Deep Learning approach for private identification

Author

November 2018

Abstract

In this era of technology, the development of machine becomes the most important problem for engineer. That leads to another problem that, teaching the machine to learn from experience. And Artificial Intelligence (AI) was born. In this report, we will apply Machine Learning – a features of AI to verify faces on images input. By using pre-trained model frame network caffe we built a net that can distinguish the ID with the owner to figure out whether that two faces are the same person. The processing network frame includes many layers: Convolution layers, Pooling layers, Fully connected layers. Throughout the net, the image will be processed by the algorithm and make a comparison between two images by calculating the normalization of two processed images. By applying machine learning, we can understand about how Big Data and AI works for future improvement.

Acknowledgement

First and foremost, we would like to sincerely thank our Professor, Prof. Dr.Le Tien Thuong because of the chance he gave us to study and research in the image processing field. Not only directly guiding us but he also give as enthusiastic support and dedicated encouragement. We always respect the guidance, professional suggestion.

Next, we would like to thank our lecturers at HCMUT who imparted valuable knowledge in the time when we were being in class. Beyond imparting knowledge, these lecturers shared their valuable experiences and gave us sincere advices when we had questions. All things have helped us to increase our understanding and use them for our following jobs in future.

Besides, we also give our heartfelt thanks to the encouragement and support of our great family, especially our parents during the study period at HCMUT. The concern, anxiety and great sacrifices of parents always motivate us to strive on our learning path.

Finally, we would like to thank all our friends who has encouraged and given advices throughout our times studying in the Ho Chi Minh City University of Technology.

Ho Chi Minh City, December 6, 2018

NGUYEN TRUONG GIANG

TA QUOC VINH

Contents

List of Figures

List of Tables

CHAPTER 1

Artificial Intelligence & Machine Learning

1.1 Overview

Nowadays, the technology has been improved so far, belongs with the development of Computer Science, the branch that called *Artificial Intelligence (AI)* was created as intelligent as human being.

"The science and engineering of making intelligent machines, especially intelligent computer program." - *John McCarthy, father of Artificial Intelligence.*

What is Artificial Intelligence? Artificial Intelligence is a way of making a computer, a computer-controlled robot, or a software think intelligently, in the similar manner the intelligent humans think. AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

Artificial intelligence was found as an academic discipline in 1956, and in the years since has experienced several waves of optimism, followed by disappointment and the loss of funding (known as an "AI winter"), followed by new approaches, success and renewed funding. For most of its history, AI research has been divided into subfields that often fail to communicate with each other. These sub-fields are based on technical considerations, such as particular goals (e.g. "robotics" or "machine learning"), the use of particular tools ("logic" or artificial neural networks), or deep philosophical differences. Subfields have also been based on social factors (particular institutions or the work of particular researchers). [2]

In this report, we apply Machine Learning (ML) − a filed which is raised out of Artificial Intelligence. Machine Learning is programming computer to optimize a performance criterion using example data or past experience. We define a trained model, then the using model may be predictive to make predictions in the future, or descriptive to gain knowledge from data or both. [1]

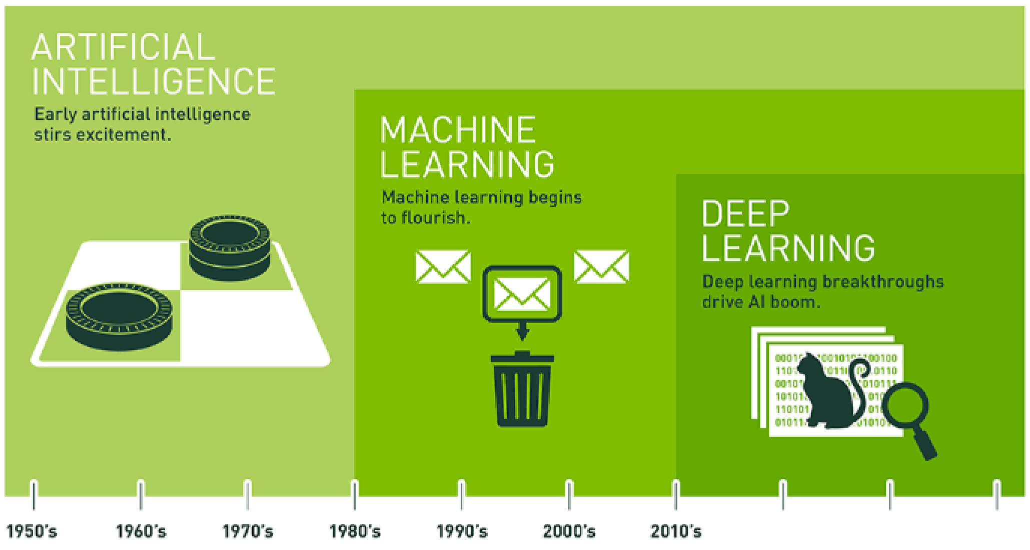

ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's

Figure 1.1: Different between AI, Machine Learning and Deep Learning

1.2 Machine Learning

1.2.1 Definition

In definition of Andrew Ng – co-founder and led Google Brain and was a former Vp & Chief Scientist as Baidu, Machine Learning is the idea that there are generic algorithms that can tell you something interesting about a set of data without you having to write any custom code specific to the problem. Instead of writing code, you need data to the generic algorithm and it builds its own logic based on the data. The name machine learning was coined in 1959 by Arthur Samuel. Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field:

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E”.

This definition of the tasks in which machine learning is concerned offers a fundamentally operational definition rather than defining the field in cognitive terms. This follows Alan Turing’s proposal in his paper “Computing Machinery and Intelligence”, in which the question “Can machines think?” is replaced with the question “Can machines do what we can do?”. In Turing’s proposal the various characteristics that could be possessed by a thinking machine and the various implications in constructing one are exposed. [2]

Consider example playing checker:

- E = the experience of playing many games of checkers.
- T = the task of playing checkers.
- P = the probability that the program will win the next game.

The application of ML methods to large databases is called data mining. But machine learning is not just a database problem; it is also a part of artificial intelligence. To be intelligent, the system that is in a changing environment should have the ability to learn. If the system can learn and adapt to such changes, the system designer need not foresee and provide solution for all possible situations. [1]

1.2.2 Type of Machine Learning

There are 3 types of machine learning considered in this section:

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Supervised learning

Classification: Discrete valued output (0 and 1). The example to understand classification overview: there are 2 classes: low-risk and high-risk customer. The information about a customer makes up the input to the classifier whose task is to assign the input to one of two classes. After training with the past data, the classification rule learned may be formed suitably. After that, the main application is prediction: Once we have the rule that fits the past data, if the future is similar to the past, then we can make correct prediction for novel instances. Given a new application with a certain income and savings, we can easily decide whether it is low-risk or high-risk.

Regression: Predict continuous value output. A regression problem is when the output variables is a real value, such as “Rupees” or “height”. In supervised learning, the system tries to learn from the previous examples that are given. Mathematically, supervised learning is where you have both input variables (X) and output variables (Y) and can use an algorithm to derive the mapping function from input to output:

$$Y = f(X)$$

Unsupervised learning

In unsupervised learning, there is no such supervised and we only have data input. Mathematically, unsupervised learning is when you only have input data (X) and no corresponding output variables (Y). The aim the to find the regularities in the input. There is a structure to the input space such that certain patterns occur more than others, and what generally happen and what does not. Unsupervised learning can be divided into Association and Clustering - Association: An association rule learning problem is where you want to discover rules that describes large portions of your data, such as “people buy X also tent to buy Y”. - Clustering: A clustering problem is where you want to discover the inherent grouping in the data, such as grouping customer by purchasing behavior.

Reinforcement learning

A computer program will interact with a dynamic environment in which it must perform a particular goal (such as playing a game with an opponent or driving a car). The program is provided feedback in terms of rewards and punishments as it navigates its problem space. Using this algorithm, the machine is trained to make specific decisions. It works this way: the machine is exposed to an environment where it continuously trains itself using trial and error method.

1.3 Purpose of this project

In this project, we using unsupervised learning to make a convolutional neural network (CNN) system that can matching a picture that capture by a camera and distinguish whether that is a right person's ID card. By far, we can learn from that how exactly ML working to help identify people. By using pre-trained model that given from Caffe framework included weight and bias that have already trained. We can make the project easier to understand and reduce training time for our neural network.

1. Capturing and detect student's face.
2. Capturing their ID card.
3. Detecting ID number on the card.
4. Detecting face on ID card.
5. Matching whether two detected face is from one person.

CHAPTER 2

Convolutional Neural Network

Convolutional Neural Network are very similar to ordinary Neural Networks, they are made up of neurons that have learnable weights and biases. Each neuron receive some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores ate the other. And they sill have loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

So what changes? ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

2.1 Architecture overview

As we known, Regular Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings it represents the class scores.

However, Regular Neural Nets don't scale well to full images. For examples, with an images only size of $32 \times 32 \times 3$ (32 wide, 32 high, 3 color channels), a single fully - connected neuron in a first hidden layer of a regular Neural Network would have 32*32*3 = 3072 weights. But with an image of more respectable size, for example: $200 \times 200 \times 3$, would lead to neurons that have 200*200*3 = 120000 weights. Moreover, we would almost certainly want to have several such neurons, so the parameters would add up quickly! Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting.

Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. Moreover, the final output layer would for CIFAR-10[4] have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. Here is a visualization:

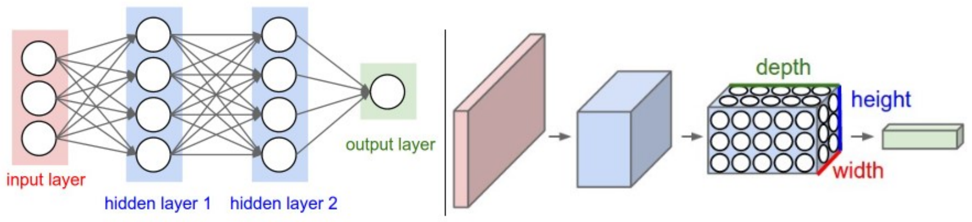


Figure 2.1: **Left:** A regular 3-layer Neural Network. **Right:** A ConvNet arranges neurons in three dimensions (width, height, depth), as visualized in one of the layers.

2.2 Layers used to build ConvNets

As we described above, a simple ConvNet is a sequence of layers, and every layer of a ConvNet transforms one volume of activations to another through a differentiable function. We use three main types of layers to build ConvNet architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). We will stack these layers to form a full ConvNet architecture. We will go into more details below, but a simple ConvNet for CIFAR-10 classification could have the architecture *[INPUT - CONV - RELU - POOL - FC]*. In more detail:

- **INPUT** [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- **CONV** layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- **RELU** layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged ($[32 \times 32 \times 12]$).
- **POOL** layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- **FC** (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

In this way, ConvNets transform the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and other don't. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the ConvNet computes are consistent with the labels in the training set for each image.

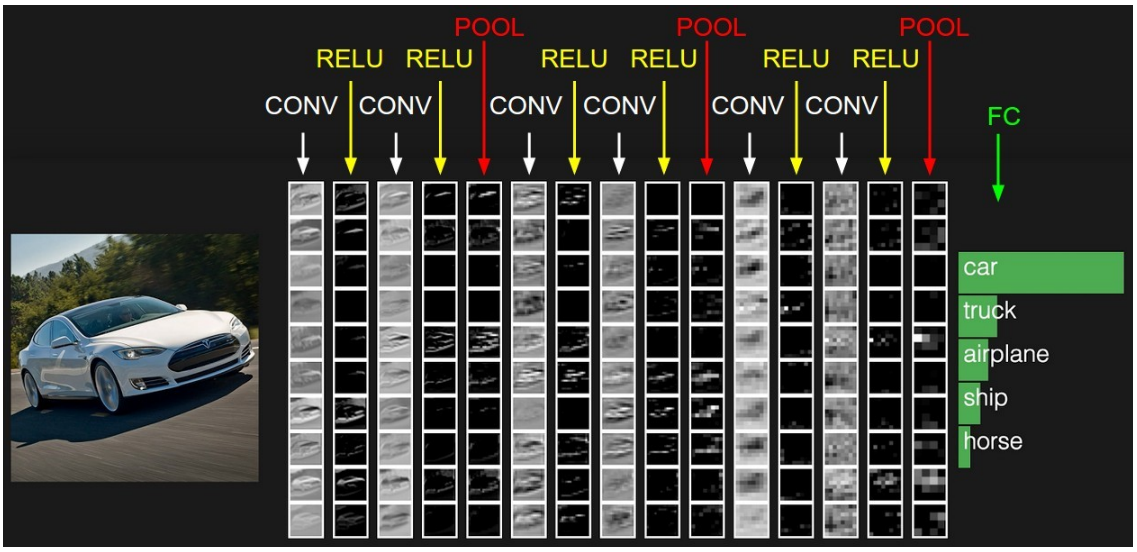


Figure 2.2: Example of Object detection using ConvNets

2.3 Convolutional Layer

The Conv layer is the core building block of a Convolutional Network that does most of the computational heavy lifting.

The CONV layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height), but extends through the full depth of the input volume. For example, a typical filter on a first layer of a ConvNet might have size $5 \times 5 \times 3$ (i.e. 5 pixels width and height, and 3 because images have depth 3, the color channels). During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position. As we slide the filter over the width and height of the input volume we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position. Intuitively, the network will learn filters that activate when they see some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer, or eventually entire honeycomb or wheel-like patterns on higher layers of the network. Now, we will have an entire set of filters in each CONV layer (e.g. 12 filters), and each of them will produce a separate 2-dimensional activation map. We will stack these activation maps along the depth dimension and produce the output volume.

- Accepts a volume of size $W1 \times H1 \times D1$
- Requires four hyperparameters:

- Number of filters **K**.
- Their spatial extent **F**.
- The stride **S**.
- The amount of zero padding **P**.

- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2 = (W1 - F + 2P)/S + 1$
  - $H2 = (H1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D2 = K$
- With parameter sharing, it introduces **F.F.D1** weights per filter, for a total of **(F.F.D1).K** weights and **K** biases.
- In the output volume, the **d**-th depth slice (of size $W2 \times H2$) is the result of performing a valid convolution of the **d**-th filter over the input volume with a stride of **S**, and then offset by **d**-th bias.

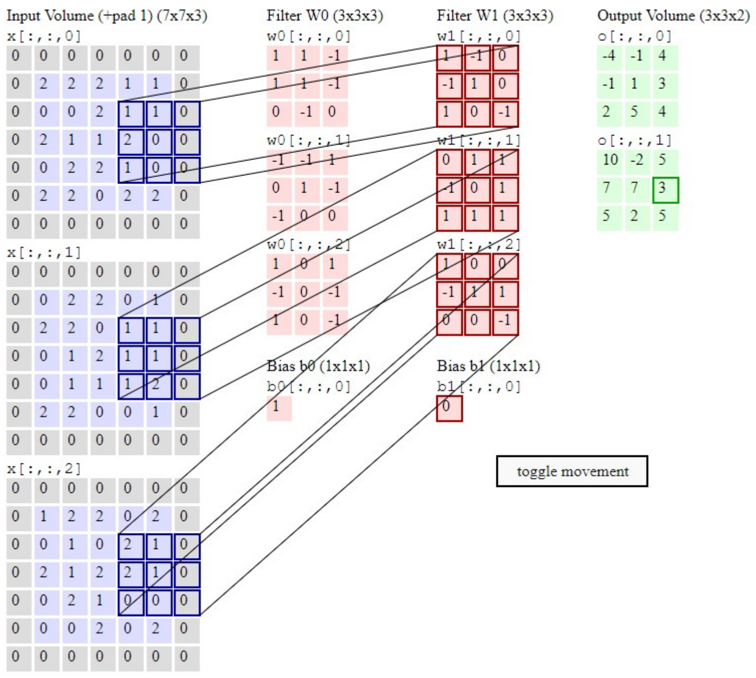


Figure 2.3: Convolution Demo with 2 filter in Conv Layer

2.4 Pooling Layer

It is common to periodically insert a Pooling layer in-between successive Conv layers in a ConvNet architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation. The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 downsamples every depth slice in the input by 2 along both width and height, discarding 75% of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). The depth dimension remains unchanged. More generally, the pooling layer:

- Accepts a volume of size $W1 \times H1 \times D1$
- Requires two hyperparameters:
  - Their spatial extent **F**
  - The stride **S**
- Produces a volume of size $W2 \times H2 \times D2$ where:
  - $W2 = (W1 - F)/S + 1$
  - $H2 = (H1 - F)/S + 1$
  - $D2 = D1$
- Introduces zero parameters since it computes a fixed function of the input.
- For Pooling layers, it is not common to pad the input using *zero-padding*.

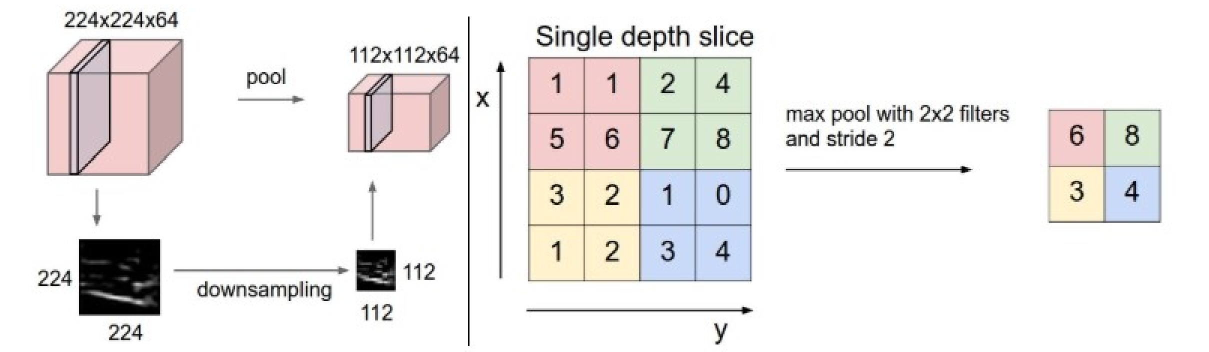


Figure 2.4: Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume.

2.5 Fully-connected layer

Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks. Their activations can hence be computed with a matrix multiplication followed by a bias offset.

2.6 Transfer Learning

In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest. The three major Transfer Learning scenarios look as follows:

- **ConvNet** as fixed feature extractor. Take a ConvNet pretrained on ImageNet, remove the last fully-connected layer (this layer's outputs are the 1000 class scores for a different task like ImageNet), then treat the rest of the ConvNet as a fixed feature extractor for the new dataset. In an AlexNet, this would compute a 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier. We call these features CNN codes. It is important for performance that these codes are ReLUd (i.e. thresholded at zero) if they were also thresholded during the training of the ConvNet on ImageNet (as is usually the case). Once you extract the 4096-D codes for all images, train a linear classifier (e.g. Linear SVM or Softmax classifier) for the new dataset.

- **Fine-tuning the ConvNet**. The second strategy is to not only replace and retrain the classifier on top of the ConvNet on the new dataset, but to also fine-tune the weights of the pretrained network by continuing the backpropagation. It is possible to fine-tune all the layers of the ConvNet, or it's possible to keep some of the earlier layers fixed (due to overfitting concerns) and only fine-tune some higher-level portion of the network. This is motivated by the observation that the earlier features of a ConvNet contain more generic features (e.g. edge detectors or color blob detectors) that should be useful to many tasks, but later layers of the ConvNet becomes progressively more specific to the details of the classes contained in the original dataset. In case of ImageNet for example, which contains many dog breeds, a significant portion of the representational power of the ConvNet may be devoted to features that are specific to differentiating between dog breeds.

- **Pretrained models**. Since modern ConvNets take 2-3 weeks to train across multiple GPUs on ImageNet, it is common to see people release their final ConvNet checkpoints for the benefit of others who can use the networks for fine-tuning. For example, the Caffe library has a Model Zoo where people share their network weights.

2.7 Some Convolutional Networks architectures

There are several architectures in the field of Convolutional Networks that have a name. The most common are:

- **LeNet.** The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's. Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc.[6]

- **AlexNet.** The first work that popularized Convolutional Networks in Computer Vision was the AlexNet, developed by *Alex Krizhevsky, Ilya Sutskever and Geoff Hinton*. The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other (previously it was common to only have a single CONV layer always immediately followed by a POOL layer).[5]

- **GoogLeNet.** The ILSVRC 2014 winner was a Convolutional Network from *Szegedy et al.* from Google. Its main contribution was the development of an Inception Module that dramatically reduced the number of parameters in the network (4M, compared to AlexNet with 60M). Additionally, this paper uses Average Pooling instead of Fully Connected layers at the top of the ConvNet, eliminating a large amount of parameters that do not seem to matter much. There are also several followup versions to the GoogLeNet, most recently Inception-v4.[12]

- **VGGNet.** The runner-up in ILSVRC 2014 was the network from *Karen Simonyan and Andrew Zisserman* that became known as the VGGNet. Its main contribution was in showing that the depth of the network is a critical component for good performance. Their final best network contains 16 CONV/FC layers and, appealingly, features an extremely homogeneous architecture that only performs 3x3 convolutions and 2x2 pooling from the beginning to the end. Their pretrained model is available for plug and play use in Caffe. A downside of the VGGNet is that it is more expensive to evaluate and uses a lot more memory and parameters (140M). Most of these parameters are in the first fully connected layer, and it was since found that these FC layers can be removed with no performance downgrade, significantly reducing the number of necessary parameters. [10]

- **ResNet.** Residual Network developed by *Kaiming He et al.* was the winner of ILSVRC 2015. It features special skip connections and a heavy use of batch normalization. The architecture is also missing fully connected layers at the end of the network. The reader is also referred to Kaiming's presentation (video, slides), and some recent experiments that reproduce these networks in Torch. ResNets are currently by far state of the art Convolutional Neural Network models and are the default choice for using ConvNets in practice (as of May 10, 2016). In particular, also see more recent developments that tweak the original architecture from Kaiming He et al. Identity Mappings in Deep Residual Networks (published March 2016).[3]

CHAPTER 3

Face Verification

3.1 Block Diagram

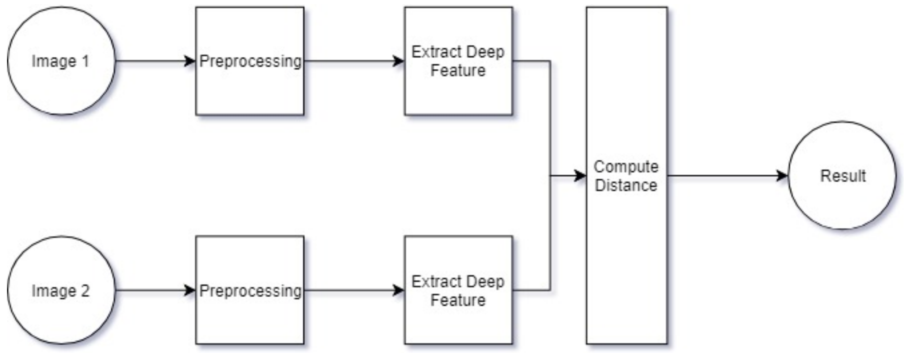


Figure 3.1: Face Verification block diagram

General diagram for face verification as shown in 3.1. Input Image 1 and Image 2 will be get from laptop integrated webcam or camera connected via USB port. Image 1 will be the picture of the person who need to be verified, Image 2 will be the picture of the ID card. In pre-processing block, we will using face detection algorithm to get two face from two image. Then align them into the size of input layer of the CNN. Extracting Deep Feature will compute feature of face image and return a feature vector for each image. By calculating norm distance between two vectors, we can evaluate if two face is from the same person.

3.2 Pre-processing

3.2.1 Face detection and Alignment Algorithm

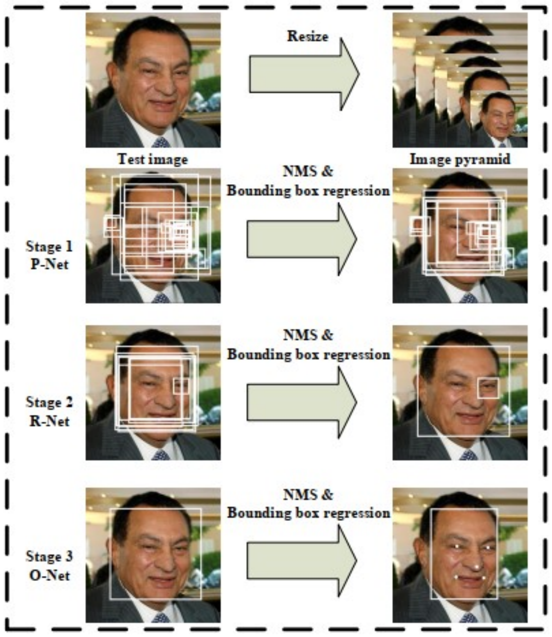
Overall Framework

The overall pipeline of our approach is shown in 3.2a. Given an image, we initially resize it to different scales to build and image pyramid, which is the input of the following three-stage cascaded framework:

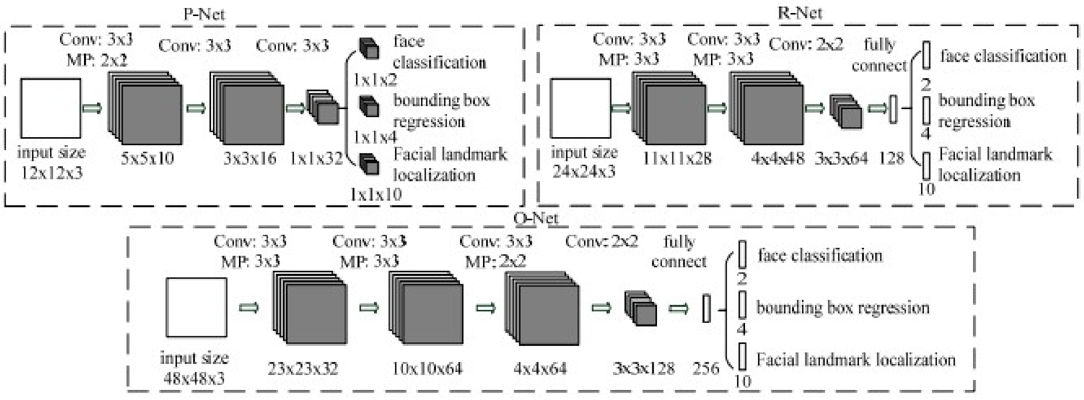
Stage 1: We exploit a fully convolutional network, called Proposal Network (P-Net), to obtain the candidate facial windows and their bounding box regression vectors. Then candidates are calibrated based on the estimated bounding box regression vectors. After that, we employ non-maximum suppression (NMS) to merge highly overlapped candidates.

Stage 2: All candidates are fed to another CNN, called Refine Network (R-Net), which further rejects a large number of false candidates, performs calibration with bounding box regression, and conducts NMS.

Stage 3: This stage is similar to the second stage, but in this stage, we aim to identify face regions with more supervision. In particular, the network will output five facial landmarks' positions.



(a) Overall structure of face detection algorithm



(b) CNN structure of Face Detection

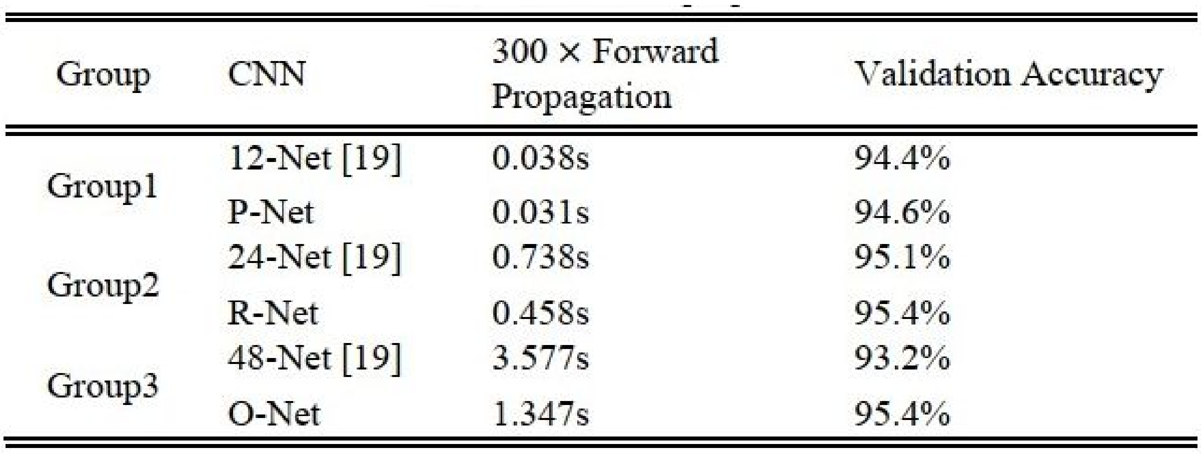
Figure 3.2: Face Detection Algorithm

CNN architectures

In [7], multiple CNNs have been designed for face detection. However, we notice its performance might be limited by the following facts:

1. Some filters in convolution layers lack diversity that may limit their discriminative ability.
2. Compared to other multi-class objection detection and classification tasks, face detection is a challenging binary classification task, so it may need less numbers of filters per layer.

To this end, we reduce the number of filters and change the 5×5 filter to 3×3 filter to reduce the computing while increase the depth to get better performance. With these improvements, compared to the previous architecture in [7], we can get better performance with less runtime (the results in training phase are shown in Table I (see Table 3.1). For fair comparison, we use the same training and validation data in each group). Our CNN architectures are shown in 3.2b. We apply PReLU[14] as nonlinearity activation function after the convolution and fully connection layers (except output layers). [15]



Group	CNN	300 × Forward Propagation	Validation Accuracy
Group1	12-Net [19]	0.038s	94.4%
	P-Net	0.031s	94.6%
Group2	24-Net [19]	0.738s	95.1%
	R-Net	0.458s	95.4%
Group3	48-Net [19]	3.577s	93.2%
	O-Net	1.347s	95.4%

Table 3.1: Comparison of speed and validation accuracy with other CNNs

3.2.2 ID Number detection

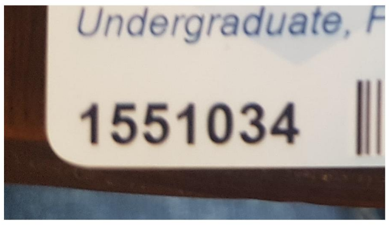
In this part, we will consider the given image with ID number on the bottom left of the image. The following step is showing how to segmenting text from an unstructured scene which will help to improve optical character recognition (OCR).

Step 1: Crop the image keeping the bottom left which contain the ID number.

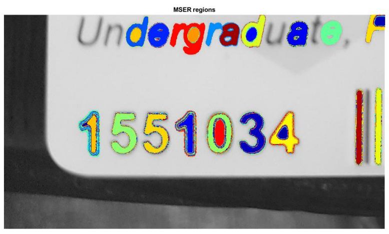
See Figure 3.3a

Step 2: Detect candidate text regions.

Using MSER function to find all the regions within the image and plot these results. Notice that there are many non-text regions detected alongside the text. See Figure 3.3b



(a) Cropped Image with ID number on the bottom left



(b) Detect candidate text region

Figure 3.3: Text detection Process

Step 3: Remove non-text Regions based on basic geometric properties.

There are several geometric properties that are good for discriminating between text and non-text regions, including:

- Aspect ratio
- Eccentricity
- Euler number
- Extent
- Solidity

Use *regionprops* to measure a few of these properties and then remove regions based on their property values.

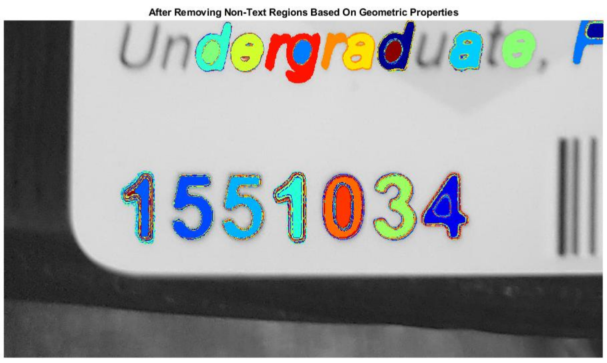

After Removing Non-Text Regions Based On Geometric Properties
Undergraduate, F
1551034

Figure 3.4: After removing non-text regions

Step 4: Merge text regions for final detection result.

At this point, all the detection results are composed of individual text characters. To use these results for recognition tasks, such as OCR, the individual text characters must be merged into words or text lines. This enables recognition of the actual words in an image, which carry more meaningful information than just the individual characters. For example, recognizing the string 'EXIT' vs. the set of individual characters 'X','E','T','I', where the meaning of the word is lost without the correct ordering.

One approach for merging individual text regions into words or text lines is to first find neighboring text regions and then form a bounding box around these regions. To find neighboring regions, expand the bounding boxes computed earlier with *regionprops*. This makes the bounding boxes of neighboring text regions overlap such that text regions that are part of the same word or text line form a chain of overlapping bounding boxes. (See Figure 3.5).

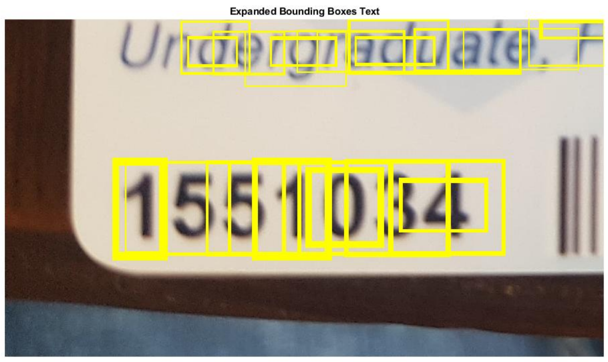


Figure 3.5: Merge text regions for final detection area

Step 5: Recognize detection text using OCR.

After detecting the text regions, use the OCR function to recognize the text within each bounding box. Note that without first finding the text regions, the output of the OCR function would be considerably more noisy. (See 3.6).

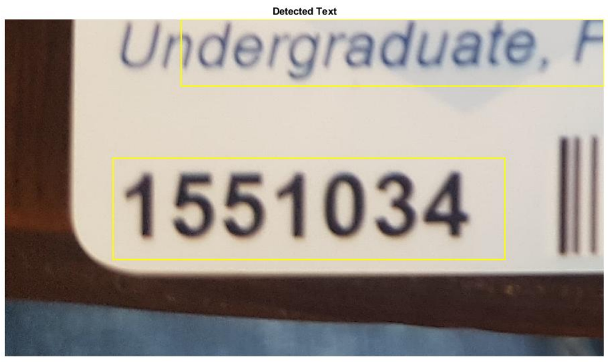


Figure 3.6: Detection result

3.3 Extracting feature and compute distance

3.3.1 Facenet

Facenet is based on learning a Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared L2 distances in the embedding space directly correspond to face similarity: faces of the same person have small distances and faces of distinct people have large distances. [9]

Once this embedding has been produced, then the aforementioned tasks become straight-forward: face verification simply involves thresholding the distance between the two embeddings; recognition becomes a k-NN classification problem; and clustering can be achieved using off-theshelf techniques such as k-means or agglomerative clustering.

Previous face recognition approaches based on deep networks use a classification layer[8] trained over a set of known face identities and then take an intermediate bottleneck layer as a representation used to generalize recognition beyond the set of identities used in training. The downsides of this approach are its indirectness and its inefficiency: one has to hope that the bottleneck representation generalizes well to new faces; and by using a bottleneck layer the representation size per face is usually very large (1000s of dimensions). Some recent work [11] has reduced this dimensionality using PCA, but this is a linear transformation that can be easily learnt in one layer of the network.

In contrast to these approaches, FaceNet directly trains its output to be a compact 128-D embedding using a tripletbased loss function based on LMNN [13]. Our triplets consist of two matching face thumbnails and a non-matching face thumbnail and the loss aims to separate the positive pair from the negative by a distance margin. The thumbnails are tight crops of the face area, no 2D or 3D alignment, other than scale and translation is performed.

Choosing which triplets to use turns out to be very important for achieving good performance and, inspired by curriculum learning, we present a novel online negative exemplar mining strategy which ensures consistently increasing difficulty of triplets as the network trains. To improve clustering accuracy, we also explore hard-positive mining techniques which encourage spherical clusters for the embeddings of a single person. As an illustration of the incredible variability that our method can handle see Figure 3.7. Shown are image pairs from PIE that previously were considered to be very difficult for face verification systems.

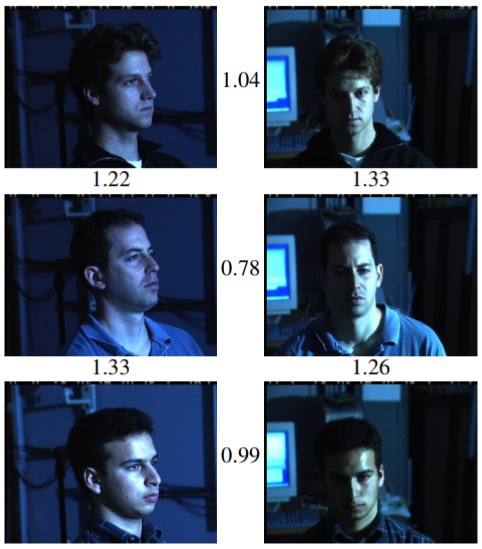


Figure 3.7: Illumination and Pose invariance

Pose and illumination have been a long standing problem in face recognition. This figure shows the output distances of FaceNet between pairs of faces of the same and a different person in different pose and illumination combinations. A distance of 0:0 means the faces are identical, 4:0 corresponds to the opposite spectrum, two different identities. You can see that a threshold of 1.1 would classify every pair correctly.

3.3.2 Compute Distance

With the ideal of Facenet, we use the *pre-trained* inception model with 128-neuron fully connected layer as its last layer. By apply Forward Propagation to faces image, we are able to encode image into 128-d vector.

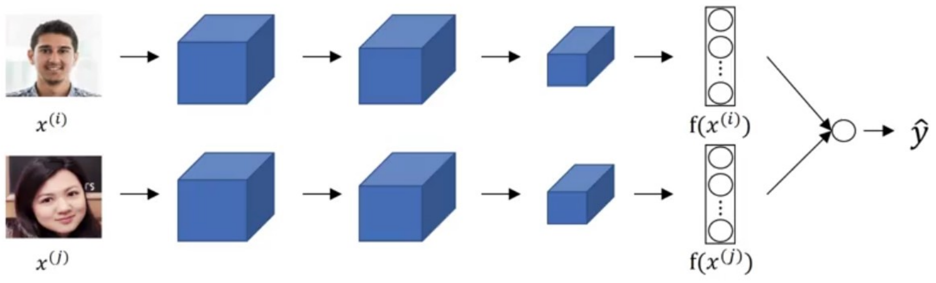


Figure 3.8: Encode Face image into vector

Using L2 norm distance then compare it with threshold (0.4 in this case) we are able to verify if it the same person.

CHAPTER 4

Result and Conclusion

4.1 Result

By using GUI integrated with MATLAB (Figure 4.1), we are able to make an application which can:

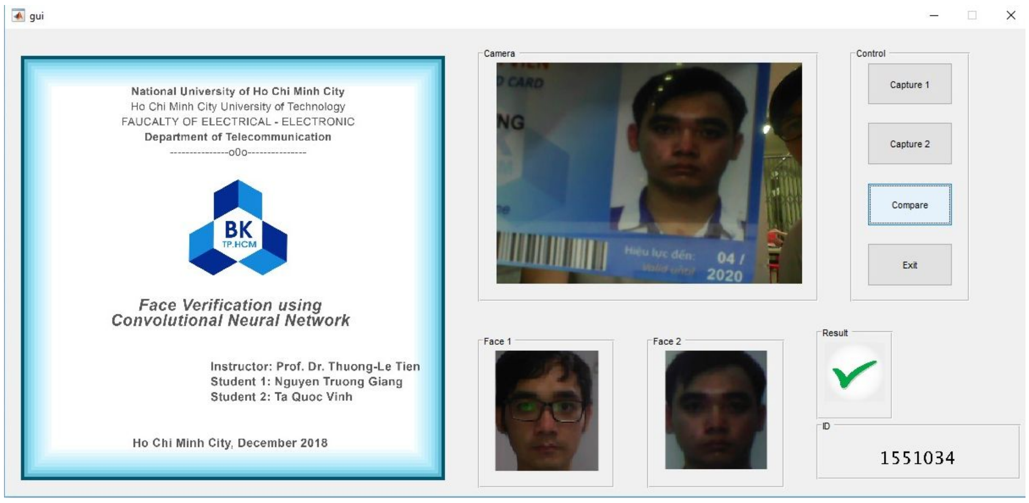
1. Capturing and detect student's face.
2. Capturing their ID card.
3. Detecting ID number on the card.
4. Detecting face on ID card.
5. Matching whether two detected face is from one person.

And to evaluate our project, by collecting picture of lost ID card and matching with their daily life's picture, we are able to set up the following table (Table 4.1)
From the result above we can have some general conclusion that:

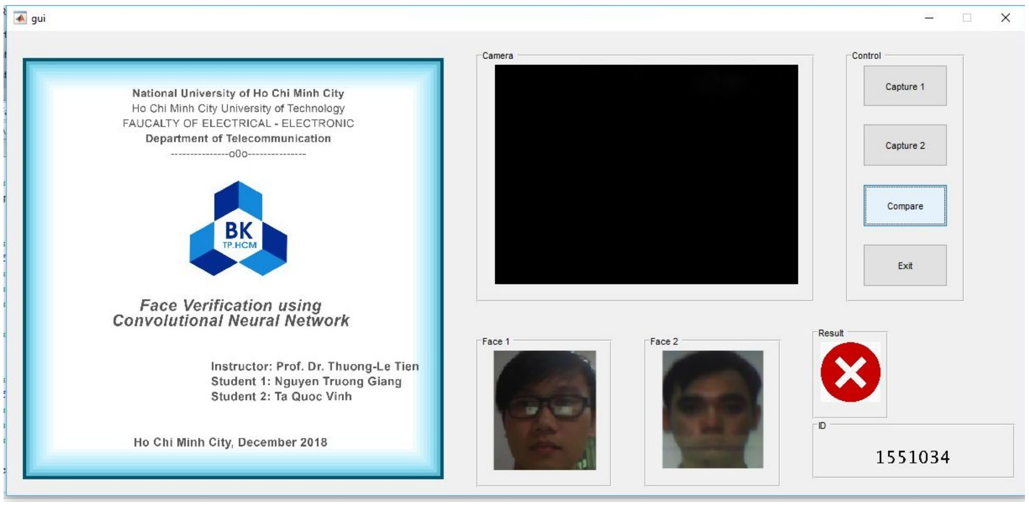
- Our method work correctly in most of the case.
- The failure of ID number detection is because of the card position need to be fixed.
- Picture need to be taken with enough light.
- Picture must not be edited before the process.

Number of Sampling	**Student ID accuracy**	**Face Verification accuracy**
10	90%	100%
20	95%	100%
30	86.67%	96.67%
40	92.50%	97.30%

Table 4.1: Evaluation Table


gui
National University of Ho Chi Minh City
Ho Chi Minh City University of Technology
FAUCALTY OF ELECTRICAL - ELECTRONIC
Department of Telecommunication
BK TP.HCM
Face Verification using Convolutional Neural Network
Instructor: Prof. Dr. Thuong-Le Tien
Student 1: Nguyen Truong Giang
Student 2: Ta Quoc Vinh
Ho Chi Minh City, December 2018
Camera
Control
Capture 1
Capture 2
Compare
Exit
Face 1
Face 2
Result
ID
1551034

(a) Correct Case


gui
National University of Ho Chi Minh City
Ho Chi Minh City University of Technology
FAUCALTY OF ELECTRICAL - ELECTRONIC
Department of Telecommunication
BK TP.HCM
Face Verification using Convolutional Neural Network
Instructor: Prof. Dr. Thuong-Le Tien
Student 1: Nguyen Truong Giang
Student 2: Ta Quoc Vinh
Ho Chi Minh City, December 2018
Camera
Control
Capture 1
Capture 2
Compare
Exit
Face 1
Face 2
Result
ID
1551034

(b) Incorrect Case

Figure 4.1: Matlab GUI demo

4.2 Conclusion and Future Development

4.2.1 Conclusion

As constructing the identification system for student ID card, this will help the university specially Bach Khoa university develop the attendance method or work counting method by verifying students or officer, lecturers faces and their ID card. In the reality, there are machine for counting and attendance by fingerprint, by co-operation these machine together, the civil management become more exact and efficient. Based on the pre-trained model that we mentioned above, we can create the algorithm that reduce the normalization between two processed images input from ID card captured and real-time face captured. The similarity of two results that processed by a net frame work is approximately 96%. The interface operation in MATLAB, we using GUI method to make the interface more observable and usable includes of screen for capturing picture from real time face and ID card beside the buttons connected to the cameras that are using to take t snapshot input images.

This algorithm is not relevant on the dataset with the high resolution image.

Also, in the low-light condition the processing image of the net is reduce the accuracy. Due to the low qualification of the image, the vectorization is low and effect on the accuracy of the process. Moreover, as used the pre-trained model, so we can control the processing of the net and the time processing.

4.2.2 Future Development

For future development, firstly we will change the program into Python to combine our system with hardware using Rasberry Py to complete the system. Following, training model network frame is crucial for this research to reduce the existing problems. The algorithm that leads to the accuracy is also an importance, therefore we will try to use the Tripless-loss algorithm method to reduce the normalization of the distance between two input images, also Tripless-loss helps us to control the threshold number easily the make the evaluation assumption more accuracy. By applying embedded systems, we are going to make our research become more complete to be improved in the new technology for industries and enterprises in the future.

References

[1] Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2009.

[2] *Artificial intelligence*. Wikipedia®. 2018. URL: `https://en.wikipedia.org/wiki/Artificial_intelligence` (visited on 11/27/2018).

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[4] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: (). URL: `http://www.cs.toronto.edu/~kriz/cifar.html`.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems*. P. 2012.

[6] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: (1998), pp. 2278–2324.

[7] H. Li, Z. Lin, J. Brandt X. Shen, and G. Hua. "A convolutional neural network cascade for face detection". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA* (2015), pp. 5325–5334.

[8] O. M. Parkhi, A. Vedaldi, and A. Zisserman. "Deep Face Recognition". In: *British Machine Vision Conference*. 2015.

[9] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering." In: *CoRR* abs/1503.03832 (2015). URL: `http://dblp.uni-trier.de/db/journals/corr/corr1503.html#SchroffKP15`.

[10] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2014). arXiv: `1409.1556`. URL: `http://arxiv.org/abs/1409.1556`.

[11] Yi Sun, Xiaogang Wang, and Xiaoou Tang. "Deeply learned face representations are sparse, selective, and robust". In: *CoRR* abs/1412.1265 (2014). arXiv: `1412.1265`. URL: `http://arxiv.org/abs/1412.1265`.

[12] Christian Szegedy et al. "Going Deeper with Convolutions". In: *CoRR* abs/1409.4842 (2014). arXiv: `1409.4842`. URL: `http://arxiv.org/abs/1409.4842`.

[13] Kilian Q. Weinberger and Lawrence K. Saul. "Distance Metric Learning for Large Margin Nearest Neighbor Classification". In: *J. Mach. Learn. Res.* 10 (June 2009), pp. 207–244. ISSN: 1532-4435. URL: `http://dl.acm.org/citation.cfm?id=1577069.1577078`.

[14] Cha Zhang and Zhengyou Zhang. "Improving multiview face detection with multi-task deep convolutional neural networks". In: *IEEE Winter Conference on Applications of Computer Vision* (2014), pp. 1036–1041.

[15] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks". In: *IEEE Signal Processing Letters* 23.10 (2016), pp. 1499–1503.