Convolutional Neural Network (CNN) for Face Recognition

Yibo Cheng
Department of Computer Science
(The Jonsson School)
The University of Texas at Dallas
Richardson, TX
yxc190039@utdallas.edu
Chaoran Li
Department of Computer Science
(The Jonsson School)
The University of Texas at Dallas
Richardson, TX
cxl190012@utdallas.edu

***Abstract*— Science and technology are developing in the fast pace. Algorithm and theoretical foundation, technology starts to lead in people's daily life and transforms the way it is. One of the common applications is using convolutional neural network to classify images or objects based upon their features. Convolutional Neural Network (CNN) has excellent ability to develop an internal representation of a two-dimensional image, making it the preferred network for image classification and recognition. In this paper, a classical deep learning algorithm, convolutional neutral network , was implemented to realize image recognition. The team built up a convolutional neutral network from the scratch. Stages include intaking dataset, preprocessing images, convolutional layer, activation layer, pooling layer, fully connected layer and forward and backward propagation. Meanwhile, effects brought by parameters, like convolutional matrix, batch size, were studied and compared in terms of network efficiency, accuracy and generality. The testing accuracy of CNN is around 90% ~ 100%, which is acceptable as there are few issues the team does not intend to tackle down at this time. Open issues include overfitting and underfitting, activation function, image resolution loss, etc. Potential improvement on recognition accuracy could be realized if these problems were addressed and will be left for the future tasks.**

Keywords—CNN, Image Classification, Tuning, Convolutional Layer

I. INTRODUCTION & BACKGROUND

(Yibo Cheng: 50%, Chaoran Li: 50%)

Science and technology are developing in the fast pace. Especially equipped with advanced computing hardware, algorithm and theoretical foundation, technology starts to lead in people's daily life and transforms the way it is. Science and technology is served as optimal solution to tackle down human problems and challenges. One of the common applications is using convolutional neural network to recognize images or objects based upon their features. In this project, the team wants to build a convolution neural network based face recognition which the network will first learn from input image and then be able to distinguish if test person is the person it sees before. This application has many uses in the daily life, places like home security, device unlocking (with more advanced neural network), and even medical fields.

Since 1943 when Warren McCulloch, a neurophysiologist, and a young mathematician, Walter Pitts, wrote a paper on how neutron might work, artificial neutral network has been studied all over the world. By now, there are thousands of types of specific neural networks proposed by researchers as modifications or tweaks to existing models and even more are still emerging. These models can be categorized into three classes in general, Multilayer Perceptron (MLP), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) . Each class has its own design purpose and focus. CNN's ability to develop an internal representation of a two-dimensional image makes it the preferred network for image classification and recognition.

The convolutional neural network (CNN) was first proposed in 1960s. Hubel and Wiesel proposed the concept of "receptive field" which observed that neurons were sensitive to moving edge on visual cortex cells of cats. Later in the 1980s, based on concept of "receptive field", Fukushima and Miyake proposed "neocognitron" which is regarded as the first implementations of CNNs. However, due to lack of proper learning algorithm, CNN was not the main focus in the network. After that, researchers started to use multilayer perceptron to learn features and incorporated backpropagation (BP) algorithm. However, since traditional BP neutral network would have series of problems requiring detailed study, the research on deep neutral network model was stopped. Until Hinton et al found that the artificial neural network with multiple hidden layers addresses those old issues and has great performance in feature learning, deep learning starts to re-gain attention and more and more sophisticated and accurate models were developed and used in daily practice, especially in the fields of OCR, autonomous drive, image recognition and analysis, social media, etc.

In this paper, a classical deep learning algorithm, convolutional neutral network (CNN), was implemented to realize image recognition. The team built up a convolutional neutral network from the scratch. Stages include obtaining input dataset, preprocessing images, convolutional layer, activation layer, pooling layer, fully connected layer and forward and backward propagation. Meanwhile, effects brought by parameters, like convolutional matrix, batch size, were studied and compared in terms of network efficiency, accuracy and

generality. Meanwhile, the team aims for designing a convolutional neural network that has high accuracy when it detects faces and judge, so parameter tuning used in the network is crucial and detrimental to the results.

II. Technique & Algorithm

(Yibo Cheng: 50%, Chaoran Li: 50%)

A brief introduction of convolutional neural network's history and this project's background is laid out in the previous section. In this section, it is mainly focused on the introducing procedures of the project and detail techniques that are utilized to achieve the goal: let the computer know you so that it can disguish you from other unseen faces.

First of all, since this project is purely made by the team, input image dataset becomes a priority to collect. In order to teach the neural network to learn from host face, extensive amount of host face images should be provided. To achieve so, the team comes up with two techniques to collect host face images. One way to do so is to use provided dlib library to capture host faces through front camera on the laptop. Another way is to use OpenCV to collect through front camera again. Below is the code snippet on how face images are collected using dlib and OpenCV.

```
detector = dlib.get_frontal_face_detector()
camera = cv2.VideoCapture(0)

index = 1
while True:
    if (index <= 10000):
        print('Being processed picture %s' % index)
        success, img = camera.read()
        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        dets = detector(gray_img, 1)

        key = cv2.waitKey(30) & 0xff
        if key == 27:
            break
    else:
        print('Finished!')
        break
```

Fig.1: Host Face Images Capture Using dlib

```
haar = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
camera = cv2.VideoCapture(0)

n = 1
while 1:
    if (n <= 10000):
        print('It`s processing %s image.' % n)
        success, img = camera.read()

        gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = haar.detectMultiScale(gray_img, 1.3, 5)
        for f_x, f_y, f_w, f_h in faces:
            face = img[f_y:f_y+f_h, f_x:f_x+f_w]
            face = cv2.resize(face, (64,64))
            face = relight(face, random.uniform(0.5, 1.5), random.randint(-50, 50))
            cv2.imshow('img', face)
            cv2.imwrite(out_dir+'/'+str(n)+'.jpg', face)
            n+=1
        key = cv2.waitKey(30) & 0xff
        if key == 27:
            break
    else:
        break
```

Fig.2: Host Face Images Capture Using OpenCV

Each one has its own advantages. However, in order to collect enough images (~10000 images), time is the first thing the team consider. It turns out that dlib will take around 20 minuates to finish the collection where it will take approximately 1 hour for OpenCV to do so. The team decides to use dlib in the project to finish the dataset collection task. Another thing the team considers to improve during the dataset collection is the quality of face images. To improve this, the team changes each collected host face images' brightness and contrastness. Meanwhile, each image is resized to 64 x 64 so that later calculation workload can be reduced. Below is the code snippet on how resize and brightness and contrastness change works.

```
def relight(img, light=1, bias=0):
    w = img.shape[1]
    h = img.shape[0]
    #image = []
    for i in range(0,w):
        for j in range(0,h):
            for c in range(3):
                tmp = int(img[j,i,c]*light + bias)
                if tmp > 255:
                    tmp = 255
                elif tmp < 0:
                    tmp = 0
                img[j,i,c] = tmp
    return img
```

Fig.3: Face Images Re-light Function

Up to now, a total of 10000 host face images are collected and preprocessed in use of input dataset. Next step is to find proper train and test face image dataset. Here, the team uses provided dataset from the University of Massachusetts Amherst. Same image preprocessing (resize, brightness and contrastness) is applied on each test and train images as well. Example testing images are provided below.

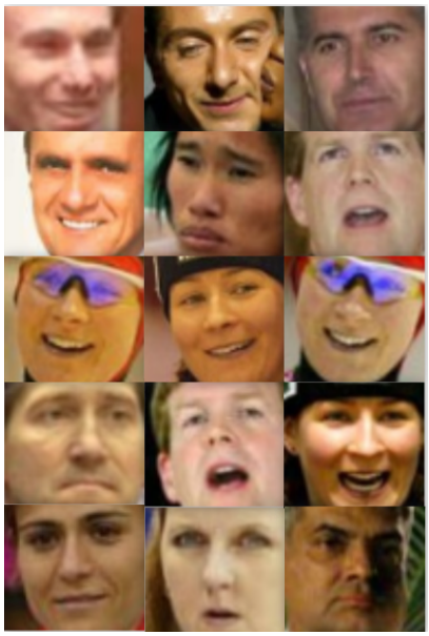

Fig.4: Example Testing Images

Before introducing details on network layers, a brief introduction on convolutional neutral network layers is provided here.

A. Convolutional Neutral Network (CNN)

Compared to other image recognition algorithms, CNNs do not require extra work on preprocessing images and this means that they can learn the filters that have to be hand-made in other algorithms. Also, CNNs have advantage in dimensional reduction without losing learning features. This could tremendously save computing time. Layers including convolution, pooling, activation and fully connection constitute the network and the sequence of each operation could be shuffled according to the need.

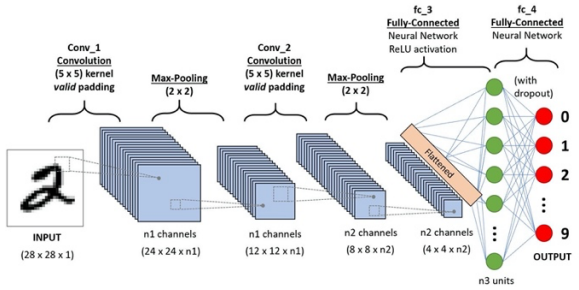

Fig. 5: A CNN sequence to classify handwritten digits

B. Convolutional Layer

First layer of CNN normally starts with a convolutional layer. A convolutional layer contains a set of filters whose parameters including height, weight, number of filters are determined through inputs. In general, the height and weight of the filters are smaller than those of the input volume (in research, input is a 2D matrix image with 1 channel, gray image). Each filter is convolved with the input volume to compute an activation map. In specific, the filter is slid across the width and height of input with appointed stride and the dot products between the input matrix and filter are computed at every spatial position. After number of filter's iterations, the output of convolutional layer is obtained by stacking the activation maps of all filters along the depth dimension. For example, in the research, given the situation that input volume is a uniform a*a matrix, convolutional filter is chosen to be b*b matrix and each stride is assigned as c, the final output's shape after convolutional layer will be (a + c – b) * (a + c – b).

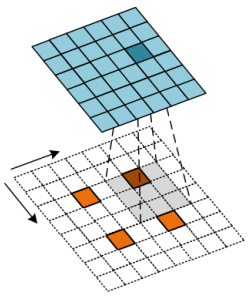

Fig. 6: Convolution process of transposed convolution layer

C. Pooling Layer

The purpose of pooling layer in the network is to progressively reduce the spatial size of representation to reduce the number of parameters and computation in the network. Normally, pooling layer will exist in-between convolutional layers and it takes a series input parameter including input volume, number of filters, height, width, stride and padding. Two main techniques in pooling are popular and widely used, average pooling and maximum pooling. In the research, maximum pooling is implemented for down sampling input volume. Max Pooling ensures a lower resolution version of an input signal is created and still contains large portion of key features, without fine detail that might not be as useful to the task.

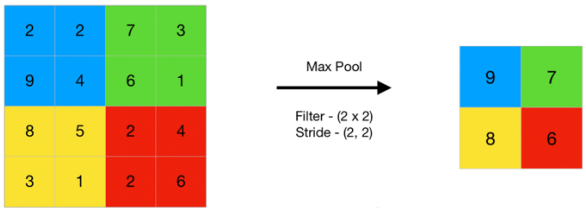

Fig. 7: Max Pooling Layer

D. Activation Layer

The function of activation layer in the network is to transform linear output from either convolutional layer or pooling layer to non-linear output so that the network is able to learn complex patterns in the data. Many activation functions are available and, in the research, rectified linear unit (ReLU) is implemented, which simply computes the function: f(x) = max (0, x).

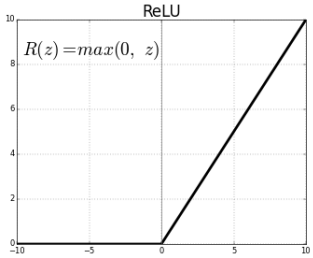

Fig. 8: ReLU Activation Function

E. Fully Connected Layer

The objective of fully connected layer in the network is used to classify the image into a target label. At this layer, it will receive output from either convolutional layer or pooling layer. The input will then be further flattened into a single vector of values, each representing a probability that a certain feature belongs to a label.

Identify applicable funding agency here. If none, delete this text box.

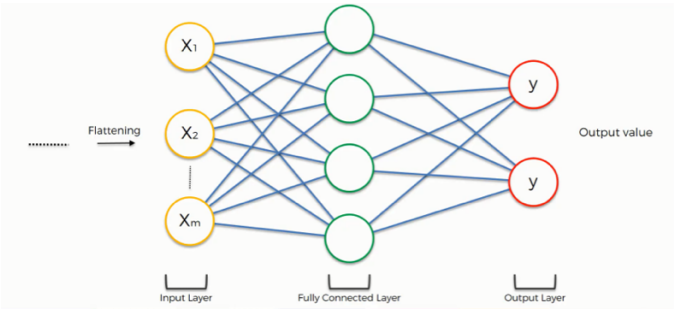

Fig. 9: Fully Connected Layer.[20]

III. PROJECT DETIALS

(Yibo Cheng: 50%, Chaoran Li: 50%)

The main purpose of project is to implement Convolutional Neural Network on teaching it to recognize the host by distinguish it from other unseen host images. In this paper, the team uses Python, Numpy, tensorflow, dlib and OpenCV to achieve input dataset collection, image preprocessing and Convolution Neural Network setup. 10000 amount of single host face images and other unseen face images provided by UMass (The University of Massachusetts Amherst) are splited into training and testing set X, Y randomly in a ratio of 0.7:0.3.

A. The Architecture of Convolutional Neural Networks

Our Convolutional Neural Networks mainly contains Convolution Layer, Pooling Layer, Dropout Layer, ReLu Layer, Flatten Layer, FullyConnected Layer, and Softmax Layer. Among them, Convolution Layer, Pooling Layer, Dropout Layer can be combined as a small layer group. And this group can be duplicated many times in the architecture of CNN.

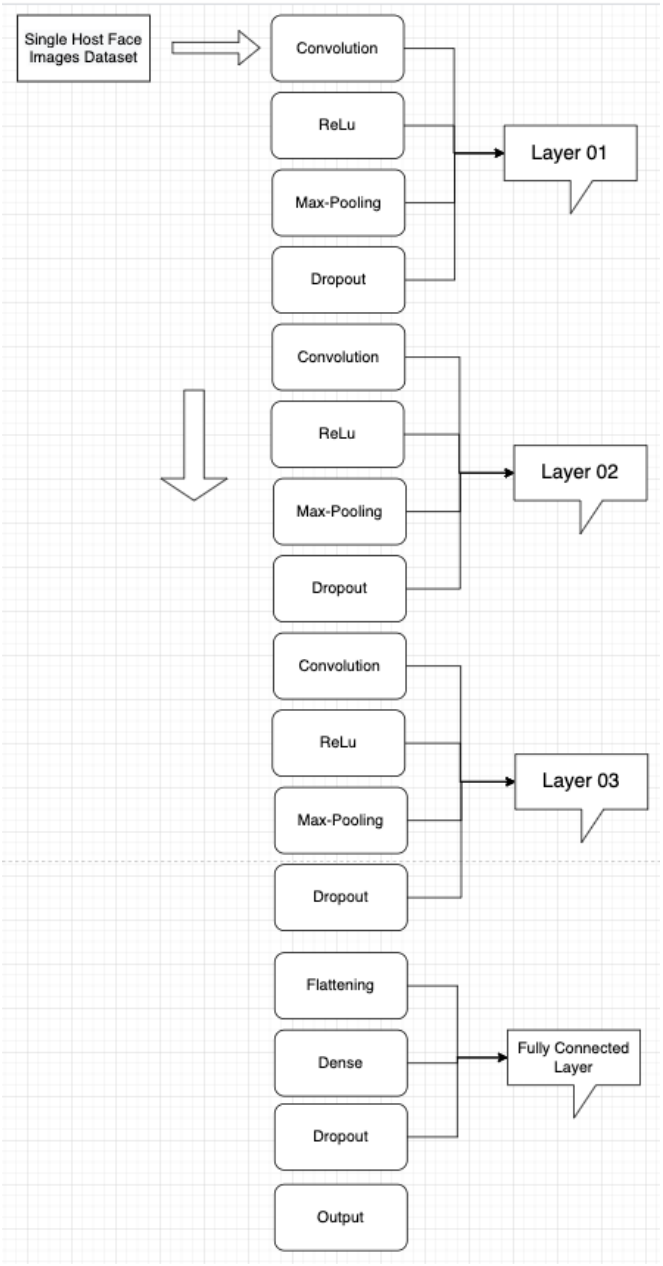

Fig. 9: The architecture of designed Convolutional Neural Networks

Figure 9 shows a representative order of layers in the Convolutional Neural Networks the team built up. Each big layer consists of one convolution layer, max-pooling layer and dropout layer. Convolution layer's kernel size, input channel size and output channel vary as needed. All Max Pooling Layers have kernel size [1, 2, 2, 1], stride size [1, 2, 2, 1] and padding to same as input size. The second Convolution layer has 16 filters with filter length 5. Detail on each layer's parameter numbers is presented below.

Table 1: Convolutional Neural Network Parameters

	Kernel Size	In-channel	Out-channel	Bias	Stride	Padding
Conv	[3,3]	3	32	/	[1,1,1,1]	Same
ReLu	/	/	/	32		/
MaxPool	/	/	/	/	[1,2,2,1]	Same
Dropout	/	/	/	/	/	/
Conv	[3,3]	32	64	/	[1,1,1,1]	Same
ReLu	/	/	/	64	/	/
MaxPool	/	/	/	/	[1,2,2,1]	Same
Dropout	/	/	/	/	/	/
Conv	[3,3]	64	64	/	[1,1,1,1]	Same
ReLu	/	/	/	64	/	/
MaxPool	/	/	/	/	[1,2,2,1]	Same
Dropout	/	/	/	/	/	/
Flatten	/	/	/	/	/	/
Dense	/	/	/	/	/	/
Dropout	/	/	/	/	/	/
Output	/	/	/	/	/	/

B. Purpose of experiment.

The most important purpose of this project is to implement Convolutional Neural Networks successfully to distinguish from other hosts from original host through face images. By writing the codes of different layers and forward and backward functions in all the layers, we can get comprehensive understanding of how CNN works and how different layers influence the result. Then the team gets much practice with training them by using different input data, batch size, different layer setup or filter number and stride etc. In this process, we can gain a better understanding of how to make CNN work better and most importantly how to fine-tune parameters to reach the optimal state.

In this paper, totally, 7000 images of single host face images have been used as training data. 3000 face images of single host and around 2000 images of other unseen other host images have been used as testing data. Result will show true or false over provided face image. True means the test image is original host's face image and false if otherwise. All the images were resized and enhanced and changed to gray before training or testing.

IV. Result and Analysis

(Yibo Cheng: 50%, Chaoran Li: 50%)

In order to obtain a fine and accurate convolutional neural network model, the team puts effort on fine-tuning parameters within the model. Parameters the team focuses on include batch size, learning rate and number of big layers, where each layer typically has convolution, ReLu, max-pooling.

A. Batch Size

The batch size defines the number of samples that will be propagated through the network. For instance, if 1000 training samples are provided and batch size is set to 100, it means the algorithm will take the first 100 smaples from the training dataset and train the network. Next, it will take the second 100 samples to train. Same procedure will be carried out until all samples are learnt. It is apparent that using a batch size that is smaller than number of all samples could require less memory consuming and speed up training process. The team exploits different batch size to evaluate the effects it brings to the final accuracy.

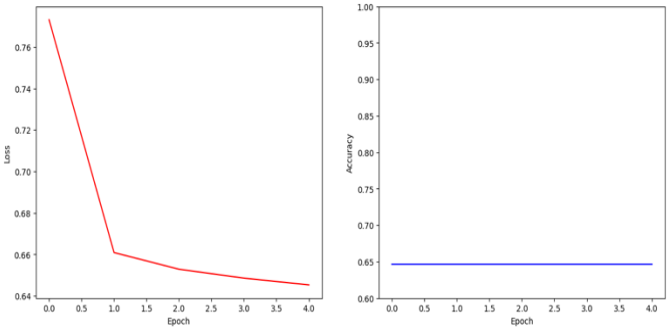

Fig.10: Batch size of 64 samples

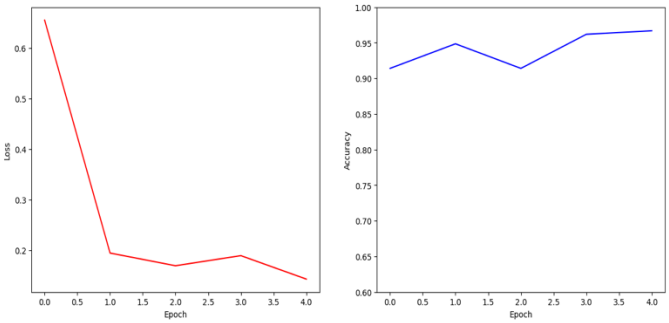

Fig.11: Batch size of 96 samples

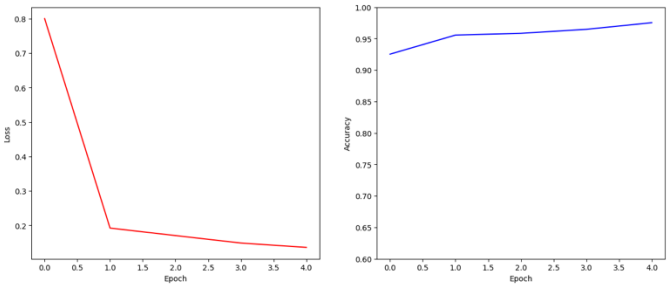

Fig.12: Batch size of 128 samples

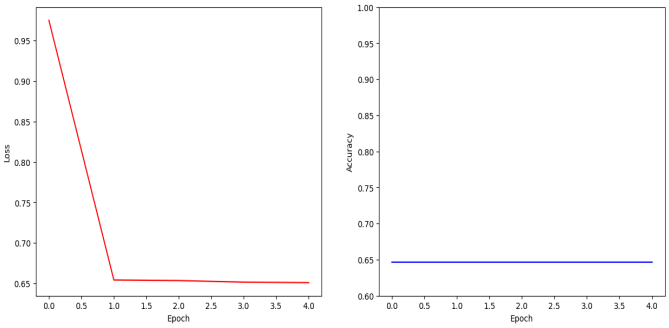

Fig.13: Batch size of 192 samples

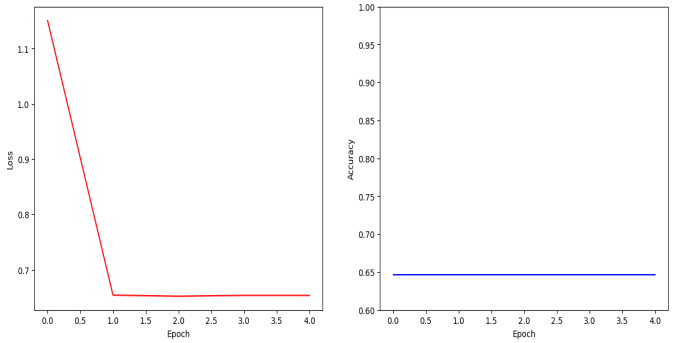

Fig.14: Batch size of 256 samples

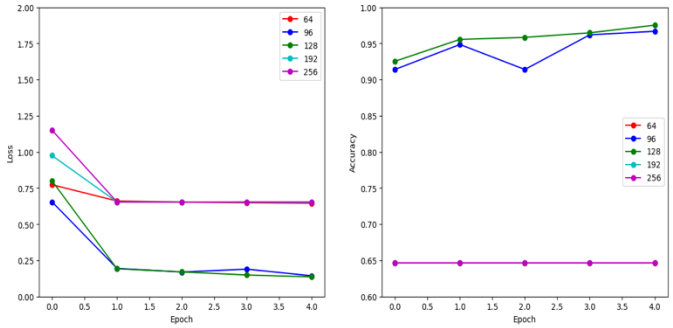

Fig.15: Overall Comparison of Different Batch Size

The team tries different batch sizes to train CNN and obtains the relation between batch size and testing accuracy. Five trials (batch size = 64, 96, 128, 192, 256) are conducted and from figure 15, it can tell that batch size of 128 outperforms other trails and provides with steady and smooth testing accuracy. Batch sizes of 64, 192 and 256 samples barely affect testing accuracy and even worse, they lead to poor generalization and further reduce accuracy overall. Even though there are only five experiments conducted on batch size effects, the team already has a good understanding on its potential and realizes higher batch sizes can somehow lead to lower asymptotic test accuracy. So, the final model uses batch size of 128 as optimal one.

B. Learning Rate

Learning rate is a hyperprameter that controls how much to change the model in response to the estimated error each time the model weights are updated. The learning rate may be the most important hyperparameter when configuring the neural work, because if the learning rate is too small, it may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process. The team fine-tunes learning rate through several experiment and exploit its effect on loss and testing accuracy.

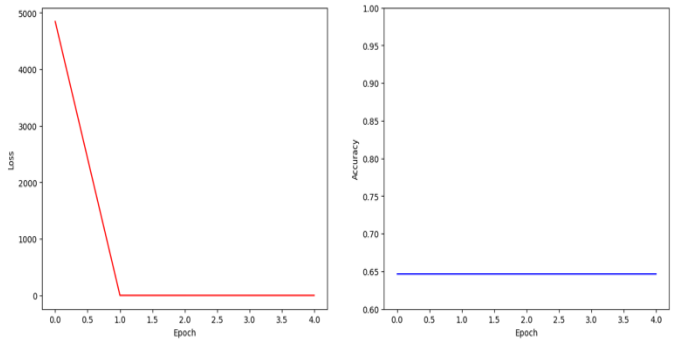

Fig.16: learning rate = 0.1

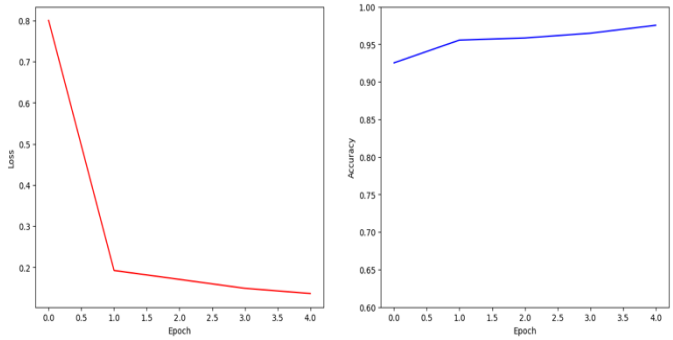

Fig.17: learning rate = 0.01

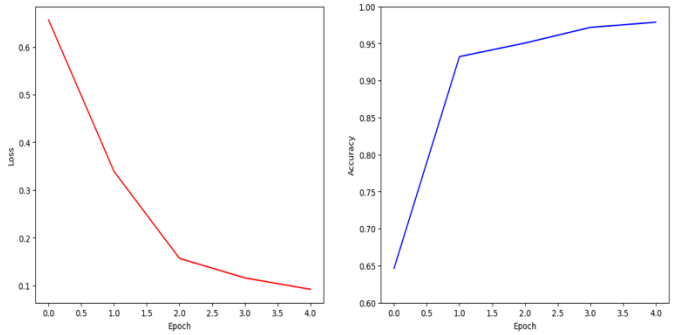

Fig.18: learning rate = 0.001

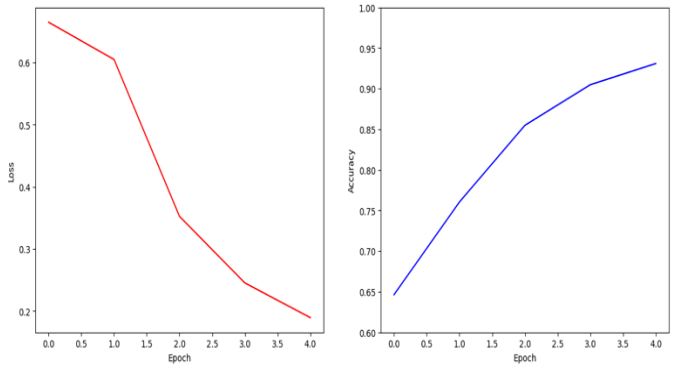

Fig.19: learning rate = 0.0001

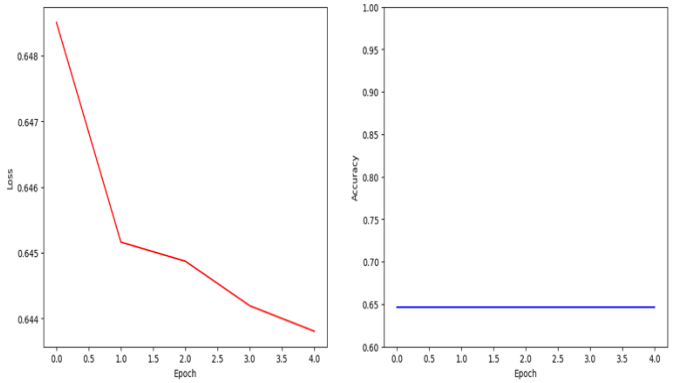

Fig.20: learning rate = 0.00001

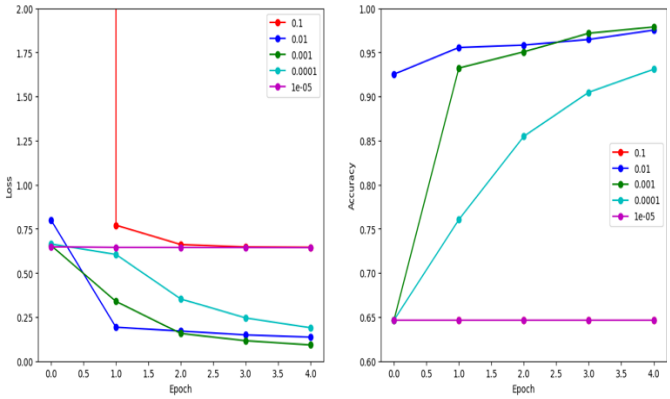

Fig.21: Overall Comparison of Different Learning Rate

The team experiments learning rate of 0.1, 0.01, 0.001, 0.0001 and 0.00001and reflects its convergence and effects on testing accuracy. Figure 21 gives a overall comparison and it can tell that when learning rate is set to either 0.1 or 0.00001, the convergence is really bad, so does its testing accuracy. Learning rates of 0.01, 0.001 and 0.0001 give good gradually decreasing loss at each epoch and learning rate of 0.001 is the best of the three because of its smothiness. This reflects in the testing accuracy as well where learning rate of 0.001 eventually outperforms others in terms of testing accuracy. Base upon this observation, the team chooses learning rate of 0.001 as the optimal one and proceed the experiment with this value.

C. Neural Network Hidden Layers

Speaking of neural network hidden layers, neural network capacity is something that should be taken account of. The capacity of a deep learning neural network model controls the scope of the types of mapping functions that is able to learn. A model with too little capacity cannot learn the training dataset meaning it will underfit, whereas a model with too much capacity may memorize the training dataset, meaning it will overfit or may get stuck or lost during the optimization process. The capacity of a neural network model is defined by the number of nodes and the number of layers. In this part, the team mainly focuses on fine-tuning number of hidden layers in the CNN model and learn how hidden layers will affect the final testing accuracy by adding more layers. Below are three experiments the team tries to learn. One thing to mention here is that a neural layer here consists of one convolution layer, ReLu layer, max-pooling layer and dropout layer.

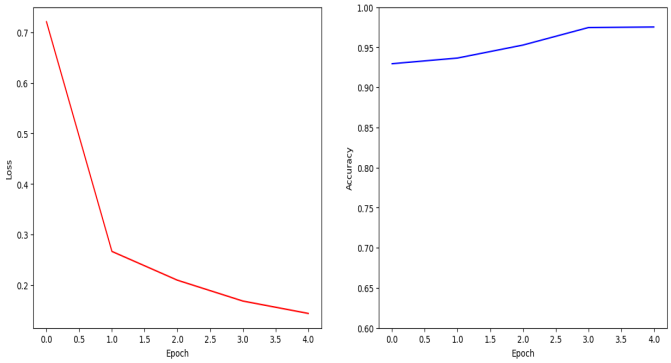

Fig.21: One Neural Layer

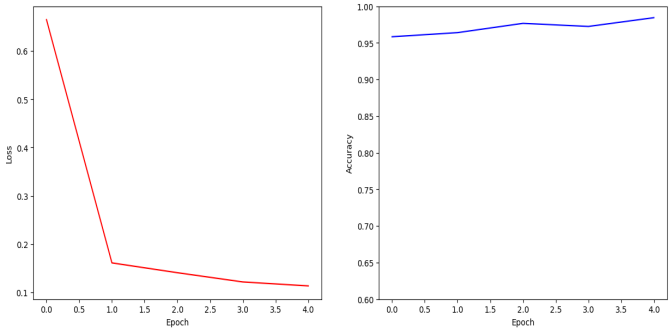

Fig.22: Two Neural Layers

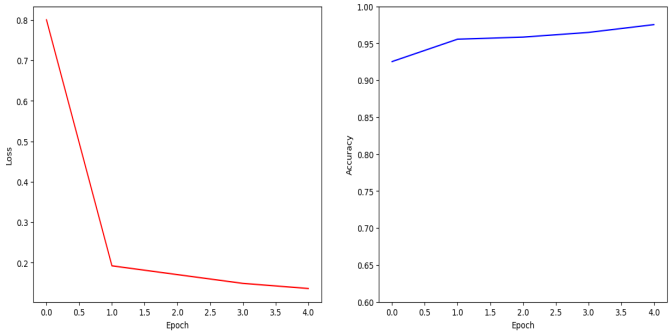

Fig.23: Three Neural Layers

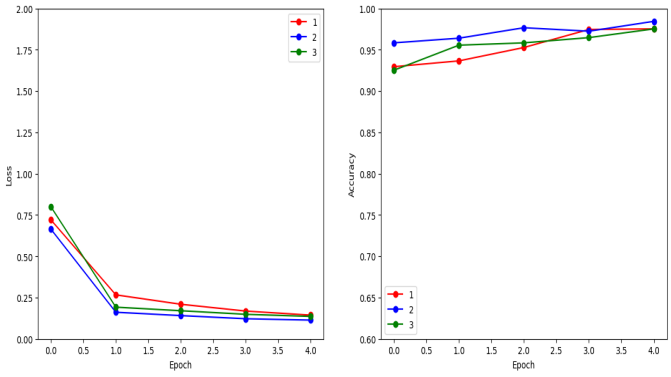

Fig.24: Overall Comparison of Different Number of Neural Layers

The team tries three models with one, two and three neural layers. The result shown in the Figure 24 suggests that within limited epochs, model with two neural layers has better performance in terms of loss and testing accuracy. However, this does not imply model with two neural layers works the best of the three since performance really depends on the complexity of the problem you are trying to solve. In this case, the team goes with model with three neural layers where each neural layer has one convolution layer, one ReLu layer, one max-pooling Layer and one dropout layer.

D. Training Data Size and Diversity

Another factor the team looks into to improve the result is to include more training face images and its diversity as well. Instead of only having single host face images to train, several other friends' faces are included here to make a comparison and see how the result accuracy could improve from it. Below is some details.

Table 02: Confidence of Predicting testing set to be my faces

Training	My faces + UMass faces		My faces + Friend faces + UMass faces	
Testing (1000 images)	my face	friend faces	my faces	friend faces
1	98.9%	65.0%	95.2%	0.5%
2	98.2%	64.5%	95.5%	0.6%
3	98.6%	64.0%	94.9%	0.2%

Finally, the fine-tuned CNN model the team uses has three neural layers with 128 batch size and 0.0001 learning rate. In order to demonstrate the model result, the team comes up with two ways to interact with users. One way is to allow user to upload testing images and get result from there. The other way is to allow front camera to capture user's face image directly and the model will decide if it knows or sees user before. The first one has 100% accuracy compared with the latter one where the accuracy could vary from 90% to 96%. This might due to the fact that when front camera captures user's face, the image is in low quality and noise might be introduced as well. Future improvement on this could focus on preprocessing face images captured by camera in a fine and better way to increase quality and reduce pixel noise.

V. Conclusion

(Yibo Cheng: 50%, Chaoran Li: 50%)

The project the team develops allows the computer to decide if the face image provided to it is the one it learns and sees. This is achieved by setting up a convolutional neural network (CNN) and fine-tuning parameters. The team provides 10,000 face images of single host and uses these as input dataset. Another 13,000 face images are obtained from the University of Massachusetts Amherst where none of them is seen before. Each image is preprocessed by brightness and contrastness so that less noise and higher quality can be achieved. The team fine-tunes the model from the aspect of batch size, learning rate, number of neural layers and training dataset size and diversity. Meanwhile, effects brought by those parameters on loss and testing accuracy are fully studied and compared to find out the optimal values and setup. The final testing accuracy of the model could maximumly achieve 100% if user uploads testing images. The accuracy might be varied between 90% and 96% if testing images are real-time captured through camera. As mentioned before, capturing real-time face images through camera could introduce more pixel noise and further generate low quality images that affect learning process. Due to the time limit, the team decides to leave this issue open and improve in the future.

VI. REFERENCE

(Yibo Cheng: 50%, Chaoran Li: 50%)

Saha, S. (2021, December 7). *A Comprehensive Guide to Convolutional Neural Networks* — the ELI5 way. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Neural Network Definition. (2021, December 8). Investopedia. https://www.investopedia.com/terms/n/neuralnetwork.asp#:%7E:text=A%20neural%20network%20is%20a,organic%20or%20artificial%20in%20nature

Brownlee, J. (2020, April 16). How Do Convolutional Layers Work in Deep Learning Neural Networks? Machine Learning Mastery. https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

LFW Face Database : Main. (2013). Face Image Dataset. http://vis-www.cs.umass.edu/lfw/

Convolutional Neural Network (CNN) | TensorFlow Core. (2021).TensorFlow. https://www.tensorflow.org/tutorials/images/cnn

dlib. (2021, August 14). PyPI. https://pypi.org/project/dlib/

NumPy Documentation. (2021). Numpy Guide. https://numpy.org/doc/

OpenCV: OpenCV Tutorials. (2021). OpenCV. https://docs.opencv.org/4.x/d9/df8/tutorial_root.html