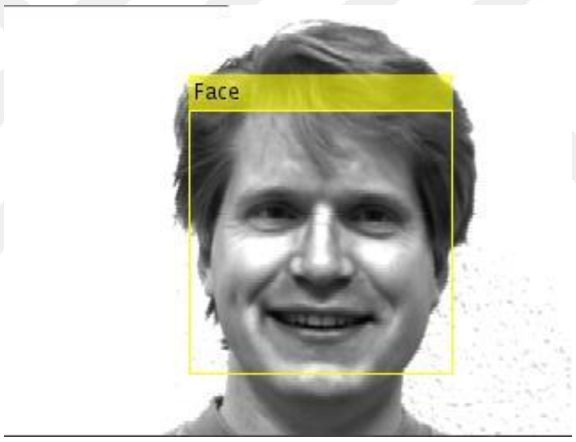


Figure 3.5: Successfully detected face by VJ.

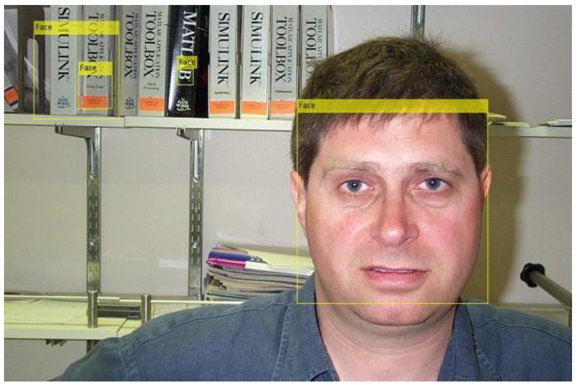


Figure 3.6: Wrongly detected faces by VJ (FP).

Nevertheless, VJ does not deal adequately with unrestricted images that contain large differences in lighting, pose, expression etc… (Figure 3.7).

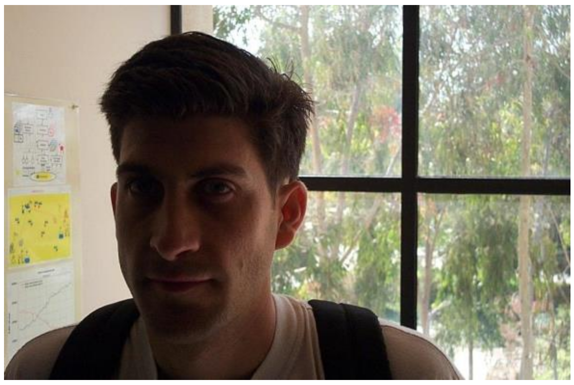


Figure 3.7: Undetected Face by VJ.

3.1.3.2. Group Sparse Learning

Group Sparse Learning (optimized part mixture model): Is a two-stage cascaded deformable shape model, produced in 2013 by Yu et al. [47]. A group sparse learning method is proposed to automatically select the optimized anchor points. Then a two-level cascaded deformable shape model is presented to search global optimal positions. This method may give better results in dealing with images that contain high pose degree, so it will only be used if the Viola and Jones method fails to capture the face (Figure 3.8), because it is more expensive in terms of time.

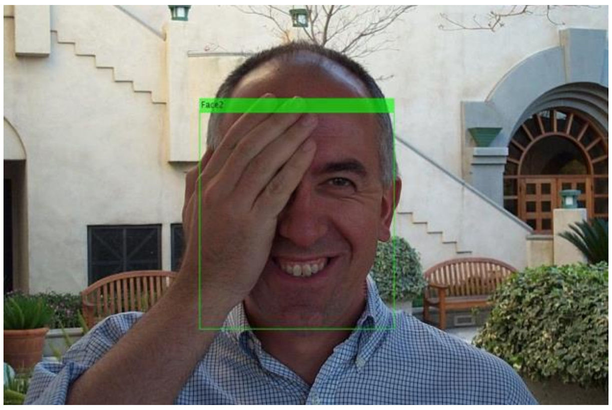


Figure 3.8: Example of an image in which the face was not detected by VJ but detected by the Group Sparse Learning method.

3.1.4. Face Landmarks Localization

Landmarks are the distinguishing points of the main facial parts like eyes, mouth, and noes. Detecting landmarks help in adjusting the face which contains varying pose includes yaw, pitch or roll.

Face Landmarks Localization (Facial feature detection) is the process to detect landmarks positions of the facial feature points: eyes, contours, mouth, nose, and eyebrows. After face detection, facial landmarks will be used in the next step to align facial images to a mean face shape. Any error in estimating landmarks leads to error in alignment and transformation processes in the next step.

The Constrained Local Neural Field (CLNF) model for robust facial landmark detection in the wild was produced in 2013 by Baltrusaitis et al. [48]. Local Neural Field (LNF) patch expert incorporates with Non-Uniform Regularized Landmark Mean-Shift (NU-RLMS) a CLM fitting method which trusts reliable patch experts more. An overview of this model can be seen in Figure 3.9.

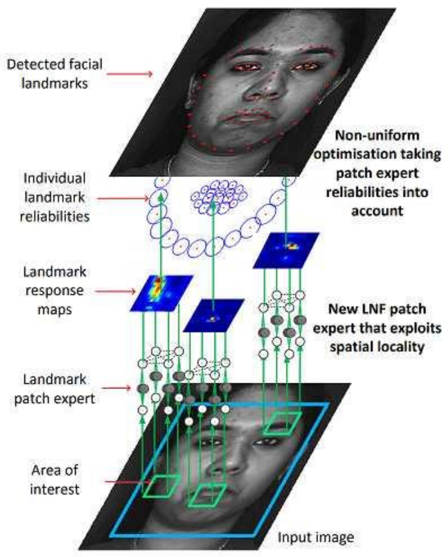

Detected facial landmarks
Non-uniform optimisation taking patch expert reliabilities into account
Individual landmark reliabilities
Landmark response maps
New LNF patch expert that exploits spatial locality
Landmark patch expert
Area of interest
Input image

Figure 3.9: Constrained Local Neural Field (CLNF) model.

CLNF is an undirected graphical model that can model the conditional probability of a continuous valued vector y depending on continuous x (the pixel intensity values in the support region). Given $X = \{x_1, x_2, \dots x_n\}$ is a set of observed input variables, $y = \{y_1, y_2, \dots y_n\}$ is a set of output variables that we wish to predict, $x_i \in R^m$ represents vectorised pixel intensities in patch expert support region, $y_i \in R$ is a scalar prediction at location i. The model for a particular set of observations is a conditional probability distribution with the probability density:

$$P(y|X) = \frac{exp(\Psi)}{\int_{-\infty}^{\infty} exp(\Psi)\, dy} \tag{3.1}$$

where $\int_{-\infty}^{\infty} exp(\Psi)\, dy$ is the normalization function and Ψ is the potential function.

This method was used to detect 68 landmarks for each face which determines the main areas of the face (Figure 3.10). The eyes coordinates (landmarks) were utilized to rotate and align the face in the next step.

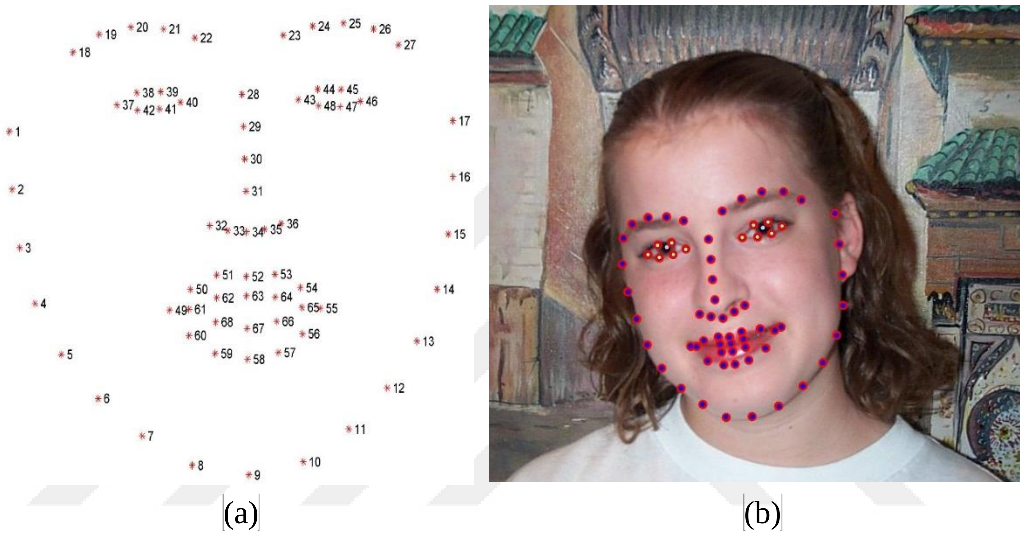


Figure 3.10: Landmarks. (a) The 68 facial landmark representation. (b) Example of 68 landmarks localization for a face from Caltech dataset.

3.1.5. Face Normalization

Face pose, position, and illumination play an essential role in distinguishing between individuals. The goal of normalization is to minimize the difference of pose, position, and illumination for the same individual, this includes the impact of non-facial details like background and clothing.

The first step of normalization can be done by face rotating, scaling, warping and cropping. The center points of the eyes obtained from detected landmarks in previous step, the line and distance between eyes centers help to determine the scale and the angle of rotation, which will be used to make an affine warping. After warping, the face will be cropped so that the eyes are centered in all faces. The cropping area is determined by the left eye center, while the margin around the eye is a parameter usually takes ranges between 0.1 and 0.3, also the width and height of cropping face are parameters can be determined manually. In experiments, 0.3 were chose for margin and [32 x 32] pixels were selected for [width x height] respectively (Figure 3.11- a).

To obtain the photometric normalization, the illumination is normalized by the grey information. And pixels are rescaled to [0,1] by dividing each pixel by 255, the mean can also be subtracted to center the pixel values in the face image see (Figure 3.11- b).

The aligned and normalized faces are inserted into matrix as well as saved in a -MAT file so that they can be used independently in the following steps as ‘ready: normalized’ faces.

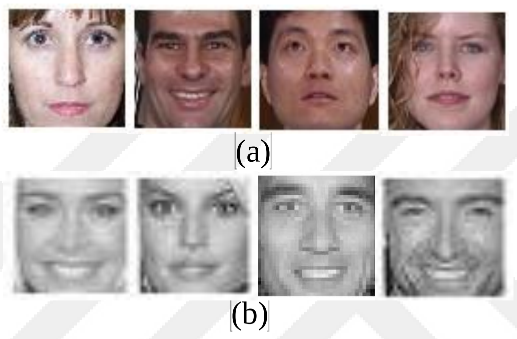


Figure 3.11: Example of normalized faces (a) face rotating, scaling, warping and cropping. (b) faces converted to grayscale.

3.1.6. Feature Extraction

3.1.6.1. Feature Extraction Theory

Feature is the distinctive characteristic of the object being researched which can distinguish it from others. When it comes to images, the simplest examples are blobs, edges and corners.

A combination of these characteristic features forms what is known as a feature vector, the vectors that represent an object form the feature space which determines the *n*-dimensions.

Feature extraction is the process of eliciting the key characteristics of input data, in order to obtain a concise and accurate model that assists in operations such as recognition or predicting. Feature extraction is useful for solving problems with large-size and high dimensional data.

3.1.6.2. Manual vs. Automated Feature Engineering

The previously used machine learning algorithms were based on manual feature extraction or selection, while deep learning relied on automatic feature extraction.

Automatically feature extraction methods outperforms manual methods in many important things; they extract the most relevant and useful features to build a robust model, and they prevent wasting valuable data. The automatic methods are less expensive in time and less prone to errors. While the manual methods are suitable for specific task being processed, the automatic method may be more general and reusable.

Many feature extraction methods have been recently introduced, such as PCA, ICA, SURF and LBP. However, choosing the appropriate method depends on the type of task to be resolved and the nature of the data.

3.1.6.3.Feature Extraction Using Deep Learning

The deep learning methods have helped to create functional models contributed to solve many problems in different categories consist of supervised learning, like classification and regression tasks, and unsupervised learning such as clustering tasks.

Deep learning methods surpass other methods in dealing with non-linear problems and unsupervised learning, they show remarkable success in building automatic feature extraction algorithms which have the precision and ability to automatically learn feature representation to solve various tasks and datasets, they are also efficient in dealing with large data.

Several design choices have been proposed. Shallow methods were excluded because they allow only the linear transformation while non-linear deep architecture methods can learn more complicated representations. Furthermore, many deep architecture designs have been proposed; Despite the great success of the CNNs, there are many determinants in their use; CNNs are mostly used for very specific task. Because of its complexity and local connectivity, the generalization performance issue poses a challenge especially for highly non-linear or time-varying object appearance variations, and there are so many more settings to manage. For example, VGG is a 16 layer neural net, while Microsoft’s ResNet model has 152 layers. However, the CNN is designed to handle supervised learning which depends on labeled data. Another issue is that it mostly needs large data for training, for example DeepFace [2] trained with 4 million images and FaceNet [37] trained with 200 million images.

On the other hand, autoencoder is an unsupervised learning algorithm investigates unlabeled data. It is simple, intuitively understandable and easy to implement. Autoencoders are more general because they specify nothing about the network topology. An additional reason for choosing autoencoders in this work is that they can be stacked on top of each other to form an effective feature extraction method while maintaining its simplicity.

Autoencoder is one of the leading deep learning methods used to automatically extract the features, especially with unlabelled data (unsupervised learning). Practical experiments have proven the superiority of the autoencoder in this field [49].

Feature extraction can be considered the heart of the system, where the face is represented by extracting the salient features that effectively reflect the individual differences between the faces. To obtain greater efficiency, features are extracted from the images that were normalized in the previous step. Deep Denoising Sparse Autoencoder has been used in the proposed system to extract a robust set of features to represent each face, more details in the following sections.

3.1.7. Face Matching

The extracted features of the previous step are used to train a classifier to match faces. One face can be compared to multiple faces, which is known as face identification. Or two faces can be compared to see whether they belong to the same person or not, this is known as face verification.

Linear classifier is the simplest possible function can be used to calculate the score of each class, Eq.3.2 shows the linear function

$$f(x_i, W, b) = Wx_i + b \tag{3.2}$$

Where W is the weights and b is the bias term, if bias is ignored, the score of class j can be written as:

$$S_j = f(x_i, W)_j = Wx_i \tag{3.3}$$

Two commonly used losses for linear classifiers were used in the proposed system:

3.1.7.1. Multi-Class SVM

It is a development of Standard SVM which is for binary to the multi-class classification, where the face is given to the class that has a larger score than others by a given margin. Common approaches in Multi-Class SVM are “one-vs-all” and “one-vs-one” which are belong to Error-Correcting Output Coding (ECOC) algorithm.

Given l_i the true class of *i*th example, the multi-class SVM loss for *i*th example is:

$$L_i = \sum_{j \neq l_i} max(0, S_j - S_{l_i} + margin) \tag{3.4}$$

Where L_i is the loss for *i*th example. S_j, S_{l_i} are the score of class *j* and correct class respectively, margin is a parameter defined by the user. The $max(0, -)$ is called the hinge loss.

3.1.7.2. Softmax Classifier

Softmax classifier is the multi-class version of Logistic Regression. Instead of the hinge loss used in SVM, a cross-entropy loss is used here to calculate the probability of each class:

$$L_i = -log\left(\frac{exp^{S_{l_i}}}{\sum_j exp^{S_j}}\right) \tag{3.5}$$

3.2. DEEP LEARNING MODEL

The deep learning term refers to network models with many hidden layers, beside of input and output layers, versus shallow nets which contain only one hidden layer. Each hidden layer in deep model is composed of hidden units, each unit takes its inputs from the previous layer and calculates its activation according to the given weights, outputs its own features. These learnt hierarchical features enable the system to decode the output from data automatically. Figure 3.12 shows a simple hierarchy of deep network model.

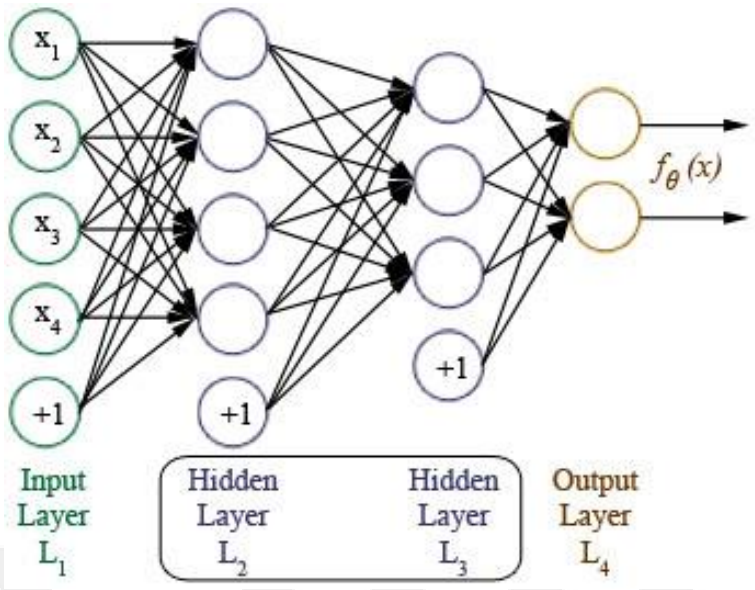


Figure 3.12: Simple hierarchy of deep network model with two hidden layers.

3.2.1. Training Deep Model

Training a deep model generally suffers from time complexity where weights are initialized randomly and local minima issues, several methods have been suggested to overcome these difficulties. One of these successful methods is proposed by Hinton [50].

3.2.2. Greedy Layer-Wise Training

Hinton, et al. (2006) provide an algorithm to pre-train each layer of the Deep Belief Network (DBN) using an unsupervised approach. This greedy layer-wise unsupervised learning algorithm first involves training the lower layer of the model with an unsupervised learning algorithm which yields some initial set of parameters for that first layer of the network. This output then acts as the input for the following layer which is similarly trained, resulting in initial parameters for that layer. Again, the output from this layer is used as the input for the next layer until the parameters for each layer are initialized. The overall output of the network is delivered as the final activation vector. Following this unsupervised pre-training phase of stacked layers, the entire network can then be fine-tuned in the opposite direction using backpropagation in this supervised learning phase.

3.2.3. The Number of Layers

There is no specific rule to determine the number of hidden layers and hidden units in each layer, the number of hidden layers and units are usually determined by the practical experiment and the evaluation of the efficiency of the test set, the number of hidden layers and units that give the best result will be chosen.

Different factors influence the determination of the number of hidden layers and units, including the nature and size of the input images. While some researchers have developed rules-of-thumb to determine the number of hidden units with respect to the number of input size, output size or training size, but it is still not possible to determine the validity of any of these rules for generalization.

3.2.4. Weights Initialization

Optimizing the weights of autoencoders is a challenging task. Large initial weights lead to falling into local minima. On the other hand, small initial makes it infeasible when using a deep model. However, if the initial weights are already close to a good solution, optimization techniques, such as gradient descent, work well. It follows that employing a similar "greedy layer-wise" learning algorithm to stacked autoencoders is an effective method of pre-training the network of a deep autoencoder [39].

3.2.5. Regularization Term (Weight Decay)

Overfitting occurs when a model with high capacity fits the noise in the data instead of the underlying relationship. A regularization term (also called a weight decay term), such as L2 regularization, dropout or input noise are good ways to prevent overfitting. The most common is to use L2 regularization.

3.2.5.1. L2 Regularization

The most common form of regularization to prevent overfitting is to use L2 regularization (Eq. 3.6). That is, sum of the square of the weights W is added to the objective (loss) function.

$$J_{weights} = \frac{1}{2}\lambda \parallel W \parallel_2^2 = \frac{\lambda}{2}\sum_{l=1}^{L}\sum_{i=1}^{N_l}\sum_{j=1}^{N_{l+1}}\left(W_{ji}^{(l)}\right)^2 \quad (3.6)$$

where λ is the parameter to control the strength of regularization, L is the number of hidden layers and N_l is the number of neurons in layer.

3.3. AUTOENCODERS

An autoencoder is an unsupervised neural network aims to minimize the error between the input data and its reconstruction (output).

An autoencoder consists of three or more layers: an input layer, some of the hidden layers that form the encoding and an output layer whose units correspond to the input layer. Since the outputs of the network are equal to the input, the autoencoder's goal is to learn an approximation of the identity function [51].

3.3.1. The Basic Autoencoder

The basic autoencoder is an unsupervised learning algorithm, that is architecturally and structurally similar to feedforward neural networks.

A simple autoencoder encloses two processes, encoding and decoding. Let $\boldsymbol{x} \in [0,1]^d$ is an d-dimensional input vector for autoencoder, $\theta = \{W, b\}$ (W: weight matrix, b: bias vector) encoding maps input to $z \in [0,1]^{d'}$

$$z = f_\theta(x) = \sigma(Wx + b) \tag{3.7}$$

where $\sigma(\cdot)$ is a nonlinear activation function of encoder like sigmoid or tanh.

Decoding process maps z back to reconstructed vector $x' \in [0,1]^d$

$$x' = g_{\theta'}(z) = \sigma'(W'z + b') \tag{3.8}$$

Where $\sigma'(\cdot)$ is a nonlinear activation function of decoder and $\theta' = \{W', b'\}$, the weight matrix W' and bias vector b' for the decoder, autoencoder has tied weights when W' is equal to W^T.

The optimization is done by minimizing the average reconstruction error:

$$\theta^*, \theta'^* = arg_{\theta,\theta'}min \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(x^i, x'^i)$$

$$\theta^*, \theta'^* = arg_{\theta,\theta'}min \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(x^i, g_\theta(f_\theta(x^i))) \tag{3.9}$$

where x is the input data, x' is the reconstruction, $g_\theta(h)$ is the decoder output, $h = f_\theta(x)$ the encoder output and θ contains the weights W and the bias b. Here $\mathcal{L}$ is a loss function such as mean squared errors:

$$\mathcal{L}(x, x') = \|x - x'\|^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - x'_i)^2 \tag{3.10}$$

One of the common loss used with autoencoder is the cross-entropy:

$$\mathcal{L}(x, x') = \sum_i x_i \log\frac{1}{x'_i} = -\sum_i x_i \log x'_i \tag{3.11}$$

While the mean squared errors loss is suitable for pre-training autoencoders, the cross-entropy function was used in the final stacked autoencoder where a softmax layer is added at the top as a classification layer to train the classification model.

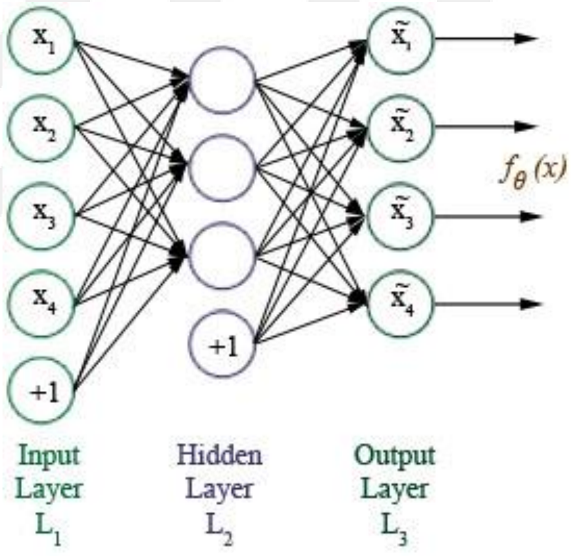


Figure 3.13: Basic Autoencoder

3.3.2. Sparse Autoencoder

While trying to learn the identity function, which seems trivial, a challenging structure of the data is discovered by bounding the number of hidden units and placing constraints on the network. But even when the number of hidden units is large (perhaps even greater than the number of input pixels), interesting structure is still be descovered by imposing other constraints on the network. In particular, if a "sparsity" constraint is imposed on the hidden units, then the autoencoder can still discover interesting structure of the data, even if the number of hidden units is large [51].

The sparsity is done by adding a sparsity constraint on the neurons, let $\hat{\rho}_j$ the average activation of hidden unit j

$$\hat{\rho}_j = \frac{1}{m} \sum_{i=1}^{m} [a_j]_i \tag{3.12}$$

Then, to approximately enforce the constraint $\hat{\rho}_j = \rho$ where ρ is a sparsity parameter close to zero, an extra penalty term will be added to the optimization objective that penalizes $\hat{\rho}_j$ deviating significantly from ρ, this penalty term is based on Kullback-Leibler divergence given by Eq. 3.13

$$KL(\rho \| \hat{\rho}_j) = \rho\, log \frac{\rho}{\hat{\rho}_j} + (1 - \rho)\, log(\frac{1-\rho}{1- \hat{\rho}_j}) \tag{3.13}$$

The sparsity penalty term J_{sparse} can be given in the following equation:

$$J_{sparse} = \beta \sum_{j=1}^{N} KL(\hat{\rho} \parallel \rho) \tag{3.14}$$

where ρ is a sparsity parameter, β is the weight of sparsity, *N* is the number of hidden units and $KL(\hat{\rho} \parallel \rho)$ is the Kullback-Leibler divergence as defined above (Eq. 3.13).

Let $\mathcal{L}(x, x')$ is the loss function as mentioned previously (Eq. 3.10 and Eq. 3.11), J_{weight} is the regularization term (Eq. 3.6) and J_{sparse} is the sparsity penalty (Eq. 3.14), then the overall cost function can be written as:

$$J_{cost} = \mathcal{L}(x, x') + J_{weight} + J_{sparse} \tag{3.15}$$

3.3.3. Denoising Autoencoder

Instead of using the input *x* in standard autoencoder, a partially corrupted copy of input $\tilde{x}$ is used to obtain a good representation. However, training is still trying to reconstruct the original input *x* (Figure 3.14).

Thus, the Eq. 3.7 above can be rewritten as:

$$z = f_\theta(\tilde{x}) = \sigma(W\tilde{x} + b) \tag{3.16}$$

The input can be corrupted in many ways, the simple way is done by setting a certain percentage of random pixels to zero or by adding random Gaussian noise to these pixels. This percentage is called the corruption level usually it is selected between 0 and 0.5.

The experiments improve that the denoising autoencoders can improve the generalization performance of the network and are able to learn Gabor-like edge detectors.

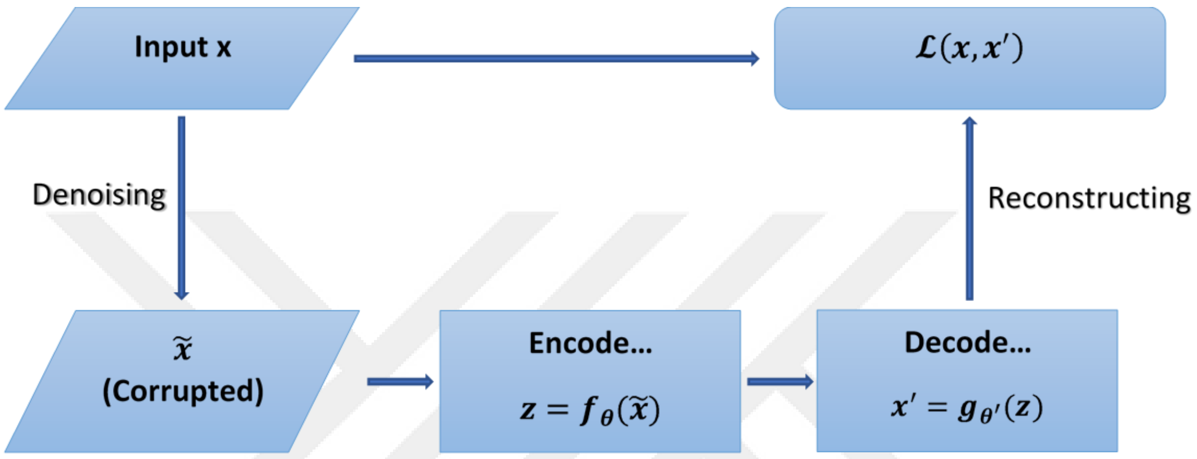


Figure 3.14: Denoising autoencoder process.

Figure 3.15 shows some samples from Yale dataset after applying gaussian denoising with corruption level = 0.3.

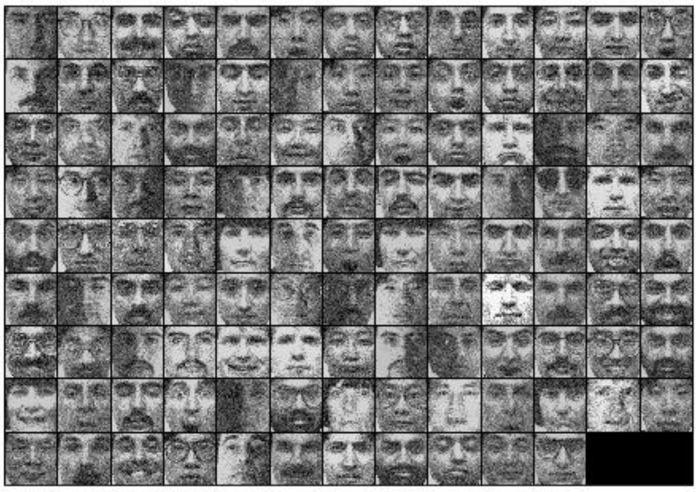


Figure 3.15: Random samples from Yale dataset after applying gaussian denoising with 0.3 corruption.

3.3.4. Stacked Denoising Sparse Autoencoder

Many denoising and sparse autoencoders are stacked together to form a robust unsupervised deep model that can be called Stacked Denoising Sparse Autoencoder. Each denoising sparse autoencoder takes its input from the activation of the previous layer and pre-trained independently. The goal of the unsupervised pre-training is to optimize some similar objective to put the parameters of all the layers in a region of parameter space layer wise. Stacked Autoencoders can extract useful features by unsupervised learning.

Stacked autoencoder training follows the greedy layer-wise strategy which is outlined in section 3.2.2, the overall output after fine-tuning will serve as an input to the classification algorithm, both (multi-class) SVM and Softmax regression were used in this study, the experimental results indicate that efficiency is close by using both methods, with a little preference for the SVM predominantly. Figure 3.16 shows the hierarchy and procedure of stacked autoencoder with two hidden layers and three output classes.

3.3.5. Fine-tuning the Parameters

The autoencoder involves a many of parameters to tune, that include layers number, neurons, weight decay term, sparsity and weights. The weights tuned as mentioned in section 3.2.4, while the grid search strategy was used to tune the other parameters, it is also coupled with cross validation technique for evaluation.

Fine-tuning step is applied to adjust the parameters of a stacked autoencoder, it has been proven to significantly improve the performance. Finetuning is done by training the whole pretrained layers as a single model using supervised learning mechanism to update the weights; the backpropagation algorithm is used to get the gradients to adjust the weights of whole network.

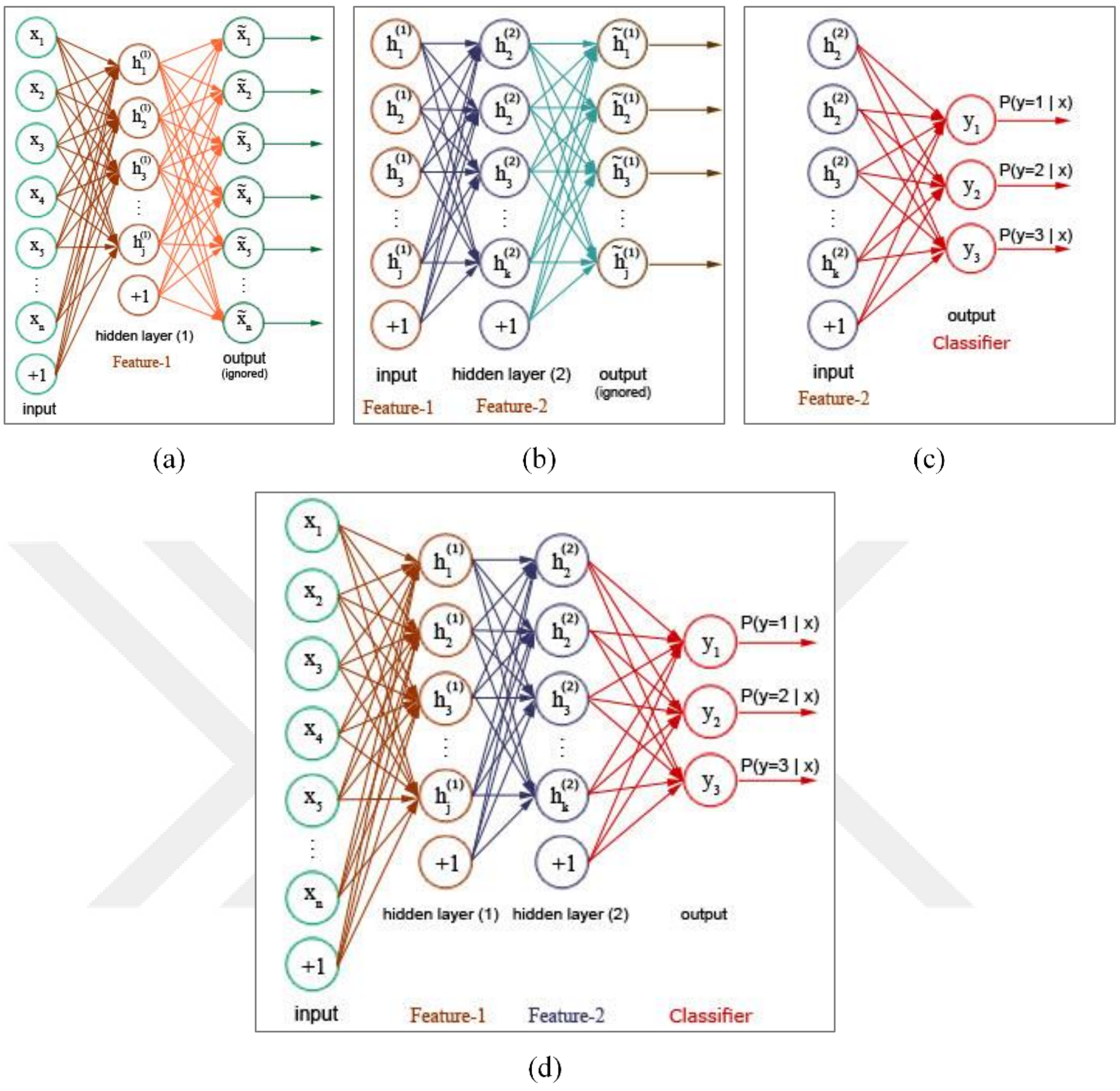


Figure 3.16: Stacked autoencoder with two hidden layers and three output classes. (a) first denoising sparse autoencoder to learn primary features $h^{(1)}$ on the raw input. (b) second denoising sparse autoencoder use the primary features $h^{(1)}$ as the "raw input" to learn secondary features $h^{(2)}$. (c) classifier layer: use secondary features $h^{(2)}$ as "raw input" to a classifier, training it to map secondary features to predict labels. (d) Full stacked autoencoder: combine all three layers together to form a stacked autoencoder with two hidden layers and a final classifier layer.

4. RESULTS

4.1. DATA SET COLLECTION

The Deep Stacked Denoising Sparse Autoencoder (DSDSA) system has been tested on four known face recognition datasets, the databases are ORL[1], Yale[2], Caltech[3] and PubFig15[4] databases.

ORL: The ORL face database, by the Olivetti Research Laboratory in Cambridge, UK.

- A total of 400 images for 40 individuals.
- There are 10 different images for each individual.
- Almost frontal, up-right and with dark background.
- Contains different expressions, varying lighting.

Yale: The Yale Face Database:

- A total of 165 images for 40 individuals.
- There are 11 different images for each individual.
- Contains different expressions, varying lighting and occlusions (sun glasses).

Caltech: Collected by California Institute of Technology.

- A total of 445 images for 26 individuals.
- There are different number of images for each individual.
- Almost frontal, with unconstrained backgrounds.
- Contains different expressions, varying lighting and occlusions.

PubFig15: A chosen subset of PubFig database which introduced by Kumar et al. [52]. The PubFig dataset contains 200 subjects and various numbers of images for different subjects. A modified subset called PubFig83, was introduced by Pinto et al. contains 13,838 images of 83 individuals.

- A total of 750 samples for 15 individuals.

[1] The ORL database was obtained from http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.
[2] The Yale database was obtained from http://cvc.cs.yale.edu/cvc/projects/yalefaces/yalefaces.html.
[3] The Caltech database was obtained from http://www.vision.caltech.edu/html-files/archive.html.
[4] The Caltech database was obtained from http://www.cs.columbia.edu/CAVE/databases/pubfig/ and PubFig83 http://vision.seas.harvard.edu/pubfig83/. A subset of 750 samples (50 images for 15 subjects) were chosen from PubFig83, it was given PubFig15 name.

- There are 50 different images for each individual.
- The images may have out-of-plane as well as in-plane rotation.
- Contains non-frontal, with unconstrained backgrounds.
- The alignment is not perfect but the majority of the face area is visible.
- Contains different expressions, varying lighting and occlusions.

After normalizing the face was automatically cropped to size 32 x 32 pixels, aligned (centered with respect to eyes) and converted to grayscale. Figure 4.1 and Figure 4.2 show the detected, aligned and cropped faces from ORL and Caltech datasets respectively.

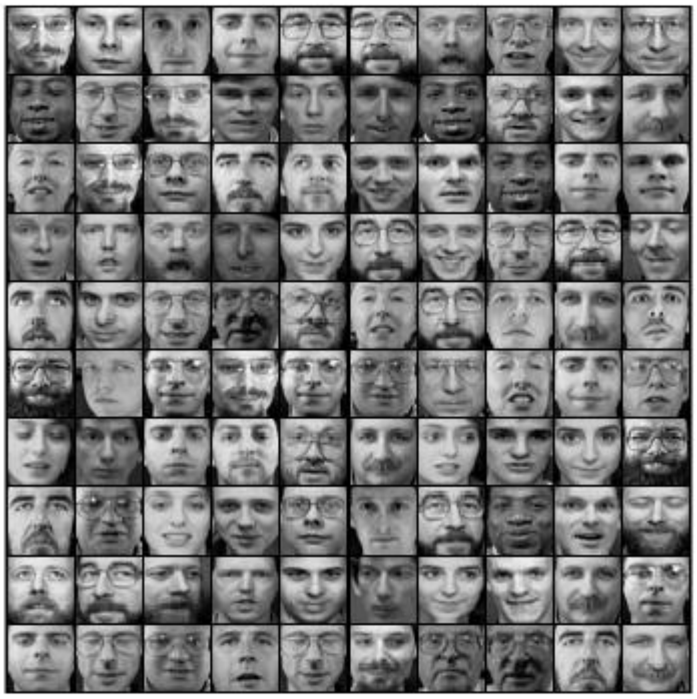


Figure 4.1: Sample images from ORL dataset after normalizing faces and cropping with 0.25 margin.

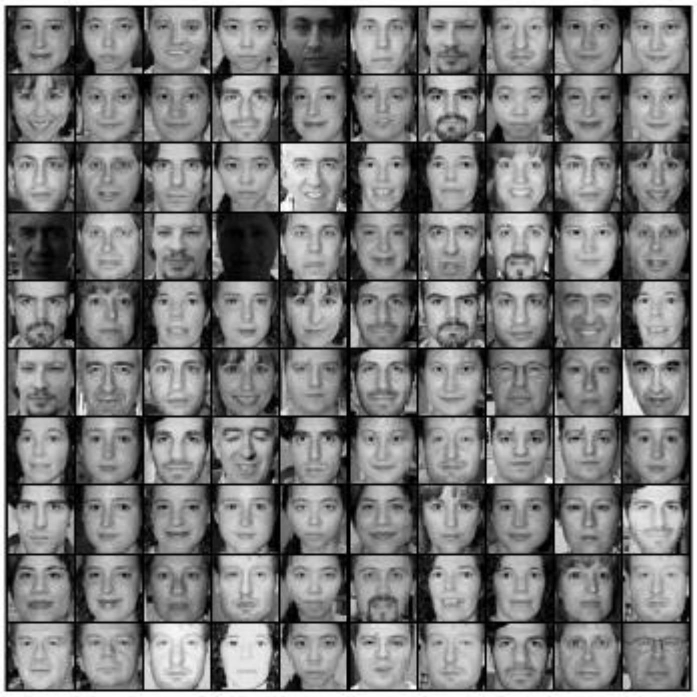


Figure 4.2: Sample images from Caltech dataset after normalizing faces and cropping with [0.3] margin.

4.2. TOOLS AND APPLICATION

4.2.1. Matlab

MATLAB [53] is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. Toolboxes are MATLAB application-specific functions and each is designed to deal with a specific issue. In this study the version 2015b of MATLAB software have been used to implement the proposed system. Image processing and neural networks and other toolboxes that come with MATLAB were also exploited.

4.2.2. Cross-Validation Technique

Cross validation is an evaluation method used to select the best model which is expected to perform better while dealing with unseen data. The idea is to exclude some data randomly while training and then use this data to evaluate the performance of the model. This method is often used to prevent overfitting, especially when data is small.

There are several approaches to apply cross validation including holdout method, Leave-one-out and K-fold. The latter was used in this research to train both Autoencoders and classifiers.

In k-fold cross validation the data is broken into *k* subsets, one of which is for testing while the rest is for training. Training is repeated as the number of subsets *k*; each time one subset is used for testing without repetition. The final error is calculated as the average of all trained k's errors (Figure 4.3).

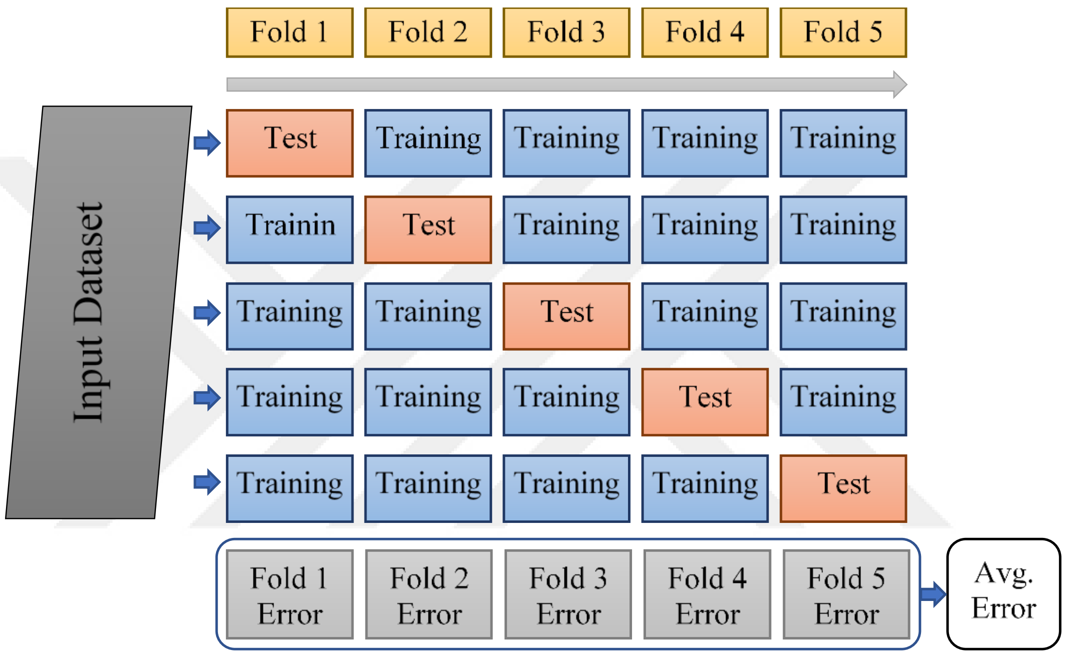


Figure 4.3: Cross validation model with 5-folds.

4.2.3. Performance Metrics

In this study, many efficiency measures were used to evaluate the result. This is a brief definition of the performance metrics used in the study:

- Confusion Matrix: Is a matrix that shows the relationship between the predicted results and the actual results in detail (Figure 4.4).
- True Positive (TP): The predicted value matches the actual (both are positive).
- True Negative (TN): The predicted value matches the actual (both are negative).
- False Positive (FP): The predicted value (positive) does not match the actual value (negative).
- False Negative (FN): The predicted value (negative) does not match the actual value (positive).

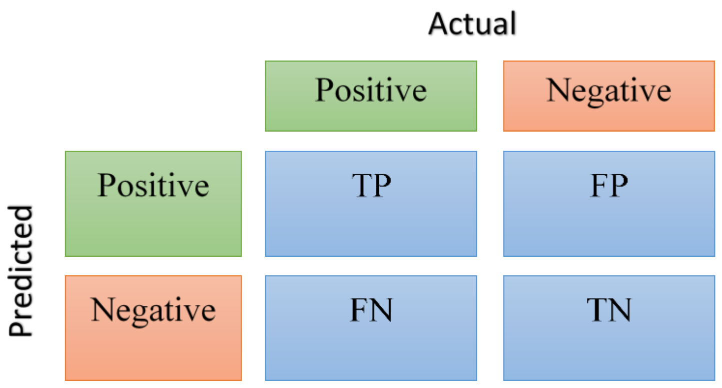


Figure 4.4: Confusion Matrix layout.

- Accuracy: The accuracy is the ratio between correct predictions and total input size.

$$Accuracy = \frac{\#\ of\ correct\ answers}{\#\ of\ input\ samples} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN} + \mathrm{TN}}$$

- Precision: Given all the predicted labels (for a given class X), how many instances were correctly predicted.

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

- Recall: For all instances that actually have a label X, how many of these were correctly labeled.

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- Sensitivity (also called the true positive rate) measures the proportion of positives that are correctly identified as positive (equal to Recall).

$$\mathrm{Sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

- Specificity (also called the true negative rate) measures the proportion of negatives that are correctly identified as negative.

$$\mathrm{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

- F-measure: The F1 score can be interpreted as a weighted average of the precision and recall.

$$\mathrm{F1\ Score} = 2 * \frac{\mathrm{Recall} * \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}}$$

- Kappa statistics:

$$\text{Kappa} = \frac{\text{Observed accuracy} - \text{Expected accuracy}}{1 - \text{expected accuracy}}$$

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

$$\text{Expected Accuracy} = \frac{(\text{TN} + \text{FP}) * (\text{TN} + \text{FN}) + (\text{FN} + \text{TP}) * (\text{FP} + \text{TP})}{(\text{TP} + \text{FP} + \text{FN} + \text{TN})\hat{}2}$$

4.3. FACE DETECTION RESULT

The performance of the integrated face detector applied to various faces databases is shown in the Table 4.1. However, in this study the result of recognition does not depend on the result of detection, because the detection errors were excluded or corrected manually. Figure 4.5 visualizes some examples of face detection and landmarks localization results from different datasets.

Table 4.1: The performance of the integrated face detector applied to various faces databases, F#: The number of faces in the dataset.

Dataset	**F#**	**TP**	**FN**	**FP**	**TN**[1]	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Yale	165	165	0	0	0	100.0%	100.0%	100.0%	100.0%
ORL	400	391	9	0	0	97.75%	100.0%	97.75%	98.86%
Caltech	450	447	3	74	0	89.94%	85.80%	99.33%	92.07%
PubFig	750	750	0	0	0	100.0%	100.0%	100.0%	100.0%

[1] TN: equal zero because all images in the databases contain faces.

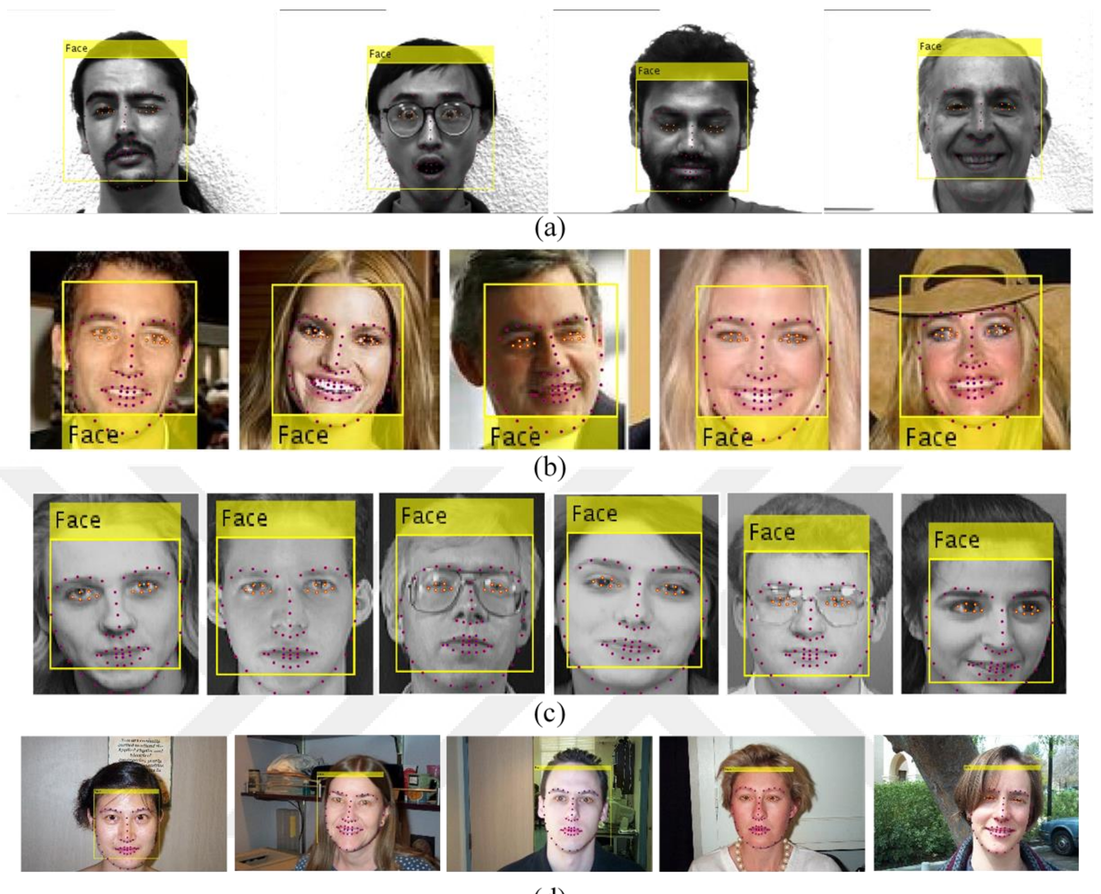


Figure 4.5: Face detection and landmarks localization results from different datasets; (a) Yale, (b) PubFig15, (c) ORL and (d) Caltech.

4.4. FACE RECOGNITION RESULTS

The efficiency of the Deep Stacked Denoising Sparse Autoencoder (DSDSA) system was measured using the four datasets mentioned above using the known (L-BFGS) for optimization. Accuracy is the ratio of faces that are correctly classified to the total number of faces in test dataset.

4.4.1. Parameters Settings

Following the main settings which used while training (DSDSA):

- The input image size is 32×32 pixels. Thus, input dimensionality is 1024.
- Number of hidden layers and nodes in each layer is a hyperparameter varies from database to another; however, experiments show that using two hidden layers has given a sufficiently

good performance considering the economy in the time complexity. However, hidden units have different efficiency with different datasets, the next part shows the effect of these differences.

- The corruption levels for denoising 0.1, 0.5 and other options were tested too.
- Regularization parameter (L2 normalization) [λ] was set to be 3e-4.
- Activation functions: sigmoid activation function was used for both input and output units in pre-training autoencoders, while softmax was used for output activation while finetuning.
- Weights initialization: Weights should be initialized randomly, it is important that they are not given equal or zeros values, otherwise the same activation will be learned, which is not useful. Common method is to sample a uniform (−r, r) where $r = 4\sqrt{6/(U_{in} + U_{out})}$ and U_{in}, U_{out} are the number of inputs/outputs units. Bias are initialized with zeros.

4.4.2. The Best Recorded Results

Table 4.2 shows the best recognition performance on different faces datasets and the number of neurons that gives the best results, using Softmax and SVM classifiers.

Table 4.2: The best recorded recognition accuracy of the Yale, ORL, Caltech and Pubfig15 datasets.

Dataset	Network k-folds	Hidden Layers and Neurons	Softmax Accuracy	SVM Accuracy
Yale	10	[121,121]	0.9824	0.9824
ORL	5	[529,529]	0.9800	0.9800
Caltech	5	[121,121]	0.9775	0.9820
PubFig15	10	[121,121]	0.9507	0.9547

Figure 4.6 shows the recognition loss (error) over iterations of the proposed (DSDSA) on varying datasets. From the shown result the effect of the dataset size is clear.

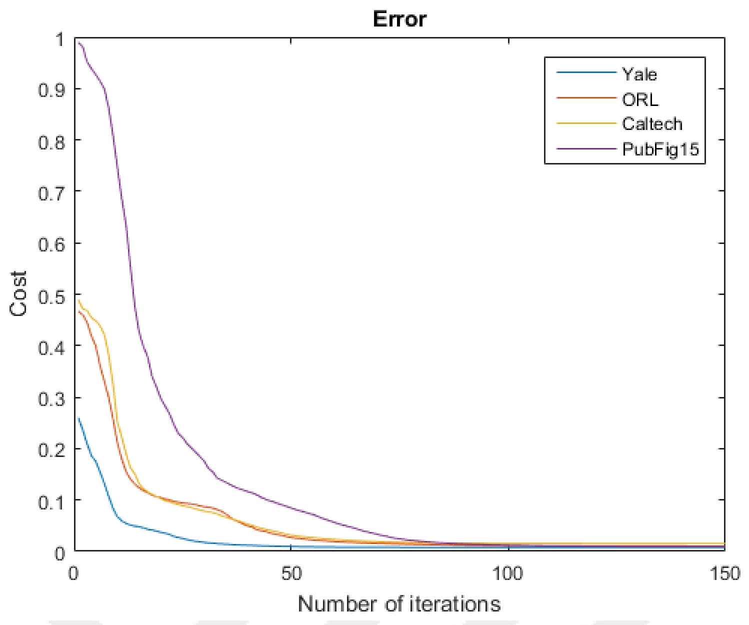


Figure 4.6: The recognition loss of the proposed (DSDSA) on different datasets after 150 iterations.

Table 4.3 and Table 4.4 show the performance metrics results for all datasets using Softmax classifier and SVM classifier respectively.

Table 4.3: Performance metrics results using Softmax classifier.

Dataset	Accuracy	Sensitivity	Specificity	Precision	Recall	F_1 Score	Kappa
Yale	0.9824	0.9833	0.9988	0.9800	0.9833	0.9800	0.8668
ORL	0.9800	0.9800	0.9995	0.9858	0.9800	0.9788	0.9500
Caltech	0.9775	0.9288	0.9991	0.9183	0.9288	0.9224	0.9231
PubFig15	0.9507	0.9507	0.9965	0.9586	0.9507	0.9500	0.8671

Table 4.4: Performance metrics results using SVM classifier.

Dataset	Accuracy	Sensitivity	Specificity	Precision	Recall	F_1 Score	Kappa
Yale	0.9824	0.9833	0.9988	0.9800	0.9833	0.9800	0.8668
ORL	0.9800	0.9800	0.9995	0.9858	0.9800	0.9788	0.9500
Caltech	0.9820	0.9442	0.9993	0.9340	0.9442	0.9376	0.9231
PubFig15	0.9547	0.9547	0.9968	0.9622	0.9547	0.9538	0.8670

4.4.3. Comparison

The proposed (DSDSA) was compared against the best reported results evaluated in the same datasets; the deep sparse autoencoder by Zhang et al. [44] symbolized with (SAE), the deep autoencoder networks by Li et al. [45] symbolized with (DAE), spatial domain sparse representation with autoencoders by Biswas et al. [54] symbolized with (SDSRA), laplacian faces by He et al. [55], principal component neural network symbolized with (PCNN) by Sudha et al. [56] and diffusion wavelet by Gudivada et al. [57]. The experimental comparison results are listed in Table 4.5. In Figure 4.7 a comparative analysis of different approaches.

Table 4.5: Comparison results between the proposed method and other methods.

Method	**Yale**	**ORL**
DSDSA	98.24%	98.00%
SAE[1]	92.00%	94.38 %
DAE	--	97.50%
SDSRA	--	91.50%
Laplacian Faces	88.70%	91.70%
PCNN	85.00%	--
Diffusion wavelet	87.56%	93.95%

SAE achieved its result using two hidden layers with [200,100] hidden units respectively and input size with 1024, while DAE used 784 input size with two hidden layers contained [800-800] hidden units for each. The proposed (DSDSA) achieved the mentioned results with input size 1024 and two hidden layers with [121,121] hidden units for Yale and [529,529] hidden units for ORL dataset.

[1] The result of fine tuning the top network as was done in this research.

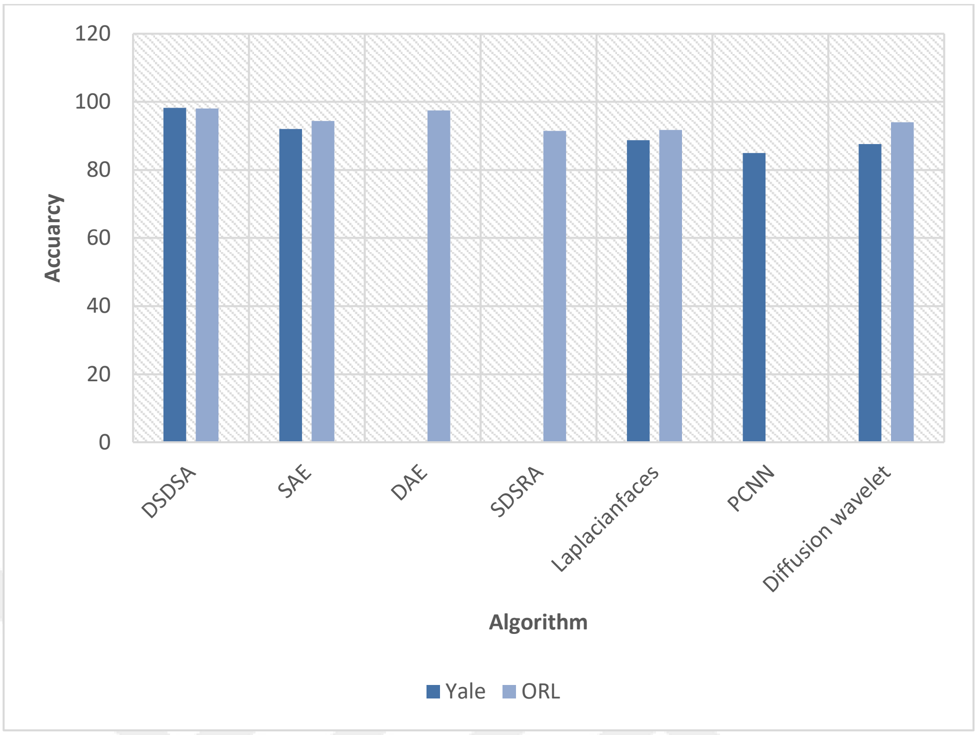


Figure 4.7: Comparative analysis of different methods on Yale (dark) and ORL (light).

4.4.4. Visualization

Figure 4.8 shows the learnt filter after pretraining the first denoising sparse autoencoder without corruption and with a corruption degree of (0.3). The reconstruction result of stacked autoencoder is shown in the Figure 4.9.

In machine learning tasks, it is common to visualize the result using the confusion matrix which explained in the section 4.2.3, Figure 4.10 shows the confusion matrix for PubFig15 dataset after 150 iterations with two hidden layers, [121,121] neurons, 0.1 corruption level and 0.5 for sparsity parameter. Figure 4.11 shows the confusion matrix of Yale dataset, trained using two hidden layers, [121,121] hidden units, 0.3 sparsity target, and 0.5 for corruption level.

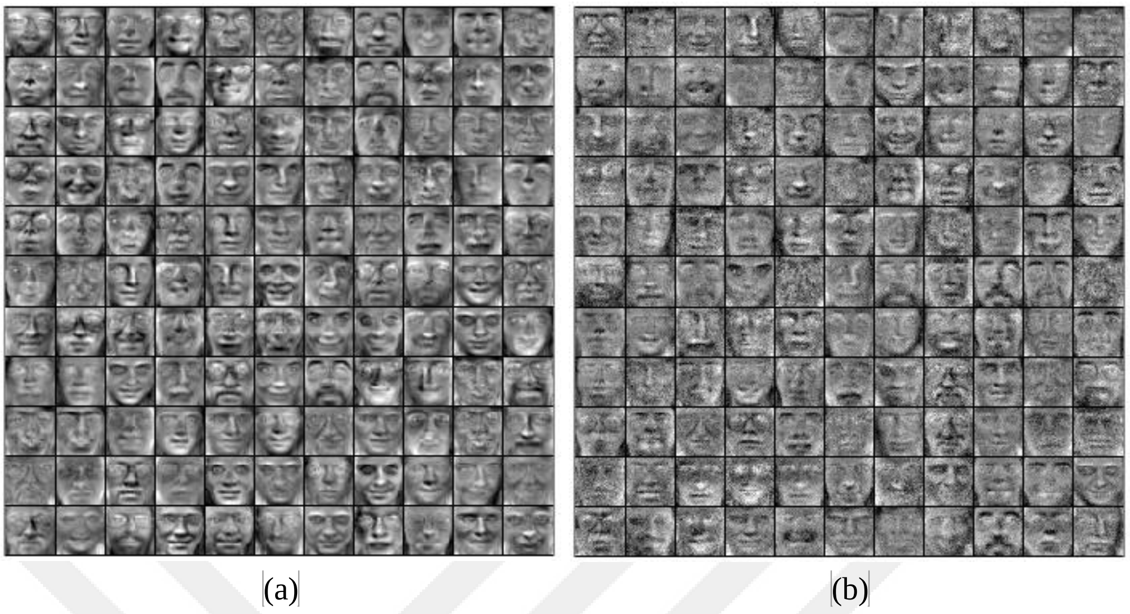


Figure 4.8: The learnt filters after pre-training the first layer/autoencoder in ORL with corruption level (a) 0.0 and (b) 0.3.

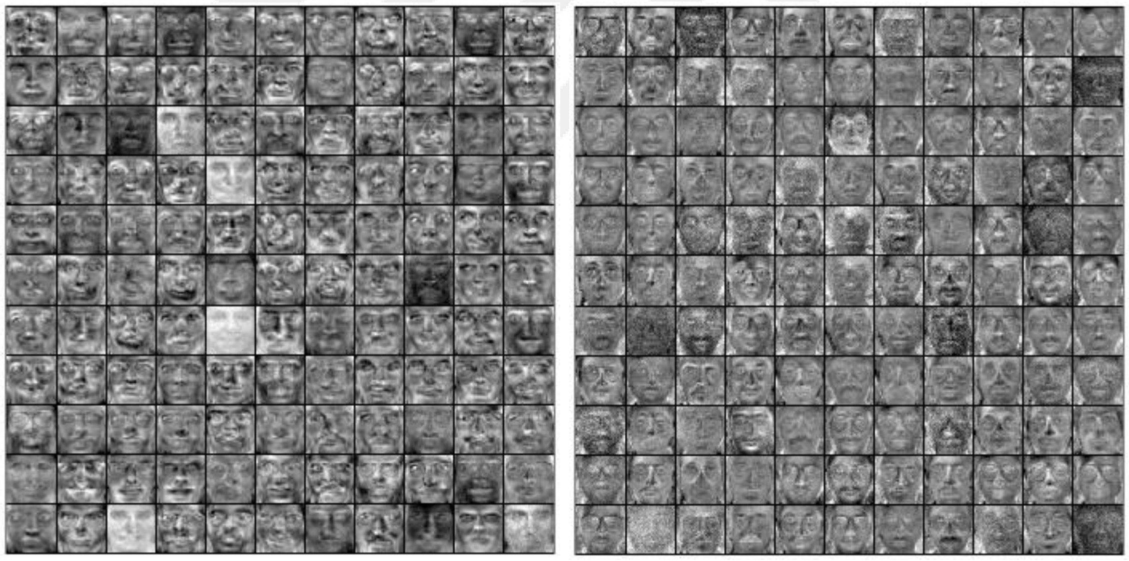


Figure 4.9: The stacked autoencoder’s reconstruction results (ORL -left, Yale -right)

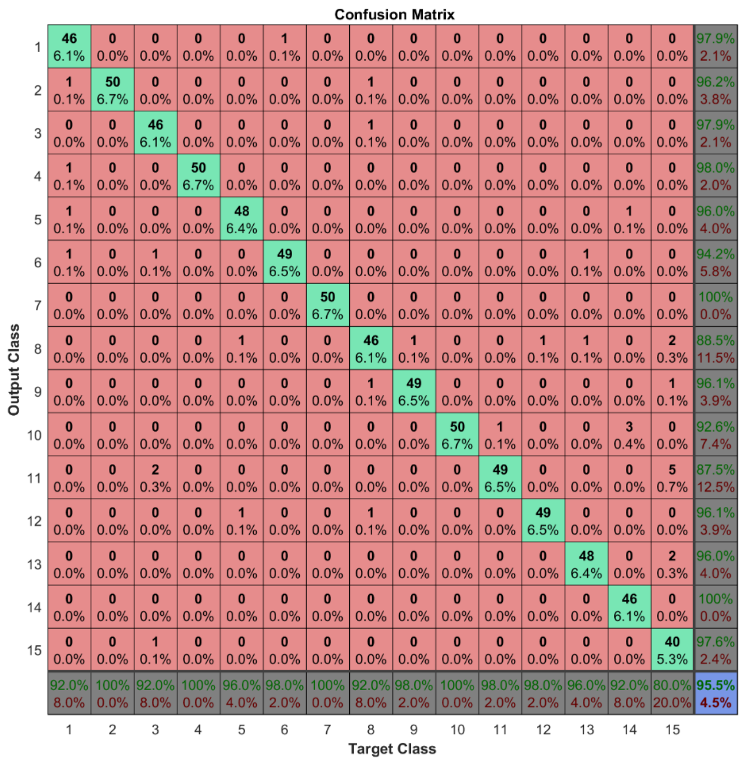


Figure 4.10: Confusion matrix for PubFig15 dataset.

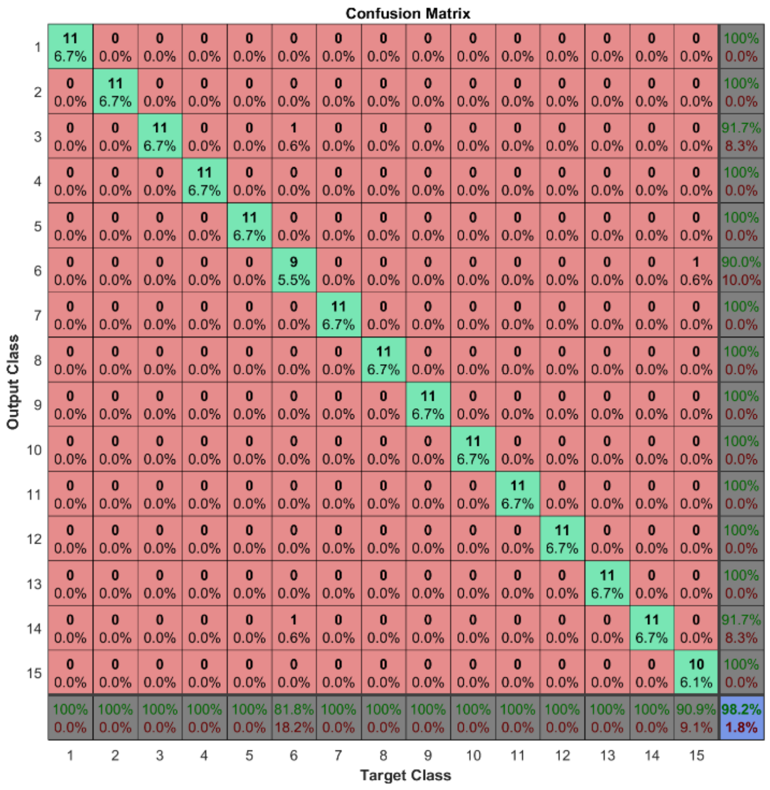


Figure 4.11: Confusion matrix of Yale dataset.

4.5. THE IMPACT OF PARAMETERS

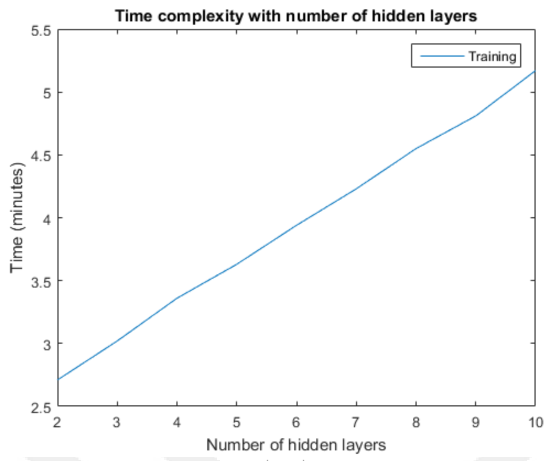
Parameters play a key role in determining evaluation result, different settings of important parameters have been experimentally evaluated in ‘PubFig15’ to show the impact of parameters. It should be noted that the best results are not the aim here, but only to note the effect of these parameters.

4.5.1. Hidden Layer Size (Depth)

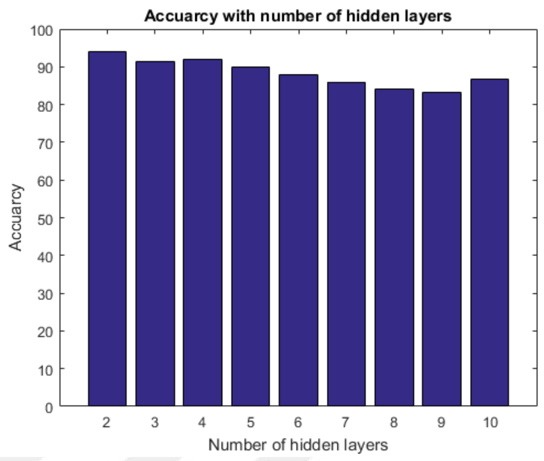
Table 4.6 and Figure 4.12 explore the effect of the number of hidden layers on performance, here the number of hidden units will not change for all hidden layers. More hidden layers mean deeper network, this will cost more training time, experiments indicate that more hidden layers also need more iterations to get better results. The results in Table 4.6 was taken using ‘PubFig15’ dataset, with 121 neurons for each hidden layer, [Corruption level: 0.5, sparsity: 0.5 and Iterations: 200].

Table 4.6: The effect of the number of hidden layers on time and accuracy.

Hidden layers	Softmax Acc.	SVM Acc.	Time (min.)
2x [121]	94.00%	94.00%	2.71
3x [121]	91.33%	91.33%	3.02
4x [121]	92.00%	92.00%	3.36
5x [121]	90.00%	90.00%	3.63
6x [121]	88.67%	88.00%	3.94
7x [121]	84.00%	86.00%	4.23
8x [121]	82.00%	84.00%	4.55
9x [121]	82.00%	83.33%	4.81
10x [121]	87.33%	86.67%	5.17



(a)



(b)

Figure 4.12: The effect of the number of hidden layers on (a) Time and (b) Accuracy.

4.5.2. Hidden Units Size (Neurons)

Table 4.7 explores the performance of the (DSDSA) with various number of hidden units' sizes. The result improves that the number of neurons has an important role on the network architecture with the note that there is no steady or general rule determines the suitable number of hidden units for specific task. The appropriate number is reached through the experiment. The result in Table 4.7 was taken using 'PubFig15' dataset with: two hidden layers, corruption level: 0.1, sparsity: 0.5 and 150 iterations.

Table 4.7: The effect of hidden units on performance.

Hidden units' sizes	**Softmax Acc.**	**SVM Acc.**	**Time (min.)**
[64, 64]	91.33%	92.67%	0.89
[121,121]	94.67%	95.33%	1.66
[200,200]	96.00%	94.67%	2.86
[441,441]	94.67%	93.33%	7.35
[667,667]	92.00%	91.33%	13.27

Figure 4.13 demonstrates the relation between the number of hidden units and the cost of time. Obviously, the higher number of neurons, the greater cost of time.

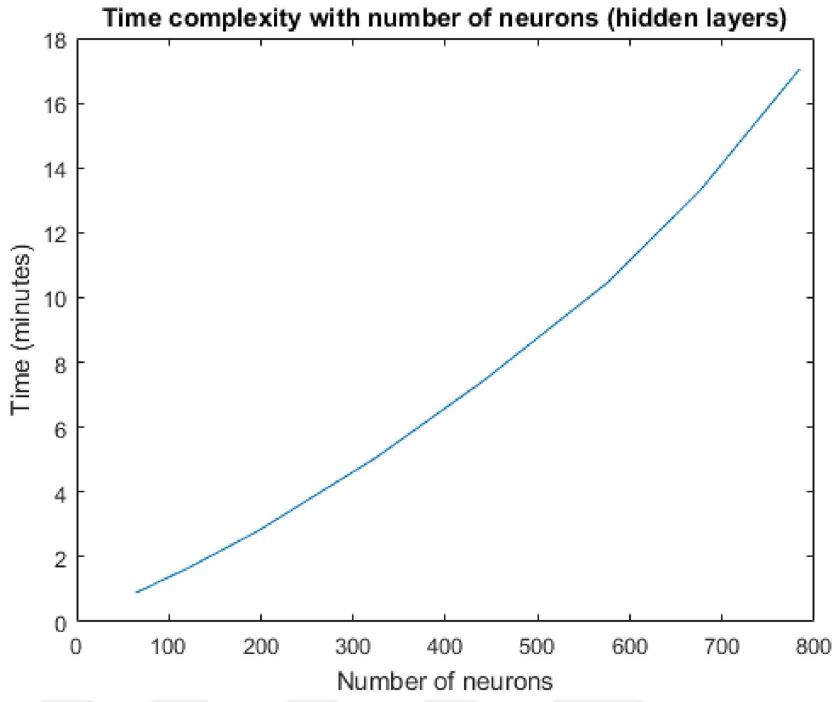


Figure 4.13: Time complexity with number of neurons

4.5.3. Denoising (Corruption Level)

Denoising term is determined by corruption level, which is usually estimated between 0 and 0.5. Table 4.8 shows the impacts of denoising level on the performance. The result shows that the autoencoder has the ability to restore a good reconstruction from corrupted input version even with high corruption level. The result in Table 4.8 was taken using 'PubFig15' dataset with two hidden layers, 121 neurons for each layer, sparsity: 0.5 and 150 iterations.

Table 4.8: The effect of denoising level on performance

Corruption levels	**Softmax Acc.**	**SVM Acc.**
[0.0]	93.33%	93.33%
[0.1]	95.33%	95.33%
[0.2]	92.67%	92.00%
[0.3]	94.67%	95.33%
[0.5]	94.67%	95.33%

4.5.4. Sparsity Target

Finally, the sparsity is determined by two parameters, the sparsity penalty (β) and the sparsity parameter. Table 4.9 shows the effect of changing the sparsity parameter while beta was chosen to be 3. The results show how the selection of the correct sparsity value plays an important role in the results. The results in Table 4.9 were taken using ‘PubFig15’dataset with: two hidden layers, 121 neurons for each layer, corruption level: 0.1 and 150 iterations.

Table 4.9: The effect of sparsity parameter on accuracy

Sparsity target	**Softmax Acc.**	**SVM Acc.**
[0.0001]	06.67%	06.67%
[0.01]	06.67%	08.00%
[0.1]	94.67%	94.00%
[0.2]	94.00%	93.33%
[0.5]	95.33%	95.33%
[0.7]	94.67%	94.00%

5. DISCUSSION AND CONCLUSION

5.1. DISSCUSION

This research aims to investigate the usage of deep learning method to solve a classification problem applied to a face recognition task. Both deep learning methods and face recognition task are still under investigation, the practical results of using deep learning methods, which have been applied to many tasks, show the ability of these methods to achieve results exceed the traditional methods.

Face recognition applications have been increasingly used in many areas, due to the accurate results obtained compared with other alternatives, as well as the ease of use in many systems and their low cost.

Although research on facial recognition has been ongoing for decades, the development of machine learning, especially the deep learning techniques, has brought it back to the fore by promising results. Deep learning allows the use of unlabeled data which is available much more widely than the labeled data, through unsupervised learning.

Challenges in the face recognition field related to images are taken or stored in the database in terms of accuracy, size, scene and illumination, as well as aspects related to the face within the image and this includes the effects of age, face pose angle, facial expression and the occlusion due to sunglasses, scarf or other.

In the other hand, the challenges in deep learning include large data, high dimensionality, hyperparameter tuning, overfitting issues and black-box nature of the network because of the high number of layers, neurons and joints.

Autoencoder has a special ability to handle unlabeled data where there is no target thus the autoencoder’s goal is to reconstruct the original input by learning the hidden representation of training data, this is a cumulative process done by several hidden layers.

Reconstructing the same input data may get to unwanted results such as learning identity function or overfitting to specific data, the proposed solutions to overcome these obstacles are to using sparsity and denoising. Sparsity is done by adding sparsity constraint on the hidden units while denoising is done by trying to reconstruct the input x from a noisy copy of it.

This pre-training process is done differently from regular methods by training every hidden layer as an autoencoder separately, where each autoencoder take the output from the former autoencoder as its input, then all these autoencoders are assembled into one stacked autoencoder. In classification tasks its common to add softmax layer to the top of the hidden layers, this layer is trained in supervised way. Then the stacked autoencoder is finetuned in supervised way too as long as a classification problem is handled, which is a supervised task.

In this research the proposed system was applied to a number of face databases and the results, as presented in the previous chapter, showed the ability of this system to achieve advanced results. The ability of the system to achieve positive results in different databases indicates its validity for generalization and that this type of algorithms may be a typical solution to many problems.

Experimental results show that the proposed Deep Stacked Denoising Sparse Autoencoder (DSDSA) has achieved satisfactory results on face recognition after examination on several databases which proves its ability to generalize, that also improve the usefulness of using sparsity and denoising to overcome the problem of overfitting and improve performance efficiency. The unsupervised pretraining allows the use of huge amounts of available unlabeled data is one of the additional pros of the proposed system. One key advantage of the proposed (DSDSA) is its ability to learn from fewer training data unlike existing systems that rely on a huge amount of data.

The results of the research also showed the importance of the hyperparameters tuning in the autoencoders, the usage of inappropriate parameters will inevitably lead to unsatisfactory results, as shown in the results section. Therefore, this parameter must be adjusted accurately, and its effect on the results should be observed constantly.

5.2. CONCLUSION

In this research a face recognition system based in deep stacked denoising autoencoder called (DSDSA) was proposed. The algorithm tries to extract robust features by unsupervised learning through deep autoencoders. To improve classification results the pretrained autoencoders were stacked and finetuned in supervised way. The experimental results show that the deep autoencoder is efficient to deal with pattern recognition tasks and achieves good results.

Many aspects such as denoising, sparsity, weights initialization and more details are still the subject of research and development area to reach more robust results.

5.3. RECOMMENDATIONS

Based on the positive results of the presented system in solving the Classification problems, it should be used and illustrated to solve other problems such as Regression and Clustering.

Parameters play an important role in system performance so there should be a way to adjust these parameters to get the best possible results.

The autoencoder structure allows using many alternatives to solve a partial problems inside. Searching these alternatives to find the best ones is recommended. For example, many activation functions could be used in encoding and decoding phases including Sigmoid, Tanh, ReLU, Binary Step and Softmax.

Regardless, deep learning systems are vital systems and they are subject to further research and development.

REFERENCES

[1] Bengio, Y., 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, *2*(1), pp.1-127.

[2] Taigman, Y., Yang, M., Ranzato, M.A. and Wolf, L., 2014. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1701-1708).

[3] Wilkin, J., 2017, Social Mapper Trustwave Holdings, Inc., https://github.com/SpiderLabs/social_mapper, [Visited in: Sep. 30, 2018, 20:00 GMT].

[4] Kowalczyk, L. and Tatina, D., 2017, https://pimeyes.com/en/, [Visited in: Sep. 30, 2018, 20:00 GMT].

[5] Big Brother Watch, 2018, *Face Off - The lawless growth of facial recognition in UK policing*, UK.

[6] Turk, M.A. and Pentland, A.P., 1991, June. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference* on (pp. 586-591). IEEE.

[7] Belhumeur, P.N., Hespanha, J.P. and Kriegman, D.J., 1997. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on pattern analysis and machine intelligence, 19(7), pp.711-720.

[8] Roweis, S.T. and Saul, L.K., 2000. Nonlinear dimensionality reduction by locally linear embedding. *science*, *290*(5500), pp.2323-2326.

[9] Schölkopf, B., Smola, A. and Müller, K.R., 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, *10*(5), pp.1299-1319.

[10] Mika, S., Ratsch, G., Weston, J., Scholkopf, B. and Mullers, K.R., 1999, August. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.* (pp. 41-48). Ieee.

[11] Ahonen, T., Hadid, A. and Pietikainen, M., 2006. Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, *28*(12), pp.2037-2041.

[12] Liu, C. and Wechsler, H., 2002. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image processing*, *11*(4), pp.467-476.

[13] Dalal, N. and Triggs, B., 2005, June. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886-893). IEEE.

[14] Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), pp.91-110.

[15] Vapnik, V., 1998. *Statistical learning theory*. *1998*. Wiley, New York.

[16] Cover, T. and Hart, P., 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory*, *13*(1), pp.21-27.

[17] Ho, J., Yang, M.H., Lim, J., Lee, K.C. and Kriegman, D., 2003, June. Clustering appearances of objects under varying illumination conditions. In *Computer vision and pattern recognition, 2003. Proceedings. 2003 IEEE computer society conference on* (Vol. 1, pp. I-I). IEEE.

[18] Guillaumin, M., Verbeek, J. and Schmid, C., 2009, September. Is that you? Metric learning approaches for face identification. In *Computer Vision, 2009 IEEE 12th international conference on*(pp. 498-505). IEEE.

[19] Huang, C., Zhu, S. and Yu, K., 2012. Large scale strongly supervised ensemble metric learning, with applications to face verification and retrieval. *arXiv preprint arXiv:1212.6094*.

[20] Cinbis, R.G., Verbeek, J. and Schmid, C., 2011, November. Unsupervised metric learning for face identification in TV video. In *Computer Vision (ICCV), 2011 IEEE International Conference on*(pp. 1559-1566). IEEE.

[21] Wright, J., Yang, A.Y., Ganesh, A., Sastry, S.S. and Ma, Y., 2009. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, *31*(2), pp.210-227.

[22] Zhang, L., Yang, M. and Feng, X., 2011, November. Sparse representation or collaborative representation: Which helps face recognition?. In *Computer vision (ICCV), 2011 IEEE international conference on* (pp. 471-478). IEEE.

[23] Gao, S., Tsang, I.W.H. and Chia, L.T., 2010, September. Kernel sparse representation for image classification and face recognition. In *European Conference on Computer Vision* (pp. 1-14). Springer, Berlin, Heidelberg.

[24] Yang, M. and Zhang, L., 2010, September. Gabor feature based sparse representation for face recognition with gabor occlusion dictionary. In *European conference on computer vision* (pp. 448-461). Springer, Berlin, Heidelberg.

[25] Lu, C.Y., Min, H., Gui, J., Zhu, L. and Lei, Y.K., 2013. Face recognition via weighted sparse representation. *Journal of Visual Communication and Image Representation*, *24*(2), pp.111-116.

[26] Min, R. and Dugelay, J.L., 2011, July. Improved combination of LBP and sparse representation based classification (SRC) for face recognition. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on* (pp. 1-6). IEEE.

[27] Yin, J., Liu, Z., Jin, Z. and Yang, W., 2012. Kernel sparse representation based classification. *Neurocomputing*, *77*(1), pp.120-128.

[28] Huang, G.B., Lee, H. and Learned-Miller, E., 2012, June. Learning hierarchical representations for face verification with convolutional deep belief networks. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (pp. 2518-2525). IEEE.

[29] Sun, Y., Wang, X. and Tang, X., 2013, December. Hybrid deep learning for face verification. In *Computer Vision (ICCV), 2013 IEEE International Conference on* (pp. 1489-1496). IEEE.

[30] Sun, Y., Wang, X. and Tang, X., 2014. Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1891-1898).

[31] Sun, Y., Chen, Y., Wang, X. and Tang, X., 2014. Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems* (pp. 1988-1996).

[32] Sun, Y., Wang, X. and Tang, X., 2014. Deeply learned face representations are sparse, selective, and robust. *arXiv preprint arXiv:1412.1265*.

[33] Sun, Y., Liang, D., Wang, X. and Tang, X., 2015. Deepid3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*.

[34] Chopra, S., Hadsell, R. and LeCun, Y., 2005, June. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*(Vol. 1, pp. 539-546). IEEE.

[35] Huang, G.B., Lee, H. and Learned-Miller, E., 2012, June. Learning hierarchical representations for face verification with convolutional deep belief networks. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on* (pp. 2518-2525). IEEE.

[36] Zhu, Z., Luo, P., Wang, X. and Tang, X., 2014. Recover canonical-view faces in the wild with deep neural networks. *arXiv preprint arXiv:1404.3543*.

[37] Schroff, F., Kalenichenko, D. and Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

[38] Parkhi, O.M., Vedaldi, A. and Zisserman, A., 2015, September. Deep face recognition. In *BMVC* (Vol. 1, No. 3, p. 6).

[39] Hinton, G.E. and Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. *science*, *313*(5786), pp.504-507.

[40] Kan, M., Shan, S., Chang, H. and Chen, X., 2014. Stacked progressive auto-encoders (spae) for face recognition across poses. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1883-1890).

[41] Zhang, J., Shan, S., Kan, M. and Chen, X., 2014, September. Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment. In *European Conference on Computer Vision* (pp. 1-16). Springer, Cham.

[42] Gao, S., Zhang, Y., Jia, K., Lu, J. and Zhang, Y., 2015. Single sample face recognition via learning deep supervised autoencoders. IEEE Transactions on Information Forensics and Security, 10(10), pp.2108-2118.

[43] Ding, C. and Tao, D., 2015. Robust face recognition via multimodal deep face representation. *IEEE Transactions on Multimedia*, *17*(11), pp.2049-2058.

[44] Zhang, Z., Li, J. and Zhu, R., 2015, October. Deep neural network for face recognition based on sparse autoencoder. In *Image and Signal Processing (CISP), 2015 8th International Congress on* (pp. 594-598). IEEE.

[45] Li, F., Gao, X. and Wang, L., 2017. Face Recognition Based on Deep Autoencoder Networks with Dropout.

[46] Viola, P. and Jones, M.J., (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. *IEEE Conf Comput Vis Pattern Recognit*. 1. I-511. 10.1109/CVPR.2001.990517.

[47] Yu, X., Huang, J., Zhang, S., Yan, W. and Metaxas, D.N., 2013, December. Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model. In *Computer Vision (ICCV), 2013 IEEE International Conference on* (pp. 1944-1951). IEEE.

[48] Baltrusaitis, T., Robinson, P. and Morency, L.P., 2013, December. Constrained local neural fields for robust facial landmark detection in the wild. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on* (pp. 354-361). IEEE.

[49] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y. and Manzagol, P.A., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, *11*(Dec), pp.3371-3408.

[50] Hinton, G.E., Osindero, S. and Teh, Y.W., 2006. A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), pp.1527-1554.

[51] Ng, A., Ngiam, J., Foo, C.Y., Mai, Y., Suen, C., Coates, A., Maas, A., Hannun, A., Huval, B., Wang, T. and Tandon, S., 2013. Unsupervised feature learning and deep learning.

[52] Kumar, N., Berg, A.C., Belhumeur, P.N. and Nayar, S.K., 2009, September. Attribute and simile classifiers for face verification. In *Computer Vision, 2009 IEEE 12th International Conference on* (pp. 365-372). IEEE.
[53] Release, M.A.T.L.A.B., 2015. The MathWorks. Inc., Natick, MA, United States.
[54] Biswas, S., Sil, J. and Maity, S.P., 2018. On prediction error compressive sensing image reconstruction for face recognition. *Computers & Electrical Engineering*, *70*, pp.722-735.
[55] He, X., Yan, S., Hu, Y., Niyogi, P. and Zhang, H.J., 2005. Face recognition using laplacianfaces. *IEEE transactions on pattern analysis and machine intelligence*, *27*(3), pp.328-340.
[56] Sudha, N., Mohan, A.R. and Meher, P.K., 2011. A self-configurable systolic architecture for face recognition system based on principal component neural network. *IEEE transactions on circuits and systems for video technology*, *21*(8), pp.1071-1084.
[57] Gudivada, S. and Bors, A.G., 2012, August. Face recognition using ortho-diffusion bases. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European* (pp. 1578-1582). IEEE.

CURRICULUM VITAE

Personal Information	
Name Surname	Ahmet ŞİMŞEK
Place of Birth	Amman
Date of Birth	1982
Nationality	[✓ T.C.] ❒ Other:
Phone Number	
Email	
Web Page	

Educational Information	
B. Sc.	
University	Mutah University
Faculty	Science Faculty
Department	Computer Science
Graduation Year	2005

M. Sc.	
University	Istanbul University-Cerrahpaşa
Institute	Institute of Graduate Studies in Science and Engineering
Department	Department of Computer Engineering
Programme	Computer Engineering Programme

Publications

GÖRGEL P., AlRaad A., 2018, A DEEP LEARNING BASED FACE RECOGNITION SYSTEM, 1st International Symposium on Graduate Research in Science (ISGRS 2018), October 4-6, 2018, İstanbul.