OPTIMIZED FACIAL KEYPOINT DETECTION TO CREATE SELFIE FILTER

PROJECT REPORT

Submitted in fulfilment for the JComponent of

Soft Computing (SWE1011)

CAL Course

in

M.Tech. – Software Engineering

by

Satyam Padhi 19MIS0384

Donta Chetan 19MIS0329

Under the guidance of

Dr. S. Hemalatha

SITE

School of Information Technology and Engineering

Fall Semester 2021-22

TABLE OF CONTENTS

ABSTRACT

We will begin with convolutional neural network. (CNN) There is no doubt that CNN is the best solution to this problem with such a low RMSE. It has been proved that CNN performs well to image recognition both from theory and practice. The advantages include feature extraction and soft weight sharing, which is the unique properties of CNN. It is these characters that make CNN such a perfect approach. However, the disadvantage still need to be mentioned since this cannot be felt from these paper unless to run our code on computers. Very long time will be taken though the code is run on the server. It raises the problem of tuning because it is time consuming. Thence, we have reasons to believe that better RMSE can be achieved if suitable parameters are applied. Secondly, Neural network doesn't get the RMSE as well as our expectation. Compared with CNN, neural network itself is not a specific method for image recognition so this would not be surprising. However, we originally expect that neural network may perform better can Regression methods. Now taking a look back, neural network's failure may be attributed to overfitting and tuning. The method of 'Dropout' can avoid overfitting but there is no theoretical guidance to the setting of parameters. Beside its high RMSE compared with CNN, it is still time consuming to train neural network. Overall, we do not recommend neural network in face recognition experiment unless perfect parameters can be found. These perfect parameters can be found using PCA when retaining somewhat from 95-97% variance. So, the main objective of this paper is to check whether PCA is suitable for improving CNN model or not and when we get a optimized model, we using the predicted key points on a person's face create a selfie filter something like sunglasses or fake moustache to display the use of the facial key points. Detecting facial key-points is a critical element in face recognition, facial filter attachment. However, there is difficulty to catch key-points on the face due to complex influences from original images, and there is no guidance to suitable algorithms. In this paper, we study how dimensionality reduction affects a predictive model to locate key- points. Finally, our conclusion after comparison of both algorithms based on the parameters such as Model Metric (mean absolute error) and Inference speed. The assumed outcome should be the increase in inference speed and better fitting of the data to the same model on applying dimensionality reduction i.e. PCA. This facial key points detection can be used to create facial filters used in various camera apps. The objective of this project is to make benchmarks about how much dimensionality reduction strategies main PCA (Principal Component Analysis) helps in decreasing over- fitting for recognizing facial key-points. If the idea of deep learning will be used, such as neural network and cascaded neural network then the results of these structures will be significantly better than state of-the-art methods, like feature extraction and dimension reduction algorithms.

1. INTRODUCTION

OpenCv is often used in practice with other machine learning and deep learning libraries to produce interesting results. Employing Convolutional Neural Networks (CNN) in Keras along with OpenCv — we built a couple of selfie filters. Facial key points can be used in a variety of machine learning applications from face and emotion recognition to commercial applications like the image filters popularized by Snapchat. Each data point in the dataset contains space-separated pixel values of the images in sequential order and the last 30 values of the data point represent 15 pairs of coordinates of the key points on the faces. So, we just have to train a CNN model to solve a classic deep learning regression problem. Detecting facial key-points is a critical element in face recognition, facial filter attachment. However, there is difficulty to catch key-points on the face due to complex influences from original images, and there is no guidance to suitable algorithms. In this paper, we study how dimensionality reduction affects a predictive model to locate key- points. Finally, our conclusion after comparison of both algorithms based on the parameters such as Model Metric (mean absolute error) and Inference speed. The increase in inference speed and better fitting of the data to the same model on applying dimensionality reduction i.e., PCA. This facial key point's detection can be used to create facial filters used in various camera apps.

• Facial feature detection improves face recognition

• Male/Female Distinction

• Facial Expression Distinction

• Head pose estimation

• Face Morphing

• Virtual Makeover

• Face Replacement

Nowadays, facial key points detection has become a very popular topic and its applications include Snapchat, how old are you, have attracted a large number of users. The objective of facial key points detection is to find the facial key points in a given face, which is very challenging due to very different facial features from person to person. The idea of deep learning has been applied to this problem, such as neural network and cascaded neural network. And the results of these structures are significantly better than state of-the-art methods, like feature extraction and dimension reduction algorithms.

With the fast development in computer vision area, more and more research work and industry applications are focused on facial key points detection. Detecting key points in a given face image would act as a fundamental part for many applications,

including facial expression classification, facial alignment, tracking faces in videos and also applications for medical diagnosis. Thus, how to detect facial key points both fast and accurately to use it as a pre-processing procedure has become a big challenge.

Biometric recognition or biometrics is an automatic recognition system, based on physiological and/or behavioural characteristics of an individual. Biometrics makes it possible to confirm, establish an individual's identity based on "Who he/she is", instead of "what he/she possesses" (ID card) or "what he/she remembers" (password). A person's biometric characteristics are unique. Such keys are impossible to copy and reproduce exactly.

Students' attendance in the classroom is very important task and if taken manually wastes a lot of time. There are many automatic methods available for this purpose, i.e., biometric attendance. All these methods also waste time because students have to make a queue to touch their thumb on the scanning device. This work describes the efficient algorithm that automatically marks the attendance without human intervention based on Embedded Linux. This attendance is recorded by using a camera attached in front of classroom that is continuously capturing images of students, detect the faces in images and compare the detected faces with the database and mark the attendance.

The principal focus of this project is to make benchmarks about how much dimensionality reduction strategies main PCA (Principal Component Analysis) helps in decreasing overfitting for recognizing facial key-points.

The method that we propose is using set of trainable image data with their facial key points in the form of (x,y) coordinates and pre-process the data using dimensionality reduction techniques mainly PCA(Principal Component Analysis) to reduce the noise of Image thereby still retaining maximum possible variance around 95-98%. Then passing these pre-processed images to a Convolution neural network model and check how the training and inference differs in PCA pre-processed images and nonpre-processed images possible results could be that the training time of model will decrease considerably and may give better results on test dataset. Since dimensions of an image is decreased, it will also increase the inference speed.

The data has been collected from one of Kaggle's popular dataset on facial key points https://www.kaggle.com/c/facial-keypoints-detection/data.

In this project, the main language we use is Python. In addition, we use the following packages to achieve our algorithm:

Front-End: Python

Back-End: https://www.kaggle.com/c/facial-keypoints-detection/data & CNN MODEL DEVELOPED BY TEAM.

Along with that the following convolution neural network models were trained on Google Collaboratory which provides Jupyter Notebook like interface and kernels connected in backend to Google cloud Platform's ML engines.

We used different packages like:

Keras:

Keras is the high-level API of TensorFlow 2: an approachable, highly-productive interface for solving machine learning problems, with a focus on modern deep learning. It provides essential abstractions and building blocks for developing and shipping machine learning solutions with high iteration velocity.

Keras was initially developed as part of the research effort of project ONEIROS (Open-ended NeuroElectronic Intelligent Robot Operating System).

OpenCV:

OpenCV was started at Intel in 1999 by Gary Bradsky and the first release came out in 2000. Vadim Pisarevsky joined Gary Bradsky to manage Intel's Russian software OpenCV team. In 2005, OpenCV was used on Stanley, the vehicle who won 2005 DARPA Grand Challenge. Later its active development continued under the support of Willow Garage, with Gary Bradsky and Vadim Pisarevsky leading the project. Right now, OpenCV supports a lot of algorithms related to Computer Vision and Machine Learning and it is expanding day-by-day.

Currently OpenCV supports a wide variety of programming languages like C++, Python, Java etc and is available on different platforms including Windows, Linux, OS X, Android, iOS etc. Also, interfaces based on CUDA and OpenCL are also under active development for high-speed GPU operations.

OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

2. DATASET SPECIFICATION

The objective of this task is to predict keypoint positions on face images. This can be used as a building block in several applications, such as:

• tracking faces in images and video
• analysing facial expressions
• detecting dysmorphic facial signs for medical diagnosis
• biometrics / face recognition

Detecing facial keypoints is a very challenging problem. Facial features vary greatly from one individual to another, and even for a single individual, there is a large amount of variation due to 3D pose, size, position, viewing angle, and illumination conditions. Computer vision research has come a long way in addressing these difficulties, but there remain many opportunities for improvement.

Each predicted keypoint is specified by an (x,y) real-valued pair in the space of pixel indices. There are 15 keypoints, which represent the following elements of the face:
left_eye_center, right_eye_center, left_eye_inner_corner, left_eye_outer_corner, right_eye_inner_corner, right_eye_outer_corner, left_eyebrow_inner_end, left_eyebrow_outer_end, right_eyebrow_inner_end, right_eyebrow_outer_end, nose_tip, mouth_left_corner, mouth_right_corner, mouth_center_top_lip, mouth_center_bottom_lip. Left and right here refers to the point of view of the subject.

In some examples, some of the target keypoint positions are misssing (encoded as missing entries in the csv, i.e., with nothing between two commas).

The input image is given in the last field of the data files, and consists of a list of pixels (ordered by row), as integers in (0,255). The images are 96x96 pixels.

Data files

training.csv: list of training 7049 images. Each row contains the (x,y) coordinates for 15 keypoints, and image data as row-ordered list of pixels.

test.csv: list of 1783 test images. Each row contains ImageId and image data as rowordered list of pixels submissionFileFormat.csv: list of 27124 keypoints to predict. Each row contains a RowId, ImageId, FeatureName, Location. FeatureName are "left_eye_center_x," "right_eyebrow_outer_end_y," etc. Location is what you need to predict.

3. LITERATURE REVIEW

3.1. Review on Various Schemes

Face recognition is an action that humans perform routinely and effortlessly in our daily lives. The person identification for the face that appears in the input data is the face recognition process. Face recognition process is shown in

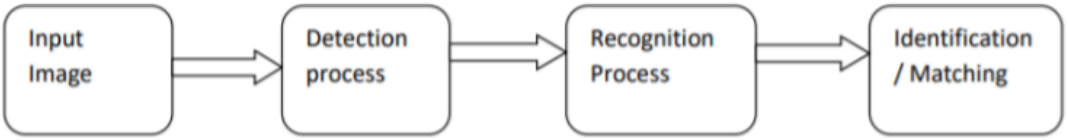


Fig. 1: Face Recognition Steps

Figure 1 below:

Holistic approaches to face recognition take into consideration global information from the given set of faces to perform face recognition and verification. The global information is primarily represented by a very small number of features which are directly derived from the pixels of facial images. These features are responsible to distinctly identify and represent the variations among the different facial images and hence uniquely identify the individual.

The Eigen face Approach:

The Eigen faces methods so called Eigenvector [4] or Principal Component Analysis [5] (PCA) methods are the general methods of face recognition. Faces can be easily reconstructed by considering only a small amount of information obtained using Eigen faces. These are nothing but principal components that divide the face into feature vectors in the form of covariance matrix. Then these vectors are used to calculate the variation among multiple faces. The faces are characterized by linear combination of highest Eigen values [6]. The low face recognition rate is due to the noise in background and Eigen features like eye, nose, mouth, cheeks, etc can be used instead of Eigen faces. This approach is less sensitive as compared to the Eigen face method. In this case the system attains 95% recognition rate. In short, Eigen face approach is most reliable, fast and efficient that endows invariance information also in the presence of varying lighting and scaling conditions. In recent developments, a face recognition technique which is based on Multi linear principal component analysis and locality preserving projection to improve the performance of face recognition system which uses MPCA for facial image pre-processing and LPP for extraction of facial features. Eigen face algorithm using principal component analysis (PCA) for reduce dimension to find vectors[7] have best value to distribute face image in input face space. This vector define subspace named face space; training set projected into face space to find set of weight that describe contribution of each vector in face space.

The Neural Networks Approach:

Neural Network [8] inspired by human brain composed of simple artificial neurons also known a perceptron's are connected to each other in multiple layers. Each perceptron [9] consist of mathematical function either a summation function or threshold function. This is a self-learning network which is trained and not explicitly programmed. In the case of face detection, neural network system examines each and every window (considerably small in size) to determine whether it consist of face or not. It reduces the computational task as it doesn't require to train with non- face images. This process is divided into two steps. In the first step, region of the image (20*20 Pixel size) is fed as input to filter made up of the neural network. The output of this filter lies between [-1, 1] depicting the absence or presence of face. The filter is applied to all the regions of the image for detection of faces. The second step is to overcome the false detections found in the first step and to increase the efficiency for better results. This is possible if all the overlapping detections of the single neural network are fused together.

The Fuzzy Pattern Matching Approach:

This approach uses fuzzy theory to represent diverse, non-exact, uncertain, and inaccurate knowledge or information. And information carried in individual fuzzy set is combined to make a decision. A new method to detect faces in colour images based on the fuzzy[10] theory is proposed where two fuzzy models are used to describe the skin colour and hair colour respectively, where a uniform colour space is used to describe the colour information to increase the accuracy and stableness. Two different models have been used to take out the skin-coloured portions and the hair-coloured portions. Then a comparison is made between them with some prebuilt templates with the help of fuzzy theory-based methods for pattern matching that identifies human faces. Processes of composition and de-fuzzification form the basis of fuzzy reasoning. Fuzzy reasoning is performed to recognize face in the context of a fuzzy system model that consists of control, solution, and working data variables; fuzzy sets; hedges; fuzzy; and a control mechanism.

Spatially Invariant Feature Matching:

Some existing feature matching methods use local feature information, such as curvature values or other information pertaining to the interest point, to provide accurate output. These dependencies limit the feature matching algorithms to the specific task and cannot be generalized to cases in which the output information of different detectors is variable. The spatial invariant feature matching (SIFM) method is generalized for all detection techniques as it only considers the feature coordinates output by any feature point detector. Moreover, as it is based on invariant feature

dissimilarity techniques, the proposed method is also invariant to local and global deformations. SIFM method based on the geometric invariant theory is used often.

Other Approaches:

Besides the above-mentioned approaches used for face recognition, some researchers also used other methods to perform the studies on face recognition, i.e., the rules of the shape and albedo of a face under all possible illumination conditions. In order to develop a universal face recognition system which can handle all face recognition factors, the integrated approach could be a choice. A method that integrates the above different methods and applies different techniques would be the answer to all the drawbacks. The reduction of measurements is an important phase in the identification of trends and computer education. This is not just because vast sections of data were not collected, but also because the incredibly high dimensional existence of raw data makes very challenging to actually exploit the raw data (ex. facial patches). Meaningful components (for identification or more involved components) usually comprise just a tiny fraction of the raw material and cannot be collected explicitly by basic methods such as sampling and cutting. As an example, an audio signal for one channel normally produces about 10,000 samples a second and a song lasting three minutes would have about 1,800,000 samples. It is prohibitive to use the raw signal for identifying musical genres directly so we may attempt to extract helpful music elements, including pitch, tempo so instrument knowledge which might convey our hearing experience better. The goal of dimensional reduction is to derive valuable knowledge and the input data element into categories to minimize computation costs and to solve the element question curse. They will have to determine the performance of the tests in relation to the options of pattern recognition techniques. Two major evaluation curves exist: the ROC curve and the curve PR (precision and recall) curve. There are two primary evaluation curves. The curve of Roc discusses how the true positive rate compares to the false positive rate, and the PR curve eliminates the correlation between the detection rate (recall) and the detection accuracy. The true positive is the portion of face pictures to be identified by the device in two category surveillance situations (for example, face and non-faces) while the fake positive is the portion of non-facial details to be identified as faces. The main positive word here is as big as the rate of identification and alert.

Conclusion:

Holistic approaches to face detection depend on global information of faces so the disadvantage of this approach is that the variances captured may not be relevant features of the face. Face recognition is indeed a difficult problem as faces can vary a lot in their orientation, facial expression and lighting conditions. The goal was to provide a survey of recent holistic approaches to face detection. The holistic approaches are eigen face based method, spatial matching detector method, neural networks method and fuzzy theory-based method. The holistic approaches have the

main advantage of distinctly capturing the most prominent features within the given facial images, so as to uniquely identify individuals amongst the given set and also automatically finding features. However, disadvantages are that face recognition performance could be drastically be affected by lighting, orientation and features found from faces may not form part of the face but may be some other feature has been captured. For example, features from the background of a facial image. So in order to develop a universal face recognition system which can handle all face recognition factors, the integrated approach could be a choice.

CNN has made remarkable progress, especially in image processing and vision related tasks, and has thus revived the interest of researchers in ANNs. In this context, several research works have been carried out to improve CNN's performance on such tasks. The advancements in CNNs can be categorized in different ways, including activation, loss function, optimization, regularization, learning algorithms, and innovations in architecture. This paper reviews advancement in the CNN architectures, especially based on the design patterns of the processing units and thus has proposed the taxonomy for recent CNN architectures. In addition to the categorization of CNNs into different classes, this paper also covers the history of CNNs, its applications, challenges, and future directions. The learning capacity of CNN is significantly improved over the years by exploiting depth and other structural modifications. It is observed in recent literature that the main boost in CNN performance has been achieved by replacing the conventional layer structure with blocks. Nowadays, one of the paradigms of research in CNN architectures is the development of new and effective block architectures. The role of a block in a network can be that of an auxiliary learner. These auxiliary learners either exploit spatial or feature-map information or even boost input channels to improve performance. These blocks play a significant role in boosting CNN performance by making problem-aware learning. Moreover, the block-based architecture of CNN encourages learning in a modular fashion and thereby, making architecture simpler and more understandable. The concept of the block being a structural unit is going to persist and further enhance CNN performance. Additionally, the idea of attention and exploitation of channel information, in addition to spatial information, is expected to gain more importance.

3.2. Comparative study on various subtitles

(Title, Year, Authors)	Methodology or Techniques used (Mention specific algorithms or recent technologies)	Advantages	Issues	Metrics used(those are used to justify the performanc eof the used scheme)

Review of MRI-based brain tumor image segmentation using deep learning methods, 2016, Ali IúÕn, Cem Direkoglu, Melike Sah	Convolution Neural Network	(CNN) have the advantage of automatically learning representative complex features forboth healthy braintissues and tumor tissues directly from the multi-modal MRI images. Future improvements and modifications in CNN architectures and addition of complementary information from other imaging modalities such as Positron Emission Tomography (PET), Magnetic Resonance Spectroscopy (MRS) and Diffusion Tensor Imaging (DTI) may improve the current methods, eventually leading to the development of clinically acceptable automatic glioma	Automatic segmentation of gliomas is a very challenging problem. Tumor bearing brain MRI data is a 3D data where tumor shapes, size and location can varygreatly from patientto patient. Alsotumor boundariesare usually unclearand irregular with discontinuities, posing great challenge especially against traditional edge-based methods. In addition to these, brain tumor MRI data obtained from clinical scans or synthetic databases11 are inherently complex. MRI devices and protocols used for acquisition can vary	Neural Networks

		segmentation methods for better diagnosis.	dramatically from scan to scan imposing intensity biases and other variations for each different slice of image in the dataset. The need for several modalities to effectively segment tumor sub-regions even adds to this complexity.	

A review of the use of convolutional neural networks in agriculture,2018, A. Kamilaris, F. X. Prenafeta-Boldú	Convolution Neural Network	Feature engineering (FE) is a complex, time-consuming process which needs to be altered whenever the problem or the data set changes. Thus, FE constitutes an expensive effort that depends on experts' knowledge and does not generalize well (Amara et al., 2017). On the other hand, CNN do not require FE, as they locate the important features automatically through the training process. Quite impressively, in the case of fruit counting, the model learned explicitly to count (Rahnemoonfar and Sheppard, 2017). Convolutional neural networks seem to generalize well (Pan and Yang, 2010) and	Their main disadvantage is that CNN can sometimes take much longer to train. However, after training, their testing time efficiency is much faster than other methods such as SVM or KNN (Chen et al., 2014; Christiansen et al., 2016). Another important disadvantage (see earlier) is the need for large data sets (i.e. hundreds or thousands of images), and their proper annotation, which is sometimes a delicate procedure that must be performed by domain experts. The current authors' personal experimentation with CNN (see earlier) reveals this	Neural Networks

		they are quite robust even under challenging conditions such as illumination, complex background, size and orientation of the images, and different resolution (Amara et al., 2017).	problem of poor data labelling, which could create significant reduction in performance and precision achieved.	

Application of Deep Learning in Food: A Review,2019, Lei Zhou,Chu Zhang,Fei Liu,Zhengjun Qiu,Yong He,	Convolution Neural Network	The most significant advantage of deep learning technology is feature learning. Traditional machine learning approaches use raw data as input or deal with classification tasks based on hand- crafted features. Deep learning methods can learn representational features from the dataset during the training process, and demonstrate stronger ability than traditional methods. Another characteristic of deep learning is its ability of transfer learning. Inthe researches mentioned in the section "Food recognition and classification," we found that most of them exploited pretrained CNN models based on large dataset (such as ImageNet) and fine-tuned the models on	The fact that deep learning has shortcomings is undeniable. Due to long training time as well as hardware restrictions, added with the high complexity and numerous hyperparameters of the model, the optimization tasks would be very complicated and time-consuming. The mentioned gpus for computing acceleration and matched processors and other hardware are very expensive. It will take much longer time to train a dnn only using cpusas the computational hardware. Furthermore, deep learning requires big data for training, and the acquisition of reliable big dataset is another difficult problem. Data collection and	Neural Networks

		their target datasets, which could reduce the difficulty and time consumption for training a model (even based on a much smaller dataset). Moreover, there were also some authors deploying features extracted by a CNN to train another classifier like SVM, to transfer the knowledge from CNN model to a new classifier.	annotation will take a lot of time and energy. Some open datasets for academic research and challenge competition were collected and manually labeled by experts or volunteers, even could be directly downloaded from the internet by machines; thus, there would be some mistakes inevitably.	

Jointly network image processing: multi-task image semantic segmentation of indoor scene based on CNN ,2020, Li Huang,Meiling He,Chong Tan,Du Jiang,Gongfa Li,Hui Yu,	Convolution Neural Network	In this paper, by adding a branch of image semantic segmentation based on a fully CNN to the target detection framework, a multi-task vision technologythat combines indoor scene target detection and semantic segmentation is achieved. While obtaining the relative position of the target object, it can also interpret the semantic information between the targets, greatly enhancing the understanding of the actual scene information. Firstly, a multi-task semantic segmentation model based on improved	The author will consider the problems of many semantic categories of indoor scene objects, complex backgrounds and uneven lighting etc. and introduce the depth information of the image to improve the robustness of the model and complete the robot's better analysis of the scene information.	Neural Networks

		FCN is proposed, and the components of the model are studied and analysed. Then through the self-built indoor scene database, RGB images are used to train them to obtain the training model. Finally, the obtained model is evaluated in terms of loss function and visualisation of actual scenes. From the experimental results, it can be seen that the obtained model achieves the semantic segmentation of joint target detection in indoor scenes, indicating that the method proposed in this paper is feasible and effective.		

A New Method for Face Recognition Using Convolutional Neural Network,2017, Patrik KAMENCAY, Miroslav BENCO, Tomas MIZDOS, Roman RADIL	Convolution Neural Network	In this work, we presented an experimental evaluation of the performance of proposed CNN. The overall performances were obtained using the different number of training images and test images. The convolutional neural networks achieve the best results so far. Using complex architectures, it is possible to reach accuracy rates of	This impressing outcome, cnns cannot work without negative impacts. Very huge training datasets lead to a high computation load and memory usage, which then needs high processing power to be able to be applied usefully.	Neural Networks

		about 98 %.		

Image Classification using Convolutional Neural Networks,2018, Muthukrishnan Ramprasath, M.Vijay Anand , Shanmugasundaram Hariharan	Convolution Neural Network	In this paper, we used Convolutional Neural Networks (CNN) for image classification using images form hand written MNIST data sets. This data sets used both and training and testing purpose using CNN. It provides the accuracy rate 98%. Images used in the training purpose are small and Grayscale images. The computational time for processing these images is very high as compare to other normal JPEG images. Stacking the model with more layers and training the network with more image data using clusters of GPUs will provide more accurate results of classification of images. The future enhancement will focus on classifying the colored images of large size and its very useful for image segmentation process.	This impressing outcome, cnns cannot work without negative impacts. Very huge training datasets lead to a high computation load and memory usage, which then needs high processing power to be able to be applied usefully.	Neural Networks

A study on Image Classification based on Deep Learning	Deep neural network	In conclusion, this research is about image classification	To achieve maximum level of accuracy the	Neural Networks

and Tensorflow,2019, Mohd Azlan Abu1 , Nurul Hazirah Indra1 , Abdul Halim Abd Rahman1 , Nor Amalia Sapiee1 and Izanoordina Ahmad1		by using deep learning via framework TensorFlow. It has three (3) objectives that have achieved throughout this research. The objectives are linked directly with conclusions because it can determine whether all objectives are successfully achieved or not. It can be concluded that all results that have been obtained, showed quite impressive outcomes. The deep neural network (DNN) becomes the main agenda for this research, especially in image classification technology. DNN technique was studied in more details starting from assembling, training model and to classify images into categories. The roles of epochs in DNN was able to control accuracy and also prevent any problems such as overfitting. Implementation of deep learning by using framework	model needs a huge and clean number of data- sets. Several images might not be recognised by the model due to less number of datasets. Time taking	

		TensorFlow also gave good results as it is able to simulate, train and classified with up to 90% percent of accuracy towards five (5) different types of flowers that have become a trained model. Lastly, Python have been used as the programming language throughout this research since it comes together with framework TensorFlow which leads to designing of the system involved Python from start until ends.		

Image Classification Using Convolutional Neural Networks,2014, Deepika Jaswal, Sowmya.V, K.P.Soman	Convolution Neural Network	This section presents the result of the classification accuracies obtained using CNN algorithm on various standard datasets. The results are presented using the classification accuracy in percentage for within train data and test data separately. Along with the classification accuracy percentage values, MSE (Mean	To achieve maximum level of accuracy the model needs a huge and clean number of data-sets. Several images might not be recognised by the model due to less number of datasets. Time taking	Neural Network

		Squared Error) graph is also presented. The Graphs show the change of MSE with respect to the training epochs. MSE metric is the simplest and widely used quality metric. It is the mean of the squared difference between original and trained approximation. A better trained image will result in lower MSE with the original image. As the value for Mean Squared Error (MSE) tends to decrease, the variation in the final reconstructed output and the original image is very less. MSE indicates the close proximity between underlying true image and the final reconstructed output. The idea here is to use enough number of epochs that would result in low MSE, high classification accuracy and with least duration for training the network.		

Image Classification with Deep Learning	Convolution Neural Network	Deep learning is a learning method for	CNN do not encode the	Neural Network

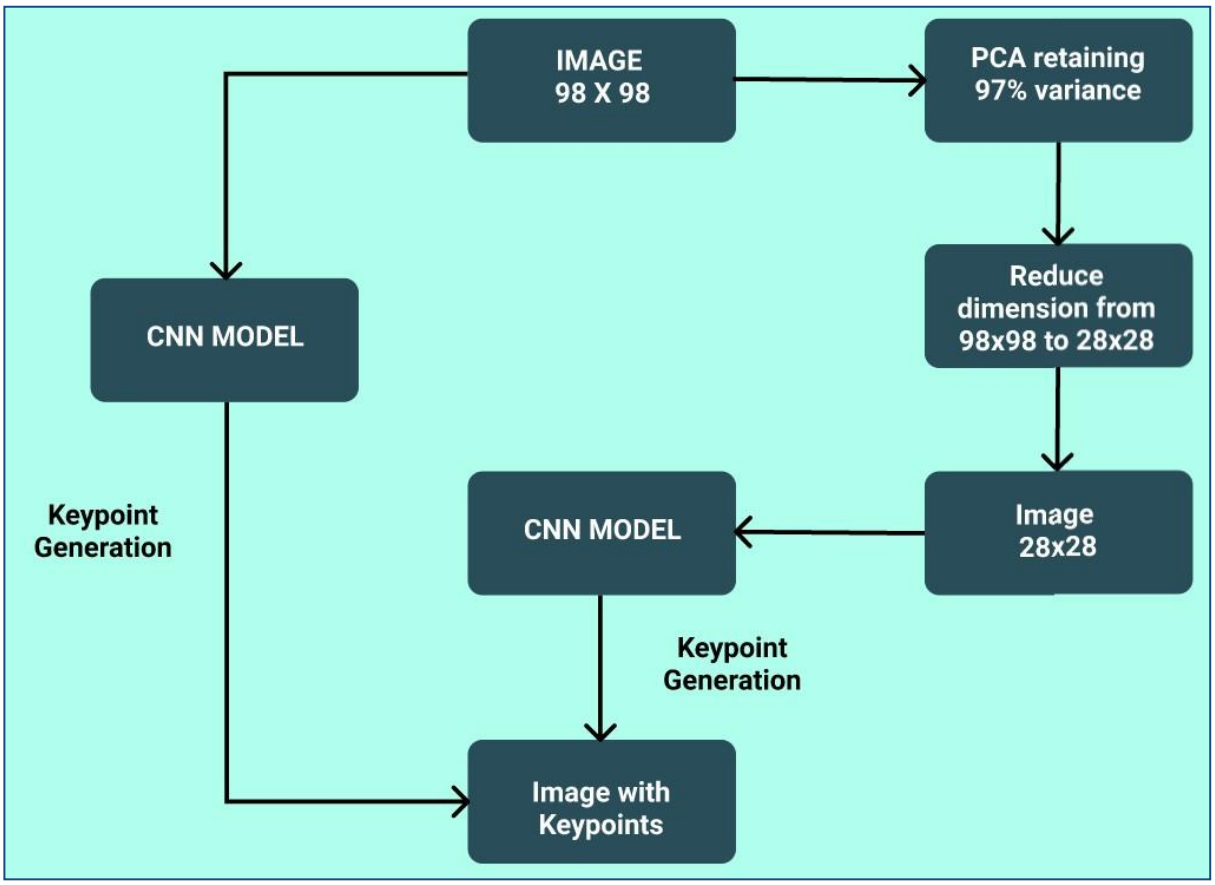
and Comparison between Different Convolutional Neural Network Structures using Tensorflow and Keras,2018, Karan Chauhan1 , Shrwan Ram2		data analysis and predictions, now days it also become very popular for image classification problems. In this paper, a deep learning convolutional neural network based on keras and tensorflow is deployed using python for binary image classification. In this study, we compare four different structures of CNN on CPU system, with different combinations of classifier and activation function. With experiments, we obtained results for each combination and observed that for binary image classification, Relu activation function and Sigmoid classifier combination gives better classification accuracy (90.54%) than any other combination of activation function and classifier. So, we conclude that on CPU system, Relu activation function	positionand orientation of object. Lack of ability to be spatially invariant to the input data. Lots of training data is required	

		and Sigmoid classifier gives better classification accuracy for binary image classification.		

The Study of Noise Effect on CNN- Based Deep Learning from Medical Images,2021, Kittipat Sriwong, Kittisak Kerdprasop, and Nittaya Kerdprasop	Convolution Neural Network	Convolution neural network, or CNN, is a deep learning algorithm that has been widely accepted as the most accurate algorithm suitable for learning patterns from images. Many research laboratories and renowned organization such as Google provide the pre-trained CNN models for further deployment. In this research work, we adopt the AlexNet architecture to build the CNN models. The focus of our research is to observe the performance of CNN models when the trained images contain noises at various levels. Medical images are the scope of our observation. We categorized medical noisy images into two groups, namely NIH and NNIH groups. The NIH	To achieve maximum level of accuracy the model needs a huge and clean number of data-sets. Several images might not be recognised by the model due to less number of datasets. Time taking	Neural Network

		group is a group of medical images that noises have been added to cover harmoniously the area of the disease symptoms, whereas the NNIH group is the group of images that noises do not harmoniously cover the area of the disease symptoms.		

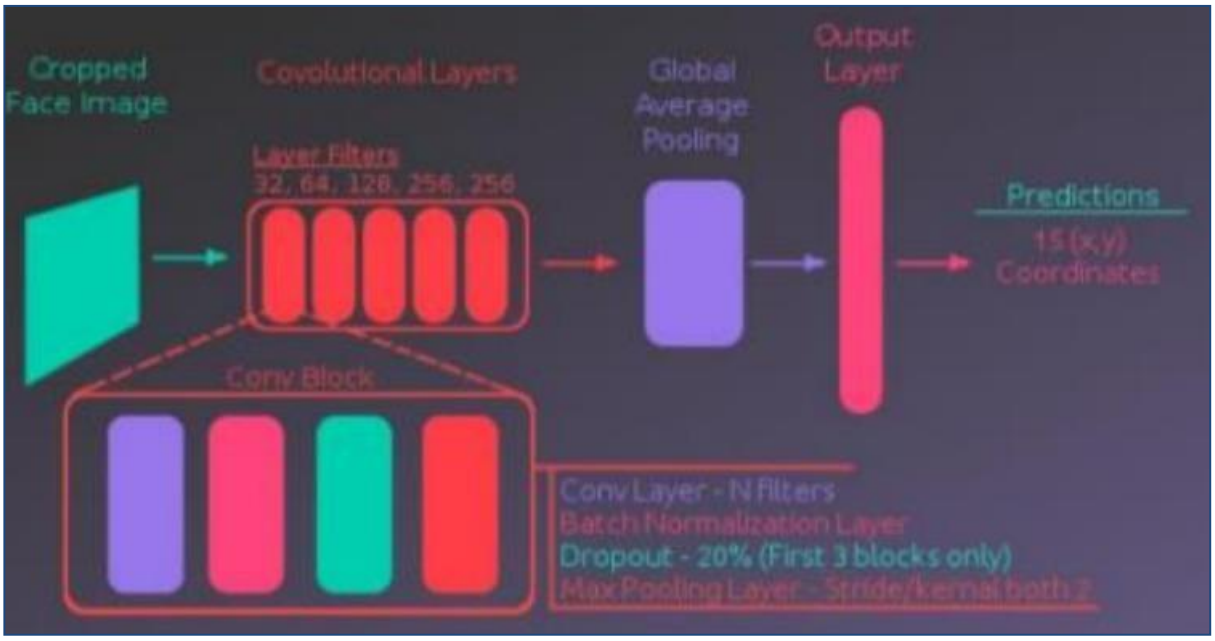
4. System design



CNN MODEL:

To achieve better key points detection accuracy (lower loss), we build a convolution neural network model as an advanced baseline. The architecture diagram of our CNN is shown in Fig. 1. The number shown above each layer demonstrates the size of its corresponding activation, and the description below each layer describes its function. The hyperparameters of those convolutional layers in the network are illustrated in Table. 1. Both hidden layers have 500 neurons and the output layers generate a 30- element vector, same as the former one hidden layer neural network, representing the coordinates of the key points.

Model Summary:



Steps Of CNN:

• Step 1: Convolution Operation The first building block in our plan of attack is convolution operation. In this step, we will touch on feature detectors, which basically serve as the neural network's filters. We will also discuss feature maps, learning the parameters of such maps, how patterns are detected, the layers of detection, and how the findings are mapped out.

• Step 1(b): ReLU Layer The second part of this step will involve the Rectified Linear Unit or ReLU. We will cover ReLU layers and explore how linearity functions in the context of Convolutional Neural Networks. Not necessary for understanding CNN's, but there's no harm in a quick lesson to improve your skills.

• Step 2: Pooling In this part, we'll cover pooling and will get to understand exactly how it generally works. Our nexus here, however, will be a specific type of pooling; max pooling. We'll cover various approaches, though, including mean (or sum) pooling. This part will end with a demonstration made using a visual interactive tool that will definitely sort the whole concept out for you.

• Step 3: Flattening This will be a brief breakdown of the flattening process and how we move from pooled to flattened layers when working with Convolutional Neural Networks.

• Step 4: Full Connection In this part, everything that we covered throughout the section will be merged together. By learning this, you'll get to envision a fuller picture of how Convolutional Neural Networks operate and how the "neurons" that are finally produced learn the classification of images.

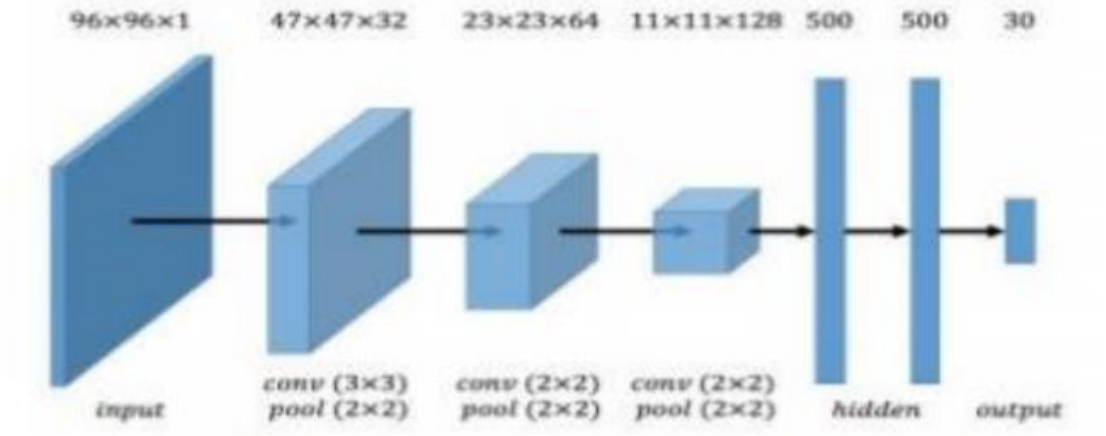


Figure 1. Convolution Neural Network Architecture

Table 1. Hyperparameters of CNN

	conv layer 1		conv layer 2		conv layer 3	
	conv	pool	conv	pool	conv	pool
filter	3×3	2×2	2×2	2×2	2×2	2×2
stride	1	2	1	2	1	2
pad	0	0	0	0	0	0

Explanation about complete description:

The objective of this project is to make benchmarks about how much dimensionality reduction strategies main PCA (Principal Component Analysis) helps in decreasing overfitting for recognizing facial key-points. If the idea of deep learning will be used, such as neural network and cascaded neural network then the results of these structures will be significantly better than state of-the-art methods, like feature extraction and dimension reduction algorithms. This method can help to locate the key points in a given image using deep architectures to not only obtain lower loss for the detection task but also accelerate the training and testing process for realworld applications. Two basic neural network structures can be constructed, one hidden layer neural network and other one the convolution neural network as our baselines. An approach to better locate the coordinates of facial key points with introduced features other than raw input will be proposed. Specifically, a block of pre - trained Inception Model was used to extract Intermediate features and using different deep structures to compute the final output vector. The experimental results will consist of two parts as well. First, the detection accuracy will be calculated by comparing training, validation and testing loss among all the 5 models illustrated in the selected journal. Second, the time complexity can be evaluated of the 5 models by comparing

their time consumption in both training and testing phases. The experiments results will show the effectiveness of deep structures for facial key points detection tasks, and using the pre-trained Inception Model has slightly improved the performance of detection compared to baseline Methods.

Another method that can be proposed is using PCA, for this we will use a list of 96×96-pixel 8- bit gray level images with their corresponding (x, y) coordinates of 15 facial key points. First, cross validation will be adopted to randomly split the data set into a training set and a test set, so that we can develop our algorithm on the training set and assess its performance on the test set. The algorithm first performed histogram stretching to enhance the image contrast by stretching the range of pixel intensity of each training image. Then, in order for noise reduction, principal components analysis may be applied on the stretched images to obtain the eigen faces. Using the resultant eigen faces, the mean patch searching algorithm can be implemented with correlation scoring and mutual information scoring to predict the left- and right-eye centers for any query test facial images. We will get to know the minimum hold-out test MSE for predicting left eye center and the minimum holdout test MSE for predicting right eye center. Thus, the best average test MSE for predicting both facial key points can be calculated.

Description of 15 key points present in dataset with (x,y) representing each coordinates. 'left_eye_center_x', 'left_eye_center_y', 'right_eye_center_x', 'right_eye_center_y', 'left_eye_inner_corner_x', 'left_eye_inner_corner_y', 'left_eye_outer_corner_x', 'left_eye_outer_corner_y', 'right_eye_inner_corner_x', 'right_eye_inner_corner_y', 'right_eye_outer_corner_x', 'right_eye_outer_corner_y', 'left_eyebrow_inner_end_x', 'left_eyebrow_inner_end_y', 'left_eyebrow_outer_end_x', 'left_eyebrow_outer_end_y', 'right_eyebrow_inner_end_x', 'right_eyebrow_inner_end_y', 'right_eyebrow_outer_end_x', 'right_eyebrow_outer_end_y', 'nose_tip_x', 'nose_tip_y', 'mouth_left_corner_x', 'mouth_left_corner_y', 'mouth_right_corner_x', 'mouth_right_corner_y', 'mouth_center_top_lip_x', 'mouth_center_top_lip_y', 'mouth_center_bottom_lip_x', 'mouth_center_bottom_lip_y'

We will begin with convolutional neural network.(CNN) There is no doubt that CNN is the best solution to this problem with such a low RMSE. It has been proved that CNN performs well to image recognition both from theory and practice. The advantages include feature extraction and soft weight sharing, which is the unique properties of CNN. It is these characters that make CNN such a perfect approach. However, the disadvantage still need to be mentioned since this cannot be felt from these paper unless to run our code on computers. Very long time will be taken though the code is run on the server. It raises the problem of tuning because it is time consuming. Thence, we have reasons to believe that better RMSE can be achieved if suitable parameters are applied.

Secondly, Neural network doesn't get the RMSE as well as our expectation. Compared with CNN, neural network itself is not a specific method for image recognition so this would not be surprising. However, we originally expect that neural network may perform better can Regression methods. Now taking a look back, neural network's failure may be attributed to overfitting and tuning. The method of 'Dropout' can avoid overfitting but there is no theoretical guidance to the setting of parameters. Beside its high RMSE compared with CNN, it is still time consuming to train neural network. Overall, we do not recommend neural network in face recognition experiment unless perfect parameters can be found. These perfect parameters can be found using PCA when retaining somewhat from 95-97% variance. So, the main objective of this paper is to check whether PCA is suitable for improving CNN model or not and when we get a optimized model, we using the predicted keypoints on a person's face create a selfie filter something like sunglasses or fake moustache to display the use of the facial keypoints.

5. Software Requirement Specifications

The code is in Python (version 3.6 or higher). You also need to install OpenCV and Keras libraries.

1. Install Anaconda

2. Go to Anaconda Prompt:

    Run these commands:

        a) conda install keras

        b) conda install opencv

3. Open Spyder

4. Now your machine is ready to execute the code.

6. Experimental Results & Discussion

6.1 Source Code

my_CNN_Model:

```
from keras.models import Sequential

from keras.models import load_model

from keras.layers import Convolution2D, MaxPooling2D, Dropout

from keras.layers import Flatten, Dense

from keras.optimizers import SGD, RMSprop, Adagrad, Adadelta,
Adam, Adamax, Nadam


def get_my_CNN_model_architecture():

'''

The network should accept a 96x96 grayscale image as input,
and it should output a vector with 30 entries,

corresponding to the predicted (horizontal and vertical)
locations of 15 facial keypoints.

'''

model = Sequential()

model.add(Convolution2D(32, (5, 5), input_shape=(96,96,1),
activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Convolution2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.1))


model.add(Convolution2D(128, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Dropout(0.2))


model.add(Convolution2D(30, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.3))


model.add(Flatten())


model.add(Dense(64, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(30))


return model;


def compile_my_CNN_model(model, optimizer, loss, metrics):
model.compile(optimizer=optimizer, loss=loss, metrics=metrics)


def train_my_CNN_model(model, X_train, y_train):
return model.fit(X_train, y_train, epochs=100, batch_size=200,
verbose=1, validation_split=0.2)


def save_my_CNN_model(model, fileName):
model.save(fileName + '.h5')
def load_my_CNN_model(fileName):
return load_model(fileName + '.h5')
```

Model_builder.py:

```
from utils import load_data

from my_CNN_model import *

import cv2




X_train, y_train = load_data()









# Setting the CNN architecture

my_model = get_my_CNN_model_architecture()



# Compiling the CNN model with an appropriate optimizer and
loss and metrics

compile_my_CNN_model(my_model, optimizer = 'adam', loss =
'mean_squared_error', metrics = ['accuracy'])



# Training the model

hist = train_my_CNN_model(my_model, X_train, y_train)




save_my_CNN_model(my_model, 'my_model')
```

Shades.py:

```
from my_CNN_model import *

import cv2

import numpy as np



# Load the model built in the previous step

my_model = load_my_CNN_model('my_model')



# Face cascade to detect faces

face_cascade                                                    =
cv2.CascadeClassifier('cascades/haarcascade_frontalface_defaul
t.xml')



# Define the upper and lower boundaries for a color to be
considered "Blue"

blueLower = np.array([100, 60, 60])

blueUpper = np.array([140, 255, 255])



# Define a 5x5 kernel for erosion and dilation

kernel = np.ones((5, 5), np.uint8)



# Define filters

filters = ['images/sunglasses.png', 'images/sunglasses_2.png',
'images/sunglasses_3.jpg',          'images/sunglasses_4.png',
'images/sunglasses_5.jpg', 'images/sunglasses_6.png']

filterIndex = 0



# Load the video
```

```
camera = cv2.VideoCapture(0)


# Keep looping
while True:
    # Grab the current paintWindow
    (grabbed, frame) = camera.read()
    frame = cv2.flip(frame, 1)
    frame2 = np.copy(frame)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    # Add the 'Next Filter' button to the frame
    frame = cv2.rectangle(frame, (500,10), (620,65), (235,50,50), -1)
    cv2.putText(frame, "NEXT FILTER", (512, 37), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)


    # Detect faces
    faces = face_cascade.detectMultiScale(gray, 1.25, 6)


    # Determine which pixels fall within the blue boundaries and then blur the binary image
    blueMask = cv2.inRange(hsv, blueLower, blueUpper)
    blueMask = cv2.erode(blueMask, kernel, iterations=2)
    blueMask = cv2.morphologyEx(blueMask, cv2.MORPH_OPEN, kernel)
    blueMask = cv2.dilate(blueMask, kernel, iterations=1)
```

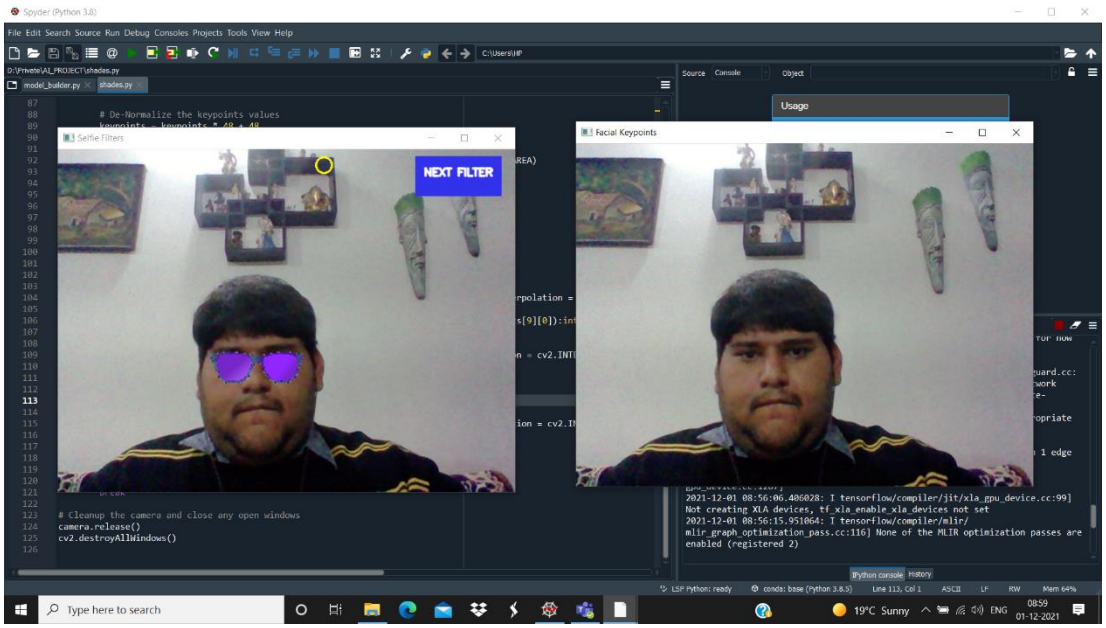
```
    # Find contours (bottle cap in my case) in the image

    (cnts,      _)      =      cv2.findContours(blueMask.copy(),
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)

    center = None



    # Check to see if any contours were found

    if len(cnts) > 0:

     # Sort the contours and find the largest one -- we

     # will assume this contour correspondes to the area of
the bottle cap

        cnt = sorted(cnts, key = cv2.contourArea, reverse =
True)[0]

        # Get the radius of the enclosing circle around the
found contour

        ((x, y), radius) = cv2.minEnclosingCircle(cnt)

        # Draw the circle around the contour

        cv2.circle(frame, (int(x), int(y)), int(radius), (0,
255, 255), 2)

        # Get the moments to calculate the center of the
contour (in this case Circle)

        M = cv2.moments(cnt)

        center = (int(M['m10'] / M['m00']), int(M['m01'] /
M['m00']))



        if center[1] <= 65:

            if 500 <= center[0] <= 620: # Next Filter

                filterIndex += 1

                filterIndex %= 6

                continue
```

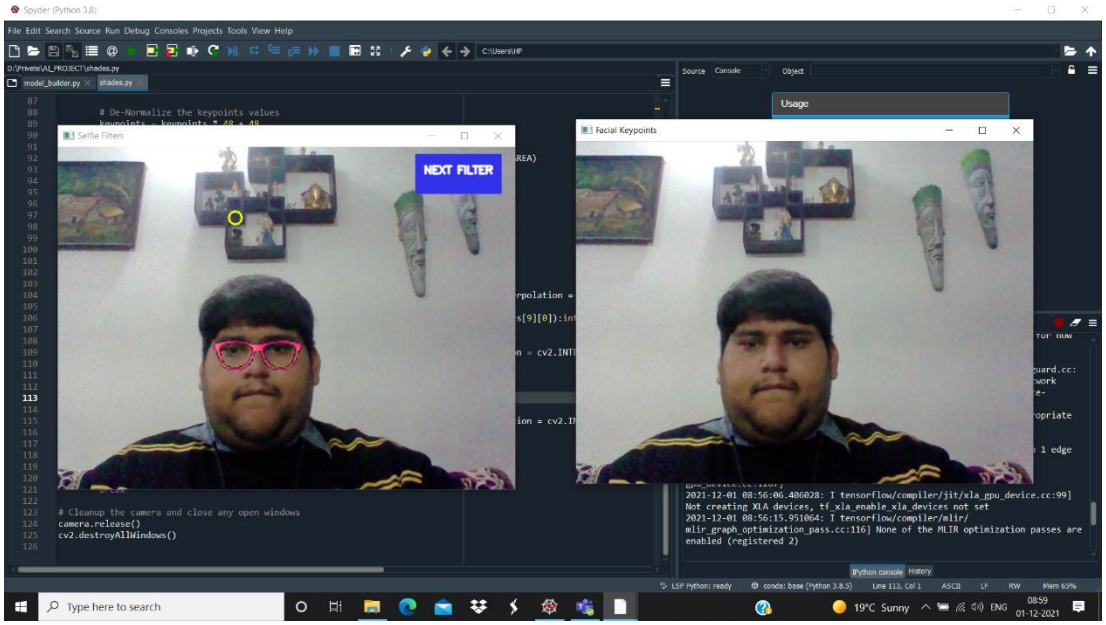
```
    for (x, y, w, h) in faces:

        # Grab the face
        gray_face = gray[y:y+h, x:x+w]
        color_face = frame[y:y+h, x:x+w]

        # Normalize to match the input format of the model - Range of pixel to [0, 1]
        gray_normalized = gray_face / 255

        # Resize it to 96x96 to match the input format of the model
        original_shape = gray_face.shape # A Copy for future reference
        face_resized = cv2.resize(gray_normalized, (96, 96), interpolation = cv2.INTER_AREA)
        face_resized_copy = face_resized.copy()
        face_resized = face_resized.reshape(1, 96, 96, 1)

        # Predicting the keypoints using the model
        keypoints = my_model.predict(face_resized)

        # De-Normalize the keypoints values
        keypoints = keypoints * 48 + 48

        # Map the Keypoints back to the original image
        face_resized_color = cv2.resize(color_face, (96, 96), interpolation = cv2.INTER_AREA)
```

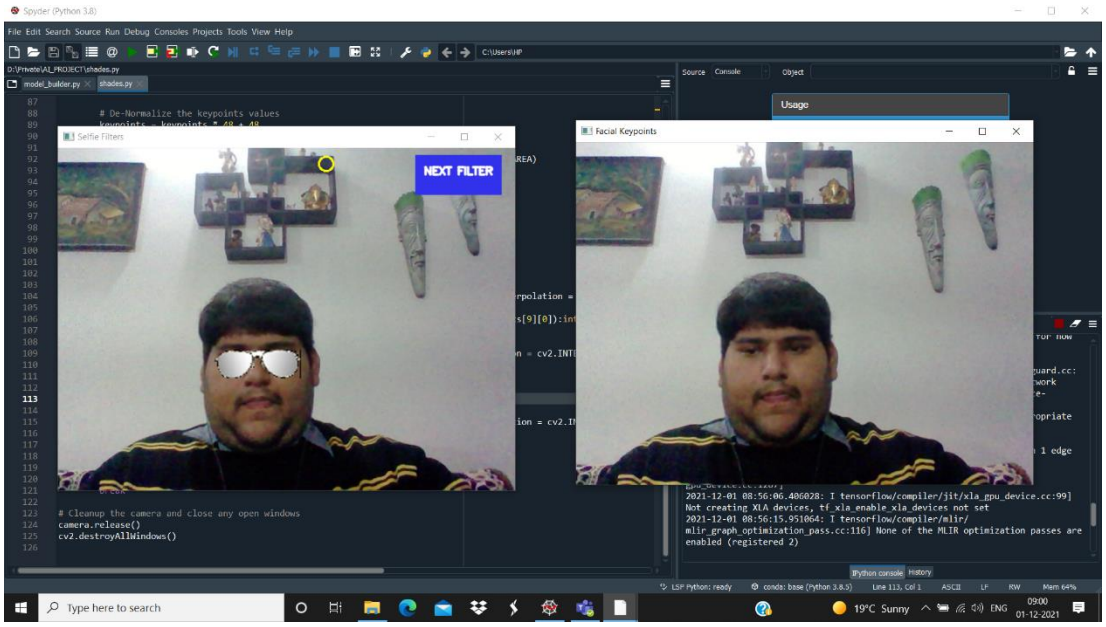
```
        face_resized_color2 = np.copy(face_resized_color)


        # Pair them together

        points = []

        for i, co in enumerate(keypoints[0][0::2]):

            points.append((co, keypoints[0][1::2][i]))


        # Add FILTER to the frame

        sunglasses      =      cv2.imread(filters[filterIndex],
cv2.IMREAD_UNCHANGED)

        sunglass_width = int((points[7][0]-points[9][0])*1.1)

        sunglass_height          =          int((points[10][1]-
points[8][1])/1.1)

        sunglass_resized        =        cv2.resize(sunglasses,
(sunglass_width,     sunglass_height),     interpolation     =
cv2.INTER_CUBIC)

        transparent_region = sunglass_resized[:,:,:3] != 0


face_resized_color[int(points[9][1]):int(points[9][1])+sunglas
s_height,
int(points[9][0]):int(points[9][0])+sunglass_width,:][transpar
ent_region] = sunglass_resized[:,:,:3][transparent_region]


        # Resize the face_resized_color image back to its
original shape

        frame[y:y+h, x:x+w] = cv2.resize(face_resized_color,
original_shape, interpolation = cv2.INTER_CUBIC)


        # Add KEYPOINTS to the frame2

        for keypoint in points:
```

```
            cv2.circle(face_resized_color2, (int(x), int(y)),
1, (0,255,0), 1)


        frame2[y:y+h, x:x+w] = cv2.resize(face_resized_color2,
original_shape, interpolation = cv2.INTER_CUBIC)

        cv2.imshow("Selfie Filters", frame)

        cv2.imshow("Facial Keypoints", frame2)


    # If the 'q' key is pressed, stop the loop

    if cv2.waitKey(1) & 0xFF == ord("q"):

        break


# Cleanup the camera and close any open windows

camera.release()

cv2.destroyAllWindows()
```

6.2. Screenshots




Selfie Filters
NEXT FILTER
Facial Keypoints
Usage
# Cleanup the camera and close any open windows
camera.release()
cv2.destroyAllWindows()
Not creating XLA devices, tf_xla_enable_xla_devices not set
Type here to search
19°C Sunny
ENG
08:59
01-12-2021


Selfie Filters
NEXT FILTER
Facial Keypoints
Usage
# Cleanup the camera and close any open windows
camera.release()
cv2.destroyAllWindows()
Not creating XLA devices, tf_xla_enable_xla_devices not set
Type here to search
19°C Sunny
ENG
09:00
01-12-2021

7. References

[1] M.Turk & A.Pentland: Eigenfaces for recognition, J.Cog.Neuroscience, Volume 3, (1991)

[2] M.Kirby & L.Sirovich: Application of the Karhunen-loeve procedure for the characterization of human faces, Springer- Verlag, Berlin, 1989

[3] T.Kohonen: Self-organization and Associative Memory, IEEE, 1990

[4] Nitish Srivastava & Geoffrey E. Hinton & Alex Krizhevsky & Ilya Sutskever & Ruslan Salakhutdinov: Dropout: A simple way to prevent neural network from overfitting, Journal of Machine learning Research, Pages:1929-1958 2015

[5] T.Berg & P.N.Belhumeur:om-vs-pete classifiers and identity-preserving alignment for face verification,BMVC, 2012

[6] Zhu.X & Ramanan D: Face detection, pose estimation, and landmark localization in the wild, CVPR, 2012

[7] N.Kumar & A.C.Berg & P.N.Belhumeur & S.K.Nayar: Attribute and smile calssifiers for face verification, ICCV, 2009

[8] Yaniv Taigman & Ming Yang & Marc' Aurelio Ranzato: Deep-Face: Closing the Gap to Human-Level Performance, CVPR, 2014

[9] D.Chen & X.Cao & L.Wang & F.Wen & J.Sun: Bayesian face revisited: A joint formulation, ECCV, 2012

[10] Y.Taigman & L.Wolf :Leveraging billions of faces to overcome performance barriers in unconstrained face recognition,

[11] Y.Bengio: Learning deep architectures for AI, Foundations and Trends in Machine Learning, 2009

[12] Y.Sun & X.Wang & X.Tang: Deep convolutional network cascade for facial point detection, CVPR, 2013

[13] T.Ahonen & A.Hadid & M.pietikainen: Face description with local

binary patterns: Application to face recognition, PAMI, 2006

[14] Karen Simonyan & Andrew Zisserman: Very deep convolutional networks

for large-scale image recognition, ICLR, 2005

[15] M. La Cascia & S. Sclaroff & V. Athitsos: Fast, Reliable Head Tracking under Varying Illumination: An Approach Based on Registration of Texturemapped 3D Models. IEEE Transactions on Pattern Analysis and Machine

Intelligence, April 2000

26

[16] P. de Cuetos, C. Neti & A. Senior: Audio-visual intent to Speak Detection for Human-computer Interaction In proceedings of the IEEE International

Conference on Acoustics, Speech, and Signal Processing 2000

[17] James M. Rehg & Kevin P. Murphy & Paul W. Fieguth: Vision- based

Speaker-detection using Bayesian Networks. In Proceedings of Computer Vision

and Pattern Recognition, Volume 2, pages 110-116, 1999

[18] Y. Matsumoto & A. Zelinsky: An Algorithm for Real-time Stereo

Vision Implementation of Head Pose and Gaze Direction Measurement. In IEEE

International Conference on Face and Gesture, page 499, 2000

[19] Second International Workshop on Performance and Evaluation of Tracking and Surveil- lance. IEEE ,December 2001

[20] T. Tan: Second IEEE International Workshop on Visual Surveillance.

IEEE, 1999

[21] Rosalind W. Picard: Affective Computing. MIT Press, 2009

[22] E. Petajan: The Communication of Virtual Human Faces using mpeg-4

Tools. In Inter- national Symposium on Circuits and Systems, Voluume 1, pages

307-310, 2010

[23] Chin-Seng Chua & FengHan & Yeong-KhingHo: 3DHumanFace Recognition using Point Signature. In International Conference on Face and Gesture

Recognition, Voluume 1, pages 307-310, 2010

[24] P. Jonathon Phillips & Patrick J. Rauss & Sandor Z. Der. FERET:

(Face Recognition Technology) Recognition Algorithm Development and Test Results. Technical Report ARL-TR-995, Army Research Laboratory, October, 1996

[25] F. Prokoski: History, Current Status, and Future of Infrared Identification. In Proceedings of IEEE Workshop on Computer Vision Beyond the Visible Spectrum: Methods and Applications, pages 5-14, June 2000. Facial Thermogram

[26] P. Jonathon Phillips & Hyeonjoon Moon & Patrick Rauss & Syed A. Rizvi: The FERET September 2009 Database and Evaluation Pro- cedure. In Josef Bigun, Gerard Chollet, and Gunilla Borgefors, editors, Audio- and Video-based Biometric Person Authentication, volume 1206 of Lecture Notes in Computer Science, pages 395-402. Springer, March 2010.

[27] Duane M.Blackburn & Mike Bone & P.Jonathon Phillips: Facial Recognition Vendor Test 2000 Evaluation Report. Technical Report, Department of Defence Counterdrug Technology Development Program Office, February 2011

[28] O. Barkan, J. Weill & L. Wolf & H. Aronowitz: Fast high dimen- sional vector multiplication face recognition, ICCV, 2013

[29] X. Cao & D. Wipf & F. Wen & G. Duan & J. Sun: A practical transfer learning algorithm for face verification, ICCV, 2013

[30] D.Chen & X.Cao & F.Wen & J.Sun: Blessing of dimensionality: Highdimensional feature and its efficient compression for face verification, CVPR, 2013

[31] M.Osadchy & Y.LeCun & M.Miller: Synergistic face detection and pose estimation with energy-based models, JMLR, 2007

[32] S. Chopra & R. Hadsell & Y. LeCun: Learning a similarity metric discriminatively, with application to face verification, CVPR, 2005

27

[33] G. B. Huang & H. Lee & E. Learned-Miller: Learning hierarchical

representations for face verification with convolutional deep belief networks, CVPR, 2012

[34] Jochen Triesch & Christoph von der Malsburg: Fisher vector faces in the wild, BMVC, 2013

[35] K. Simonyan & O. M. Parkhi & A. Vedaldi & A. Zisserman: elforganized Integration of Adaptive Visual Cues for Face Tracking. In International Conference on Face and Gesture Recognition, IEEE, 2015

[36] G. J. Edwards & C. J. Taylor & T. F. Cootes: Interpreting Faces using Active Appearance Models. In International Conference on Face and Gesture Recognition, April 2012

[37] Roberto Brunelli & Tomaso Poggio: Face Recognition: Features versus Templates. IEEE Transactions on Pattern Analysis and Machine Intelligence, October 2013

[38] Ian Craw & Peter Cameron: Face Recognition by Computer Proceedings of the British Machine Vision Conference, Springer Verlag, 2012

[39] G.J.Edwards & C.J.Taylor & T.F.Cootes: Interpreting Faces using Active Appearance Models. In International Conference on Face and Gesture Recognition ,April 2012

[40] Christopher M.Bishop: Pattern recognition and Machine learning, Springer 2006

Annexure

Facial Keypoints Detection

Shenghao Shi

Abstract

Detect facial keypoints is a critical element in face recognition. However, there is difficulty to catch keypoints on the face due to complex influences from original images, and there is no guidance to suitable algorithms. In this paper, we study different algorithms that can be applied to locate keyponits. Specifically: our framework (1)prepare the data for further investigation (2)Using PCA and LBP to process the data (3) Apply different algorithms to analysis data, including linear regression models, tree based model, neural network and convolutional neural network, etc. Finally we will give our conclusion and further research topic. A comprehensive set of experiments on dataset demonstrates the effectiveness of our framework.

1 INTRODUCTION

1.1 Background

Face recognition in unconstrained images is at the fore- front of the algorithmic perception revolution. The social and cultural implications of face recognition technologies are far reaching, yet the current performance gap in this domain between machines and the human visual system serves as a buffer from having to deal with these implications.

Recognizing faces is something that people usually do effortlessly and without much conscious thought, yet it has remained a difficult problem in the area of computer vision, where some 20 years of research is just beginning to yield useful technological solutions. As a biometric technology, automated face recognition has a number of desirable properties that are driving research into practical techniques.

The history of face recognition is as old as computer vision, both because of the practical importance of the topic and theoretical interest from cognitive scientists. Despite the fact that other methods of identification (such as fingerprints, or iris scans) can be more accurate, face recognition has always remains a major focus of research because of its non-invasive nature and because it is people's primary method of person identification.

The problem of face recognition can be stated as 'identifying an individual from images of the face' and encompasses a number of variations other than the most familiar application of mug shot identification. One notable aspect of face recognition is the broad interdisciplinary nature of the interest in it

within computer recognition and pattern recognition; biometrics and security; multimedia processing; psychology and neuroscience. It is a field of research notable for the necessity and the richness of interaction between computer scientists and psychologists.

Perhaps the most famous early example of a face recognition system is due to Kohonen [3], who demonstrated that a simple neural net could perform face recognition for aligned and normalized face images. The type of network he employed computed a face description by approximating the eigenvectors of the face image's autocorrelation matrix; these eigenvectors are now known as eigenfaces.

Kohonen's system was not a practical success, however, because of the need for precise alignment and normalization. In following years many researchers tried face recog- nition schemes based on edges, inter-feature distances, and other neural net approaches. While several were successful on small databases of aligned images, none successfully addressed the more realistic problem of large databases where the location and scale of the face is unknown.

Kirby and Sirovich (1989) [2] later introduced an algebraic manipulation which made it easy to directly calculate the eigenfaces, and showed that fewer than 100 were required to accurately code carefully aligned and normalized face images. Turk and Pentland (1991) [1] then demonstrated that the residual error when coding using the eigenfaces could be used both to detect faces in cluttered natural imagery, and to determine the precise location and scale of faces in an image.

As the technology becomes mature, it has been widely in our daily life such as tracking faces in images and video analysing facial expressions detecting dysmorphic facial signs for medical diagnosis biometrics. I will list some of them in the next section.

In this article, we begin with detecting the location of keypoints on face images. This idea comes from our daily life because we can always identify our friends by their important feature on their face. It would be a good choice since this method really reduce the Computational complexity and could also achieve good Accuracy.

1.2 Potential Applications

Face recognition is closely related to many other domains, and shares a rich common literature with many of them. From our experiments, it would be clear that good testing performance could be achieved by predicting keypoint positions on face images. It provides us with a unique way to solve the problem.

For recognition of faces in video, face tracking is necessary, potentially in three dimensions with estimation of the head pose [15]. This naturally leads to estimation of the person's focus of attention[16,17] and estimation of gaze [18] which are important in human computer interaction for understanding intention, particularly in conversational interfaces. This tack can be completed by predicting keypoint positions. Correspondingly there is much work on person tracking [19] and activity understanding [20] which are important guides for face tracking and

for which face recognition is a valuable source of information.Recent studies have also begun to focus on facial expression analysis either to infer affective state [21] or for driving character animations particularly in MPEG-4 compression [22]. I believe that recent studies can be improved by this technology. The recognition of visual speech (i.e. lip-reading, particularly for the enhancement of acoustic speech recognition) is also a burgeoning face image processing area [23].

For example, The Face Recognition Technology (FERET) tests from Jonathan Phillips [26] provided an early benchmark of face recognition technologies.Phillips has continued the evaluation of face systems for US government agencies in the Face Recognition Vendor Tests [27]. This report provides an excellent independent evaluation of three state-of-the-art systems with concrete performance figures. The report highlights the limitations of current technology ,while under ideal conditions performance is excellent, under conditions of changing illumination, expression, resolution, distance or aging, performance falls off, in some cases dramatically. Current face recognition systems are not very robust yet against deviations from the ideal face image acquisition but there is continual performance improvement. For there published paper, the usage of getting keypoint positions on face images is obviously but not widely.

In addition, we can also apply this system to Access Control, Identification Systems, Surveillance, Pervasive Computing. Furthermore, the application should not be limited to face recognition. Although we will just discuss face recognition in this article, this idea of extracting several features to represent a complex picture may be widely used.

2 REVIEW OF THE STATE-OF-THE-ART

2.1 hand-crafted features

Most current face verification methods use hand-crafted features. Moreover, these features are often combined to improve performance, even in the earliest LFW contributions. The systems that currently lead the performance charts employ tens of thousands of image descriptors [28,29,30]. In contrast, our method is applied directly to RGB pixel values, producing a very compact yet sparse descriptor.

2.2 Deep neural nets

Deep neural nets have also been applied in the past to face detection [31], face alignment [12] and face verification [32, 33]. In the unconstrained domain, Huang et al. [33] used as input LBP features and they showed improvement when combining with traditional methods. In our method we use raw images as our underlying representation, and to emphasize the contribution of our work, we avoid combining our features with engineered descriptors. We also provide a new architecture, that pushes further the limit of what is achievable with these networks by incorporating 3D alignment, customizing the architecture

for aligned inputs, scaling the network by almost two order of magnitudes and demonstrating a simple knowledge transfer method once the network has been trained on a very large labeled dataset.

2.3 Metric learning methods

Metric learning methods are used heavily in face verification, often coupled with task-specific objectives [34, 10, 9]. Currently, the most successful system that uses a large data set of labeled faces [29] employs a clever transfer learning technique which adapts a Joint Bayesian model [9] learned on a dataset containing 99,773 images from 2,995 different subjects, to the LFW image domain. Here, in order to demonstrate the effectiveness of the features, we keep the distance learning step trivial.

2.4 Face Detection

Naturally, before recognizing a face, it must be located in the image. This topic is very important and popular among researchers in recent years. In some cooperative systems, face detection is obviated by constraining the user. Most systems use a combination of skin-tone and face texture to determine the location of a face and use an image pyramid to allow faces of varying sizes to be detected. Increasingly, systems are being developed to detect faces that are not fullfrontal [36]. Cues such as movement and person detection can be used [35] to localize faces for recognition. Typically translation, scale and in-plane rotation for the face are estimated simultaneously, along with rotation-in-depth when this is considered.

2.5 Face Recognition

There is a great diversity in the way facial appearance is interpreted for recognition by an automatic system. Currently a number of different systems are under development, and which is most appropriate may depend on the application domain. A major difference in approaches is whether to represent the appearance of the face, or the geometry. Brunelli and Poggio [37] have compared these two approaches, but ultimately most systems today use a combination of both appearance and geometry. Geometry is difficult to measure with any accuracy, particularly from a single still image, but provides more robustness against disguises and aging. Appearance information is readily obtained from a face image, but is more subject to superficial variation, particularly from pose and expression changes. In practice for most purposes, even appearance-based systems must estimate some geometrical parameters in order to derive a 'shape-free' representation that is independent of expression and pose artefacts [38, 39]. This is achieved by finding facial landmarks and warping the face to a canonical neutral pose and expression. Facial features are also important for geometric approaches and for anchoring local representations.

3 OBJECTION AND CONTRIBUTIONS

The goal of the project is to locate specific keypoints on face images.And we will build an algorithm, that, given an image of a face, automatically locates where these keypoints are located.

In our project, we try exploiting machine learning algorithms taught in the course, including linear regression, KNN, logistic regression, neutral network as well as tree- based methods. Of course, all above algorithms we apply in practice are quite basic, and have been developed relatively mature especially the Convolutional Neural Network, namely CNN. However, our contributions are that we compare these methods and explain the advantage and disadvantage of using these methods for the project problem. What's more, we spend a lot of time on data preprocessing. Because we think in the case where the methods have been fixed, the better the data is process, the better result we can get. This work is unique and exactly important. And these ideas are likely to be used by other people to better their experimental results. What's more, in the end of our article, we put forward some further improvement which may inspires new ideas with regard to the detection of keypoints.

In this experiment, the main language we use is *Python*.

In addition, we use these packages to achieve our algorithm:

(1)Numpy

(2)Scipy

(3)Scikit

(4)Theano

(5)Keras

Let's start by describing some of the work that we have done.

4 DATA PREPROCESSING

4.1 original data

Each predicted keypoint is specified by an (x,y) real-valued pair in the space of pixel indices. There are 15 keypoints, which represent the following elements of the face:

$left_eye_center, right_eye_center,$

$left_eye_inner_corner, left_eye_outer_corner,$

$right_eye_inner_corner, right_eye_outer_corner,$

$left_eyebrow_inner_end, left_eyebrow_outer_end,$

$right_eyebrow_inner_end, right_eyebrow_outer_end,$

$nose_tip,$

$mouth_left_corner, mouth_right_corner,$

$mouth_center_top_lip, mouth_center_bottom_lip$

Left and right here refers to the point of view of the subject.

In some examples, some of the target keypoint positions are missing (encoded as missing entries in the csv, i.e., with nothing between two commas).

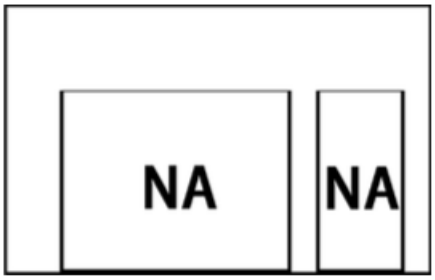


Figure 1: Sketch Map

The input image is given in the last field of the data files, and consists of a list of pixels (ordered by row), as integers in (0,255). The images are 96x96 pixels.

Then we download the data files from Kaggle websites:

training.csv: list of training 7049 images. Each row contains the (x,y) coordinates for 15 keypoints, and image data as row-ordered list of pixels.

test.csv: list of 1783 test images. Each row contains ImageId and image data as row-ordered list of pixels.

4.2 spilt the data

We spilt the *training.csv* into *keypoint.csv* and *im.csv*. Each row of *keypoint.csv* contains the (x,y) coordinates for 15 keypoints, while that of *im.csv* contains the image data as row-ordered list of pixels.

To check the data integrity, we have a close-up view of the data stored in *keypoint.csv*, and find the dataset has some problems. It is two different datasets put together into one for this competition, i.e., the target keypoints positions of the first 2284 examples are almost complete and the latter 4765 examples only contain the (x,y) coordinates for 4 keypoints while the other 11 keypoints positions are all missing. To illustrate, a sketch map is displayed below:

So it is important to split the data and train seperate models. We firstly split it into one dataset with 15 keypoints, i.e. section A, and one with 4 keypoints, i.e. section B, just as illustrated below:

However, at the half-way through our project, we come up with a better way to spilt the data. Actually, after visualizing the image data in *im.csv*, we find there is no significant difference between the data corresponding to section A and section B. Therefore, we just spilt the data into one dataset with 4 keypoints, i.e. section D, and one with 11 keypoints, i.e. section E, which is illustrated as follow:

As for the new way to spilt the data, there are 7049 examples in $keypoint_4f.csv$ and $im_4f.csv$, 2284 examples in $keypoint_11f.csv$ and $im_11f.csv$. Compared to the former way, we just train a model for the 4 keypoints, and the other for the rest 11 keypoints.

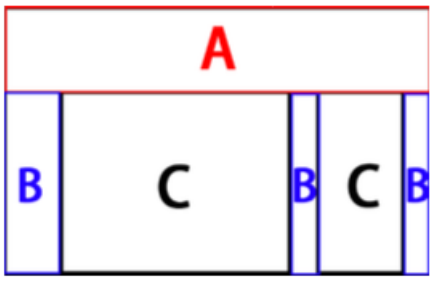


Figure 2: One Way to Spilt Data Set

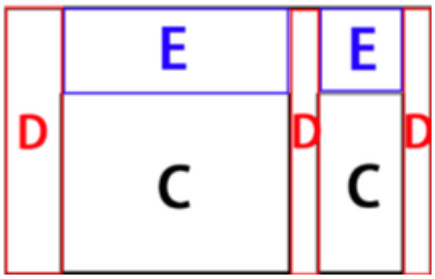


Figure 3: One Way to Spilt Data Set

In this way, we can make a better prediction for these 4 keypoints. Of course, we need to notice there is some inter- relation among the 15 keypoints position, for example, the nose is likely to fall on the perpendicular bisector of the line-segment connecting two eyes with good probability. Considering the interrelation, the prediction accuracy for the rest 11 keypionts may slightly decline for the absence of the 4 keypoints. But in our models, i.e. linear regression, K- NN, logistic regression, SVM, Neural network, we seldom make use of the inter-relation among the 15 keypoints, thus, the second way is more reasonable. Keep in mind when the inter-relation is included in the model, the first way to spilt the data may be better.

Take a notice that, apart from section C, there are also few target keypoint points are missing in section A and B. In order to facilitate subsequent analysis process, we substitude the missing data with the corresponding column mean.

Moreover, since we don't have the ground-truth for *test.csv*, for the paper writing purposes, we split the training set into 90% and 10% respectively. And the 10% held-out data can be served as the testing data to evaluate the algorithms. In other words, *test.csv* is put aside in our project. Therefore, we divide the

$keypoint_4f.csv$ into $keypoint_train_4f.csv$ and $key-point_test_4f.csv$;

$im_4f.csv$ into $im_train_4f.csv$ and $im_test_4f.csv$;

$keypoint_11f.csv$ into $keypoint_train_11f.csv$ and $key-point_test_11f.csv$;

$im_11f.csv$ into $im_train_11f.csv$ and $im_test_11f.csv$.

Figure 4: the first image

4.3 Visualize the data

For better understanding, we visualize the first image in $im_train_4f.csv$. We first reshape these 9216 integers into a 96x96 matrix, then use R's image function:

We can the add some keypoints(from the corresponding file $keypoint_train_4f.csv$) to check if everything is correct so far. Let's color the coordinates for the eyes and nose as an example:

We can see the positions of the centers of the nose and two eyes are all labelled correctly.

Another good check is to see how variable is our data. For example, where are the nose centers in the 7049 images?

We can see from above, most nose points are concentrated in the central region(as expected), but there are quite a few outliers that deserve further investigation, as they could be labeling errors. Looking at one extreme example:

In this case there's no labeling error, but this shows that not all faces are centralized as one might expect.

4.4 Face Detector

Just as Fig.7 indicates, not all faces are centralized, thus we decide to build a face detector first to find bounding boxes for faces in the image. We get the result as follow via opencv:

We can see from above the method works quite well. However, when we apply the method to the training data, the result is really upset:

This is because the size of images data is 96x96, which indicates the images are badly out of focus.

To get a better result, we may train a neural network separately to detect the face in the image, which need further exploration. Or otherwise, we can

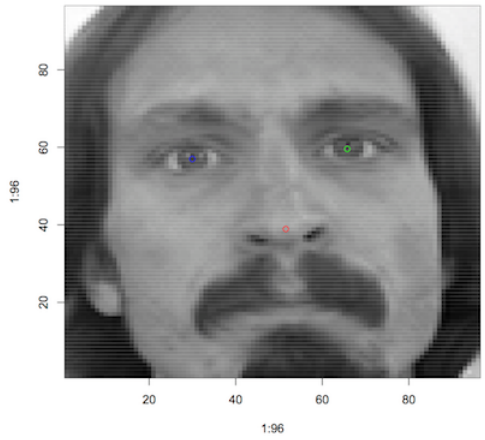


Figure 5: add keypoints to the image

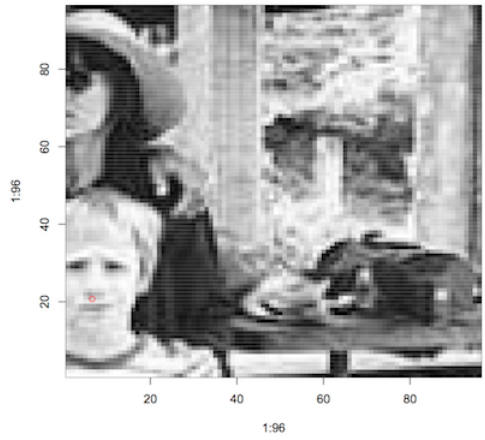


Figure 6: an extreme example

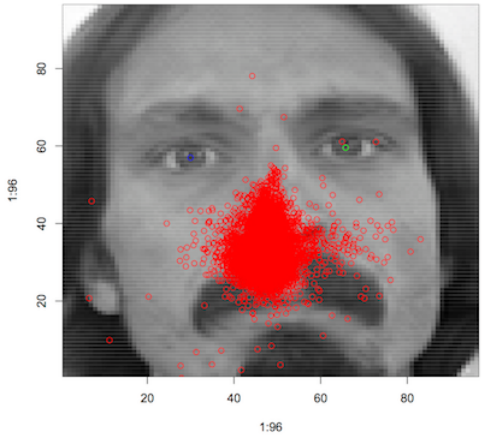


Figure 7: centers of each nose in the 7049 images

Figure 8: find bounding box for face

Figure 9: find bounding box for face

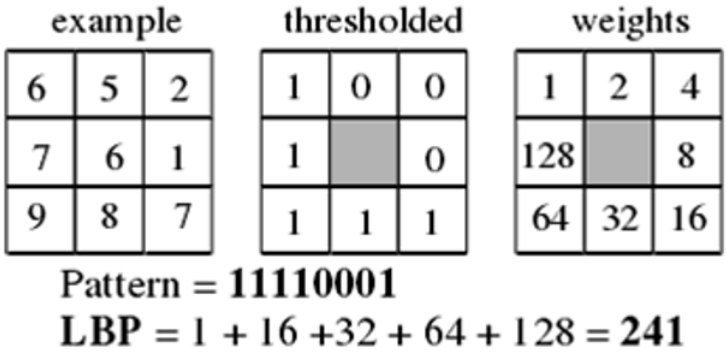


Figure 10: Calculation of LBP

use opencv to detect the keypoints of the image directly. However, opencv doesn't utilize the training dataset and it's a mature model that has been trained. Therefore, this method is of little significance as to our project.

4.5 LBP

LBP (Local Binary Pattern) is an operator used to describe the local texture features of images. It has the advantages of rotation invariance and gray invariance. It was first proposed by T. Ojala, M. Pietikainen, and D. Harwood in 1994 for texture feature extraction. Moreover, the extracted features are local texture features of the image.It has since been found to be a powerful feature for texture classification; it has further been determined that when LBP is combined with the Histogram of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. A comparison of several improvements of the original LBP in the field of background subtraction was made in 2015 by Silva et al. A full survey of the different versions of LBP can be found in Bouwmans et al..

4.5.1 Description of LBP features

The original LBP operator is defined as a 3 * 3 window, and the window center pixel is set as a threshold, compared to its 8 neighbors (on its left-top, left-middle, left-bottom, right-top, etc.). If the neighbor's value is greater than the center pixel value, then write "1", otherwise write "0". In this way, we can obtain an 8-digit binary number (which is usually converted to decimal for convenience), which is the LBP value of the center pixel. This value is always used to reflect the texture information for this area. For better understanding, we give a simple case to calculate LBP value just as shown below:

So that the pixel value of center point is replaced by 241. Note that it doesn't mandate a particular order in which LBP value is calculated. We only need to

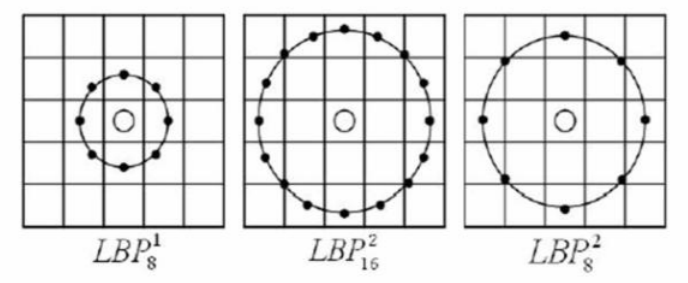


Figure 11: Circular LBP Operator

maintain the same order in the same process.

The calculation formula of LBP is

$$LBP = \sum_{p=0}^{P-1} s(g_p - g_c)2^p$$

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

where g_p is the neighbor's pixel value and g_c is the center pixel value

After the original LBP was put forward, the researchers constantly propose a variety of improvements and optimization.

4.5.2 Improved version of LBP

(1)Circular LBP Operator

The main drawback of the basic LBP operator is that it only covers a small area with a fixed radius, which obviously can't meet the needs of different size and frequency texture. In order to adapt to different scale of texture features as well as achieve the requirements of gray and rotation invariance, Ojala et al. improves the LBP operator: extend the 3x3 neighborhood to any size of neighborhood, and replace the square neighborhood with a circular neighborhood. The improved LBP operator allows any number of pixels in a circular neighborhood with the radius R.

Note: LBP_p^r, p is the number of sampling pixels; r is the radius of circular neighborhood.

(2)LBP Rotation Invariant Mode

As can be seen from the definition of LBP, LBP operator is gray-invariant, but not rotation-invariant. Rotation of the image can result in different LBP values.

Maenpaa et al. extends the LBP operator and proposed a rotation invariant LBP operator, that is, rotating the circular neighborhood to obtain a series of

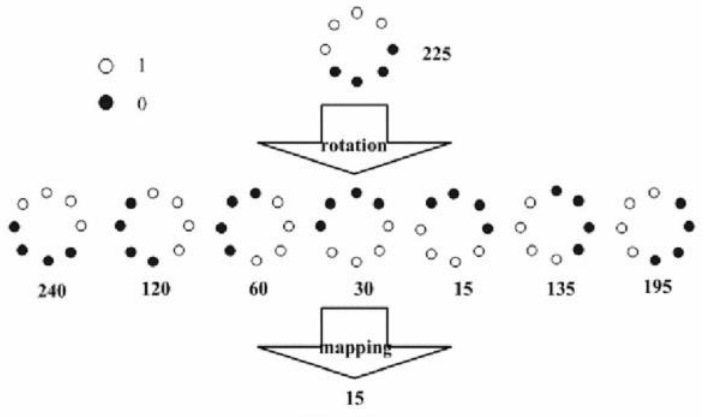


Figure 12: Diagram of Rotation Invariant LBP

initially defined LBP values, then taking the minimum value as the LBP value of the neighborhood.

Figure 3 shows the diagram of obtaining the rotation in- variant LBP. The number below the operator represents the corresponding LBP value. After the treatment of rotation, the final rotation-invariant LBP value is 15. In other words, the 8 LBP modes shown in the figure correspond to the same rotation-invariant LBP pattern 00001111.

Of course, there are still many other improvements of LBP, but we'll not described in this report. Interested people can learn more about it by yourselves.

4.5.3 Steps of Extracting the LBP Feature Vector

The LBP feature vector, in its simplest form, is created in the following manner:

- Divide the examined window into cells (e.g. 16x16 pixels for each cell).
- For each pixel in a cell, compare the pixel to each of its 8 neighbors (on its left-top, left-middle, left- bottom, right-top, etc.). Follow the pixels along a circle, i.e. clockwise or counter-clockwise.
- Where the center pixel's value is greater than the neighbor's value, write "0". Otherwise, write "1". This gives an 8-digit binary number (which is usually converted to decimal for convenience).
- Compute the histogram, over the cell, of the fre- quency of each "number" occurring (i.e., each com- bination of which pixels are smaller and which are greater than the center). This histogram can be seen as a 256-dimensional feature vector.
- Optionally normalize the histogram.

Figure 13: post_LBP

- Concatenate (normalized) histograms of all cells. This gives a feature vector for the entire window.

The feature vector can now be processed using the SVM or some other machine-learning algorithm to classify images. Such classifiers can be used for face recognition or texture analysis.

It is obvious that the LBP operator can get a LBP value at each pixel point. Then, after extracting the LBP feature vector for an image(each pixel records the gray value), the LBP feature we get is still a "picture" (each pixel records the LBP value) just as below:

4.6 PCA

In our project, the dimension of the image data is 9216 (96x96), which is rather large and adds to the difficulty of data analysis. And we naturally think of dimensionality reduction because of its advantages in the case of processing high-dimensional data:

- low-dimensional data is easier to handle;
- correlation features of the data set, especially the important features can stand out;
- remove the data noise;
- reduce the cost of the algorithm and more conducive to solving the problem when computational resources are limited;

PCA, Factor Analysis and Independent Component Analysis are the common algorithms for dimensionality reduction. Among them, PCA is the most widely used method and is also adopted by us.

4.6.1 Principle of PCA

PCA is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorre- lated variables called principal components. The number of principal

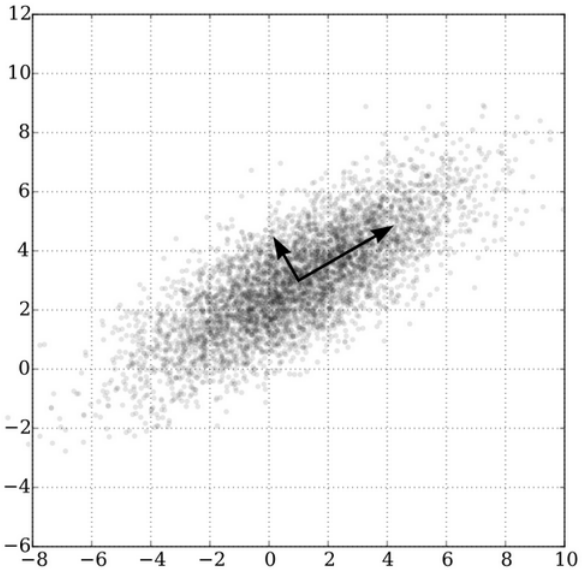


Figure 14: Principle Sketch of PCA

components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possi- ble variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an uncorrelated orthogonal basis set. And PCA is sensitive to the relative scaling of the original variables.

4.6.2 Implementation Process of PCA

On the basis of linear algebra, we can obtain the values of these principal components by analysing the covariance ma- trix of the data set and its eigenvalue. Once the eigenvectors are obtained, we can retain the largest N vectors. The data set can then be converted to the new space by multiplying these N eigenvectors.

More detailly, we can describe the implementation pro- cess of PCA as follows:

- normalize the data;
- obtain the covariance matrix;
- calculate the eigenvalues and eigenvectors of the covariance matrix;
- sort the eigenvalues from large to small;

- retain the largest N eigenvectors;
- convert the data into the new space constructed by the above N eigenvectors.

It should be stressed that the retained eigenvectors need to cover more than 90% of the information. And in practical application, actually, we retain the largest 256 eigenvectors, which cover more than 95% of the information.

5 ALGORITHMS AND ANALYSIS

In this part, we will talk about the algorithms we used to recognize the keypoints on the faces and their task completions in order to explore their differences and find which would be the most suitable one. In addition, all the codes are also provided in case you want to run it by yourselves. However, please note that there is randomness among different experiments, which means you should not be surprised if you cannot obtain the same result.

In this section, I just focus on the specific algorithm itself. And in the next section, the contrast among different algorithms will be proposed. And we will describe our step in P ython to build Neural network and Convolutional neural network since they are the most challenging aspect in the experiment.

First of all, I list all the results here in tables:

	RMSE1	RMSE2
Knn	3.375	2.346
Linear	4.513	6.020
Lasso	3.558	2.979
Elastic	4.044	2.959
Ridge	8.464	2.609
Decision tree	3.745	4.101
Neural network	2.923	2.875
CNN	1.972	2.086

5.1 Knn

We prefer to start our discussion among the algorithms from kNN(k-NearestNeighbor) because it is very simple to understand and visualize.

KNN regression can be used in cases where the data la- bels are continuous rather than discrete variables. The label assigned to a query point is computed based the mean of the labels of its nearest neighbors. The advantages lie in its simplicity and effectiveness. Theoretically, the more training instances we can provide, the better the performance of Knn algorithms will be. Luckily, there is a big data set for us to train so RMSE1 has been reduced to about 3.375 with $k = 5$.

Talking about the weak points about Knn. I want to mention the Curse of Dimensionality, which significantly cut down the power of kNN algorithm. In this case, this problem is obvious because we have 30 and 8 dimensions in

each data sets. Although the huge number of instances reduce the Curse of Dimensionality, to some extent, it is still rigorous.

However, Knn receives 2.346 RMSE2 on a the data set of 8 dimensions while 3.375 RMSE1 on a data set of 30 dimensions . It uncovers the Curse of Dimensionality, to some extent.

5.2 linear regression

We fit a linear model with coefficients $w = (w_1, \ldots, w_p)$ to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted by the linear approximation. Mathematically it solves a problem of the form

$$min||\mathbf{X}w - y||_2^2$$

where $\mathbf{X}$ stands for the observed variables in the dataset and y represents the responses.

Obviously, linear regression is too simple to solve the problems in real world unless the relationships between observed variables and responses are linear itself. However, there is no evidence to prove this kind of bind. Therefore, we received 8.4634 RMSE1, which is the worst among all the methods we used in the experiment.

Considering another reason of high RMSE, we think the problem of overfitting is the biggest problem. In the following section, you will see how Lasso regression and Elastic Net regression improve the RMSE by overcoming the problem of too much parameters.

5.3 Lasso

The Lasso regression is a linear model that estimates sparse coefficients. It is useful in some contexts due to its tendency to prefer solutions with fewer parameter values, effectively reducing the number of variables upon which the given solution is dependent. For this reason, the Lasso and its variants are fundamental to the field of compressed sensing. Under certain conditions, it can recover the exact set of non-zero weights.

Mathematically, it consists of a linear model trained with ℓ_1 prior as regularizer. The objective function to minimize is:

$$min\frac{1}{2n_{samples}}||\mathbf{X}w - y||_2^2 + \alpha||w||_1$$

And it can be proved that solving for the lasso regression is equivalent to solve the following problem:

$$min\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2, \sum_{j=1}^{p}|\beta_j| \leq s$$

Lasso regression enjoys a good RMSE1 with only 3.559. Some parameters may be extra since sometimes we do not need such information to recognize

a face. On the other hand, parameter redundancy may cause the problem of overfitting. Lasso regression has a the ability to overcome overfitting and that is the reason why it is so popular among machine learning skills.

The choice of the α is very important because large α will squeeze almost all the parameters to zero, which obviously cannot meet our requirements, while small α does not have the ability to fulfill the task as a lasso regression, in contrast, it may perform as a normal linear regression.

The best way to choose α is cross validation. However, cross validation is not easily to do on this dataset because it is too much big so that long times would be taken to run the code. Luckily, we almost find the best choice (α =0.1). However, you should note that since there is no Theoretical guidance to α, so this may not be the unique choice.

5.4 Elastic Net

Elastic Net is a linear regression model trained with ℓ_1 and ℓ_2 prior as regularizer. This combination allows for learning a sparse model where few of the weights are non- zero like Lasso, while still maintaining the regularization properties of Ridge.

Elastic-net is useful when there are multiple features which are correlated with one another. Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.

A practical advantage of trading-off between Lasso and Ridge is it allows Elastic-Net to inherit some of Ridge's stability under rotation.

The objective function to minimize is in this case:

$$min \frac{1}{2n_{samples}} ||\mathbf{X}w - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$$

In elastic net there is two parameters to be adjusted, which significantly increases the difficulty of tuning. In our experiment the RMSE1 is 4.04. Therefore, we prefer to attribute the poor performance to parameters. Although we determine the parameters using cross validation, it is still hard to tune. Finally, the parameters are fixed with $\alpha = 0.1, \rho = 0.5$.

Compared with Lasso regression, Elastic Net achieves a higher test error. However, we cannot conclude that Elastic Net cannot compare with the Lasso regression. Instead, Lasso regression is strong enough to prevent the system from overfitting rather than.

5.5 Ridge Regression

Compared with linear regression ridge regression balance the error and variance by adding penalty. The parameters of ridge regression are obtained my minimize:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})_\lambda^2 \sum_{j=1}^{p}\beta_j^2$$

If $\lambda = 0$ it will degenerate into normal linear regression. In addition, it can be proved that solving for ridge regression is equivalent to solving the following problem:

$$min \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2, \sum_{j=1}^{p} \beta_j^2 \leq s$$

Ridge regression has closed form solution as a convex optimization problem. This property improved the speed of the code largely. However, ridge regression has huge gap between RMSE1 and RMSE2, as can be seen from the table above. This problem confused us a lot during the experiment. However, after dividing the data set again, we obtain a normal result for ridge regression with RMSE1 = 2.967 and RMSE2 = 3.104. In the table above we still report the previous RMSE because we want to mention this surprising result may due to luck but still need further investigation.

5.6 Decision tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- Use a white box model. If a given situation is ob- servable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model results may be more difficult to interpret.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalize the data well. This is called over- fitting. Mechanisms such as pruning (not currently sup- ported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.

The RMSE1 of decision trees is 3.75 with $k = 5$. We have the confidence to believe that overfitting would not occur with applying only $k = 5$ to such a big data set. However, decision tree may ignore the correlation between characters, which is an important property in this experiment.

5.7 Neural network

5.7.1 Package description

As you know Keras is a high-level neural networks library, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

In this experiment, we first study how to use the pack- ages Keras and Theano to build our neural network, then we choose two different neural networks to achieve our goal.They have a common feature is very precise, but the calculation is very large, we spent 5 days to debug them. Unfortunately, BP neural network has not reached an ideal result, but CNN's result is very good.

5.7.2 Steps to build neural network

Now, we'll give you an idea of how to build the BP neural network.

This has the same idea as handwritten number recognition, but is slightly different in some detail. This is a regression problem, not a classification problem. First we'd better to use the 'MSE' to replace the 'categorical crossentropy'. Second, we must use the linear function $f(x) = x$ as the activation function in the last layer, because the value of the other functions will be limited to 0 and 1, can not complete the predict. The other things we have done as follow:

1 We choose the different optimizer to build our neural network, such as 'Rmsprop' and 'SGD'.

2 We use 4 layers which has 300,150,50,8 neural for training.

3 After some times attempt, we choose the nb_epoch=500, batch_size=30.

4 we also use the dropout idea to get a model with stronger generalization ability.

5.7.3 Analysis

First, we construct M linear combinations of the input variables $x_1, \ldots, x_D$ in the form

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

where $j = 1, \ldots, M$ and the superscript (1) indicates that the corresponding parameters are in the first 'layer' of the network. The quantities a_j are known as activations. Each of them is then transformed using a differentiable, nonlinear activation function $h(x)$ to give

$$z_j = h(a_j)$$

The nonlinear function $h(x)$ are generally chosen to be sigmoidal functions such as the logistic sigmoid or the 'tanh' function.

Following these steps, we can extend the neural network to any hidden units and any layers we want.

The method we used to train the network is Backpropagation, I won't list the details mathematically, you can refer to [40] and check our code.

Neural network has the advantage of high classification rate and good learning skills. All the knowledges are evenly distributed in the system so that neural that can approximate any nonlinear system and it also has the ability of fault tolerance.

However, a lager number of parameters are needed for neural network. And there is no theoretical guidance to the choice of layers and hidden units, including other important parameters, which means that tuning is a pivotal step to a perfect neural network.

In the experiment we only receive 2.923 RMSE1, which is a terrible result obtained from neural network. Tuning must be one of the reasons but a further research should be requested.

5.8 Convolutional Network Network

5.8.1 Steps to build convolutional neural network

As we all know, the most popular technique in image recognition is CNN. So we spend the longest time in this method, now I will introduce the idea of our CNN.

- We use PCA method to reduce dimension from 96×96 to 16×16, This method saves 95% information, and makes the calculation faster.
- Because Keras requires the input of 4-dimensional vectors, so we need replace the $7049 \times 16 \times 16$ image as $7649 \times 16 \times 16 \times 1$ image.
- We use 2 convolution layers. The first layer we use 32 filters with dimension 5×5, then the pool_size is 2×2. In the second layer, we use the 8 filter with dimension 3×3, then the pool_size is 2×2. we also use the dropout method in each layer.
- After the convolution layer, we flatten it and add a normal hidden layer with 100 neural.
- Optimizer and loss function is similar with BP network.
- After some times attempt, we choose the nb_epoch=400, batch_size=50.

We try to install the cuda and use the GPU to compute it but fail, so we only use the CPU to train this neural network,it takes a very long time. Each test time are more than 10 hours, so the number of adjusting the parameters is not enough. But fortunately, the results are pretty good!

Following is a picture of a small demo run on our own computer. However, the whole code should be run on the server.

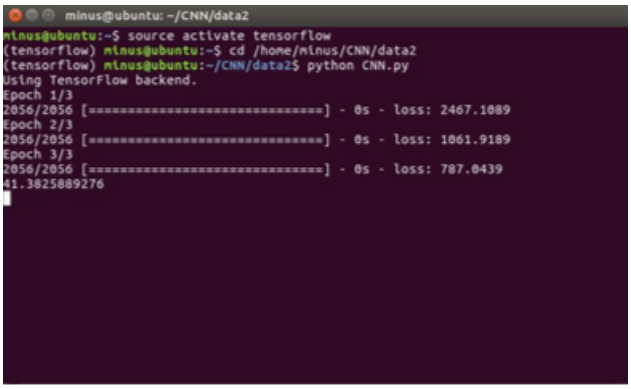


Figure 15: Principle Sketch of PCA

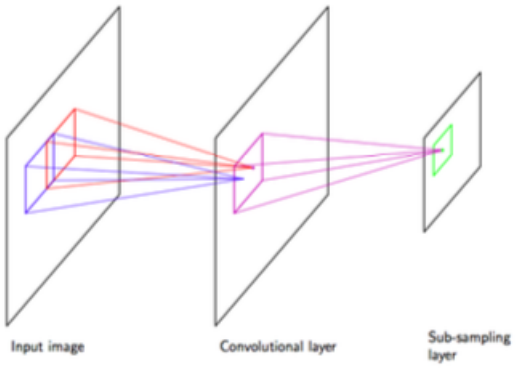


Figure 16: Principle Sketch of PCA

5.8.2 Analysis

The most successful approach is the convolutional neural network, which has been widely applied to image data. Convolutional neural network has three important mechanisms: (i) local receptive fields, (ii) weight sharing, and (iii)subsampling. The structure of a convolutional network is illustrated below:

In the convolutional layer the units are organized into planes, each of which is called a feature map. Units in a feature map each take inputs only from a small subregion of the image, and all of the units in a feature map are considered to share the same weight values. For instance, a feature map might consists of 100 units arranged in a 10×10 grid, with each unit taking inputs from a 5×5 pixel patch of the image. The whole feature map therefore has 25 adjustable weight parameters plus one adjustable bias parameter. Input values from a patch are linearly combined using the weights and the bias, and the result transformed by a sigmoidal nonlinearity.

The whole network can be trained by error minimization using backpropagation to evaluate the gradient of the error function. This involves a slight modification of the usual backpropagation algorithm to ensure that the shared-weight constraints are satisfied. Due to the use of local receptive fields, the number of weights in the network is smaller than if the network were fully connected. Furthermore, the number of independent parameters to be learned from the data is much smaller still, due to the substantial numbers of constraints on the weights.

Convolutional neural network is an excellent approach to image recognition because it makes the full use of local feature and shares soft weights, which is critical to images.

CNN achieves the best result with 1.972 RMSE1 and 2.086 RMSE2.

6 RESULTS OPTIMIZING

We re-run our codes on the pre-processed data by means of LBP and PCA, and get the better result:

	RMSE1	RMSE2
Knn	2.900	2.256
Linear	3.011	2.456
Lasso	3.009	2.352
Elastic	3.010	2.354
Ridge	3.011	2.356
Decision tree	3.827	2.354
Neural network	2.498	2.432
CNN	1.909	1.874

6.0.1 Analysis of PCA

The transformation of PCA is to project the original sample data (n-dimensions) into a k-dimensional(k<n) orthogonal coordinate system, discarding the information in other dimensions. In this way, PCA can extract the main influencing factors, revealing the properties of the data.

What's more, in fact, the variables in image data have a quite strong internal correlation. For instance, eyes symmetrically distribute on both sides of the nose. However, in our models, we ignore this correlation subconsciously, which makes the RMSE not very good.

But when we adopt PCA, we can remove the correlation among different variables because the project space is an orthogonal coordinate system. And the running results also strongly support this conclusion.

By the way, for PCA is a dimension-reduction algorithm, it will discard part of the information inevitably, but it does't hurt the important essentials. Because we retain 95% of the information in practice.

We originally didn't expect PCA to behave well, instead, we just want to reduce the time for running our code. Surprisingly, PCA reduces all the

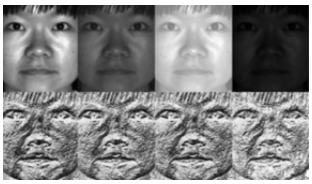


Figure 17: images under different illumination conditions

RMSE without exception. After the gun, we may conclude that 'Curse of Dimensionality' is a serious problem in this experiment. PCA has the ability to overcome overfitting and balance bias-variance.

6.0.2 Analysis of LBP

Facial features vary greatly from one individual to another, and even for a single individual, there is a large amount of variation due to 3D pose, size, position, viewing angle, and illumination conditions. LBP allows you to capture the details of the image very well. In fact, researchers can achieve the state-of-the-art level in texture classification by using it. Actually, LBP is kind of prudent with the monotonous change in gray-scale. We can see the LBP images of the same photo under different illumination conditions:

We can judge that LBP feature is not sensitive to light, in other words, it can eliminate the impact of illumination conditions effectively when we are locate the position of keypoints. Thus, there is no surprise that LBP data can reduce the root mean squared error significantly.

7 COMPARISON OF VARIOUS METHODS AND FURTHER IMPROVEMENT

In this section, we will focus on the differences among all the proposed methods above.

First of all, we will begin with convolutional neural network.(CNN) There is no doubt that CNN is the best solution to this problem with such a low RMSE. It has been proved that CNN performs well to image recognition both from theory and practice. The advantages include feature extraction and soft weight sharing have been discussed before, which is the unique properties of CNN. It is these characters that make CNN such a perfect approach. However, the disadvantage still need to be mentioned since this cannot be felt from these paper unless to run our code on computers. Very long time will be taken though the code is run on the server. It raises the problem of tuning because it is time consuming. Thence, we have reasons to believe that better RMSE can be achieved if suitable parameters are applied.

Secondly, Neural network doesn't get the RMSE as well as our expectation. Compared with CNN, neural network itself is not a specific method for image

recognition so this would not be surprising. However, we originally expect that neural network may perform better can Regression methods. Now taking a look back, neural network's failure may be attributed to overfitting and tuning. The method of 'Dropout' can avoid overfitting but their is no theoretical guidance to the setting of parameters. Beside its high RMSE compared with CNN, it is still time consuming to train neural network. Overall, we do not recommend neural network in face recognition experiment unless perfect parameters can be found.

Thirdly, decision tree is in my recommendation. You may question of my choice because this method doesn't have access to low RMSE. However, decision tree is very interpretable and visible and it can be used to understand and control the data set. In conclusion, we suggest applying decision tree to the data set first so that you will know what information do the data set conveys. Talking about the adjustments, there is no standard methods here since you can cut or add some branches to the tree as you wish.

Then, we applied a series of linear regression and penalized linear regression. These methods receive 'not bad' results expect linear regression. In practice, people won't just use linear regression to a problem unless they can prove a strong linearity. However, we should not abandon penalized linear regression since it is not computational consuming and you can get the result quickly though the code is not run on server. It is hard to make huge improvement because parameters are carefully selected.

Finally, Knn got a surprising result to our expectation. The data set is given in high dimensions, which is the theoretically inappropriate to KNN. The reason might come from the data size. From this experiment, it can be seen that there may exist a relation between dimensions and the amount of data, which need further investigation. However, we still insist not recommending Knn to other high dimension problems like image recognition unless theoretical guidance can be found.

8 Conclusions

Overall, we have applied ten different methods to our data set. CNN gets the best RMSE with only 1.972 but it is very time consuming. Decision tree takes advantage of interpretability though it does not have low RMSE. Linear Regression methods are simple enough and only takes a little time to train but they are not recommended for real world applications since we need to pay much more attention to RMSE rather than algorithm complexity.

Neural network and Knn both get surprising result and need further investigation. In this article, we only illustrate some supposes which should be verified in the future.

LBP allows us to capture the details of the image very well, which is important for image processing. Although some results show that LBP may make RMSE worse, most experiments display a good trend. Therefore, LBP is also included as a good solution to this experiment. Moreover, LBP won't put too much additional information to the data set so that extra computational burden won't

exist.

PCA is a dimension-reduction algorithm, the effect of this method on the data is hard to predict. However, it greatly reduces the dimensions and cut down the consuming time. In this experiment, PCA receives a excellent result. Especially for neural network, this proves our suppose of overfitting, to some degree.

In our experiment, we simply combine LBP and PCA together. However, further improvements can be made by applying these methods independently after a full study of their mechanisms.

References

[1] M.Turk & A.Pentland: Eigenfaces for recognition, *J.Cog.Neuroscience*, Volume 3, (1991)

[2] M.Kirby & L.Sirovich: Application of the Karhunen-loeve procedure for the characterization of human faces, *Springer- Verlag, Berlin*, 1989

[3] T.Kohonen: Self-organization and Associative Memory, *IEEE*, 1990

[4] Nitish Srivastava & Geoffrey E. Hinton & Alex Krizhevsky & Ilya Sutskever & Ruslan Salakhutdinov: Dropout: A simple way to prevent neural network from overfitting, *Journal of Machine learning Research*, Pages:1929-1958 2015

[5] T.Berg & P.N.Belhumeur:om-vs-pete classifiers and identity-preserving alignment for face verification,*BMVC*, 2012

[6] Zhu.X & Ramanan D: Face detection, pose estimation, and landmark localization in the wild, *CVPR*, 2012

[7] N.Kumar & A.C.Berg & P.N.Belhumeur & S.K.Nayar: Attribute and smile calssifiers for face verification, *ICCV*, 2009

[8] Yaniv Taigman & Ming Yang & Marc' Aurelio Ranzato: Deep-Face: Closing the Gap to Human-Level Performance, *CVPR*, 2014

[9] D.Chen & X.Cao & L.Wang & F.Wen & J.Sun: Bayesian face revisited: A joint formulation, *ECCV*, 2012

[10] Y.Taigman & L.Wolf :Leveraging billions of faces to overcome performance barriers in unconstrained face recognition,

[11] Y.Bengio: Learning deep architectures for AI, *Foundations and Trends in Machine Learning*, 2009

[12] Y.Sun & X.Wang & X.Tang: Deep convolutional network cascade for facial point detection, *CVPR*, 2013

[13] T.Ahonen & A.Hadid & M.pietikainen: Face description with local binary patterns: Application to face recognition, *PAMI*, 2006

[14] Karen Simonyan & Andrew Zisserman: Very deep convolutional networks for large-scale image recognition, *ICLR*, 2005

[15] M. La Cascia & S. Sclaroff & V. Athitsos: Fast, Reliable Head Tracking under Varying Illumination: An Approach Based on Registration of Texture-mapped 3D Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, April 2000

[16] P. de Cuetos, C. Neti & A. Senior: Audio-visual intent to Speak Detection for Human-computer Interaction *In proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* 2000

[17] James M. Rehg & Kevin P. Murphy & Paul W. Fieguth: Vision- based Speaker-detection using Bayesian Networks. *In Proceedings of Computer Vision and Pattern Recognition*, Volume 2, pages 110-116, 1999

[18] Y. Matsumoto & A. Zelinsky: An Algorithm for Real-time Stereo Vision Implementation of Head Pose and Gaze Direction Measurement. *In IEEE International Conference on Face and Gesture*, page 499, 2000

[19] Second International Workshop on Performance and Evaluation of Tracking and Surveil- lance. *IEEE* ,December 2001

[20] T. Tan: Second IEEE International Workshop on Visual Surveillance. *IEEE*, 1999

[21] Rosalind W. Picard: Affective Computing. *MIT Press*, 2009

[22] E. Petajan: The Communication of Virtual Human Faces using mpeg-4 Tools. *In Inter- national Symposium on Circuits and Systems,* Voluume 1, pages 307-310, 2010

[23] Chin-Seng Chua & FengHan & Yeong-KhingHo: 3DHumanFace Recognition using Point Signature. *In International Conference on Face and Gesture Recognition*, Voluume 1, pages 307-310, 2010

[24] P. Jonathon Phillips & Patrick J. Rauss & Sandor Z. Der. FERET: (Face Recognition Technology) Recognition Algorithm Development and Test Results. Technical Report ARL-TR-995, *Army Research Laboratory*, October, 1996

[25] F. Prokoski: History, Current Status, and Future of Infrared Identification. *In Proceedings of IEEE Workshop on Computer Vision Beyond the Visible Spectrum: Methods and Applications*, pages 5-14, June 2000. Facial Thermogram

[26] P. Jonathon Phillips & Hyeonjoon Moon & Patrick Rauss & Syed A. Rizvi: The FERET September 2009 Database and Evaluation Pro- cedure. *In Josef Bigun, Gerard Chollet, and Gunilla Borgefors, editors, Audio- and Video-based Biometric Person Authentication, volume 1206 of Lecture Notes in Computer Science,* pages 395-402. Springer, March 2010.

[27] Duane M.Blackburn & Mike Bone & P.Jonathon Phillips: Facial Recognition Vendor Test 2000 Evaluation Report. *Technical Report, Department of Defence Counterdrug Technology Development Program Office,* February 2011

[28] O. Barkan, J. Weill & L. Wolf & H. Aronowitz: Fast high dimen- sional vector multiplication face recognition, *ICCV*, 2013

[29] X. Cao & D. Wipf & F. Wen & G. Duan & J. Sun: A practical transfer learning algorithm for face verification, *ICCV*, 2013

[30] D.Chen & X.Cao & F.Wen & J.Sun: Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification, *CVPR*, 2013

[31] M.Osadchy & Y.LeCun & M.Miller: Synergistic face detection and pose estimation with energy-based models, *JMLR*, 2007

[32] S. Chopra & R. Hadsell & Y. LeCun: Learning a similarity metric discriminatively, with application to face verification, *CVPR*, 2005

[33] G. B. Huang & H. Lee & E. Learned-Miller: Learning hierarchical representations for face verification with convolutional deep belief networks, *CVPR*, 2012

[34] Jochen Triesch & Christoph von der Malsburg: Fisher vector faces in the wild, *BMVC*, 2013

[35] K. Simonyan & O. M. Parkhi & A. Vedaldi & A. Zisserman: elf-organized Integration of Adaptive Visual Cues for Face Tracking. *In International Conference on Face and Gesture Recognition, IEEE*, 2015

[36] G. J. Edwards & C. J. Taylor & T. F. Cootes: Interpreting Faces using Active Appearance Models. *In International Conference on Face and Gesture Recognition*, April 2012

[37] Roberto Brunelli & Tomaso Poggio: Face Recognition: Features versus Templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2013

[38] Ian Craw & Peter Cameron: Face Recognition by Computer *Proceedings of the British Machine Vision Conference, Springer Verlag*, 2012

[39] G.J.Edwards & C.J.Taylor & T.F.Cootes: Interpreting Faces using Active Appearance Models. *In International Conference on Face and Gesture Recognition* ,April 2012

[40] Christopher M.Bishop: Pattern recognition and Machine learning, *Springer* 2006