Final Project Status Report for CIS 419/519
Fake Smile Detection

Eddie Wu WUED@SEAS.UPENN.EDU
Yixuan Li LIYIXUAN@SEAS.UPENN.EDU
Kuan-Cheng Chiu CHIUK@SEAS.UPENN.EDU

Abstract

Humen can recognize a fake smile within a split second when encountering one. However, this graphical sentiment analysis procedure is difficult to be programmed into a simple algorithm. Therefore, we are proposing a method using Convolutional Neural Network to perform the fake smile detection task. Various well-known network architectures, such as AlexNet, LeNet, and ResNet, will be compared and fine-tuned to get maximum performance.The network will be constructed and analyzed using Python and TensorFlow.

1. Application and Motivation

In a world filled with advanced technologies, the amount of images taken everyday has been increasing exponentially, and as a result there are many more interesting patterns and things to detect in those pictures. As the younger generation who are quite familiar with modern technologies, we see hundreds up to thousands of pictures everyday and the different patterns of every single picture always keep us entertained. One fascinating task from those pictures would be to detect fake smiles, as it can enhance the current automated face recognition further. People should not be strangers with pictures of other people smiling, as most of the time people pose smiling facial expressions when taking pictures; however, not all those smiles are genuine, and that is why our group is interested in developing an algorithm that can help machine(s) differentiate between true smiles and false smiles.

2. Method and Approach

Due to the extremely large number of images these days, our group is able to manually find images with true smiles and those with fake smiles from the resourceful internet. Currently we have around 210 labeled images (half for fake smile and half for real smile) as our dataset, all of which are labeled by us manually. Through supervised studies with labelled data, we are focusing on specific parts of the pictures that have most significant effects in determining the genuity of smiles. These parts include the eyes, month and faces as a whole, and we used built-in python library to process these features and to help identify the difference between fake smiles and true smiles.

With a goal to detect fake smile, we are focusing on the facial expressions and muscles, and because the color of images doesn't matter we transformed all images into grey scale during pre-processing to achieve higher efficiency. Since all the images are found manually online, we use pickle to save and load them. Subsequently, We use TensorFlow to train Convolutional Neural Network(s) based on our training data, and tuning the parameters to get higher accuracy. Convolutional Neural Network works well in recognizing images because with a large number of convolutional kernels, it can implement different kinds of edge detections [1], which is the very reason why we choose Convolutional Neural Network as our classifier.

At first, we try to train the model and identify fake smiles by just using the raw input images, but the resulting accuracy is not ideal. So we decide to also include further extractions of only the face, eyes and mouths in the training and predicting process. Moreover, since the traditional LeNet doesn't guarantee ideal accuracy, we are investigating the different structures of neural networks, such as ResNet and AlexNet. Figure 1 graphically depicts a major component of our project.

2.1. Algorithm

Data Processing (See Algorithm 1):
We are first extracting faces from the resized original images, and then using those faces (instead of the original

University of Pennsylvania, CIS 419/519 Course Project.

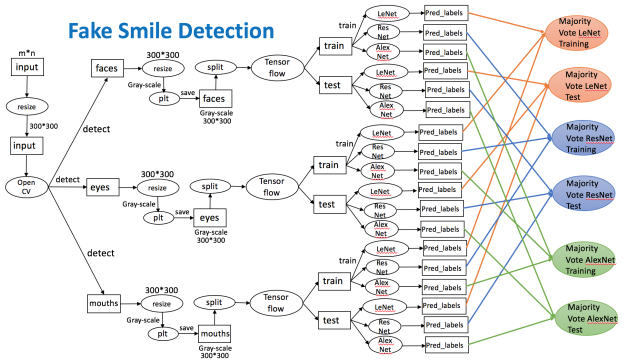


Figure 1. A major component of our project

pictures) to further extract eyes and mouths. Then we train our CNN models with faces, eyes and mouths, respectively, to obtain three trained models.

Training Phase (See Algorithm 2):
For each feature face, eyes and mouth, we tried different neural networks to see the performance of different CNNs under different features. After training different CNNs on individual features, we did a weighted majority vote on predicted labels of them to see the performance.

Since we have three trained models, we are going to produce three predicted labels (from extracted face, eye and mouth) for a single input image, then we will do majority vote to decide the single predicted label for the input.

2.2. Feature Extraction

For extracting faces, we are loading the original labelled images (real smile and fake smile) found online, selecting faces which we then store in two arrays (one for real smile and another for fake smile), and saving them separately (real and fake) in two other folders; while for extracting eyes and mouths we are just loading the previously saved face images (in the same order as the loading above to make sure that one set of source image, detected face, eye, and mouth can correspond with each other), selecting eyes from them and generating two arrays from extracted eyes and doing the same for extracting mouths. To be more specific, the extraction process is carried out using OpenCV-Python through face detection, eye detection and mouth detection. For all arrays generated, we also add labels to them such that features extracted from fake smile images have label of 1 and the rest 0. In short, from feature extraction we should have 6 sets of data (fake smile faces, true smile faces, fake smile eyes and so on) for training the fake smile detection models plus testing them.

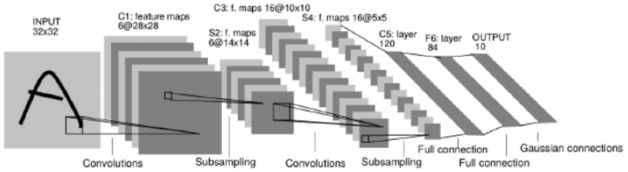
Algorithm 1 Extract Features

Input: data (image) x_i, size $m * n$
for image x_i in folder **do**
  Rescale the size of photo x_i to be 300*300
  Use OpenCV to extract face f_i from photo x_i
  Rescale the size of extracted face f_i to be 300*300
  Save extracted face f_i into a new folder in gray scale
  if OpenCV cannot detect face from x_i **then**
    Manually extract face from x_i to replace the non-face image
  end if
end for
for face image f_i in a the face image folder **do**
  Use OpenCV to extract eyes e_i from image f_i
  Rescale the size of extracted eyes e_i to be 300*300
  Save extracted eyes e_i into a new folder in gray scale
  if OpenCV cannot detect the eyes from f_i **then**
    Manually extract face from f_i and replace the non-eyes image
  end if
  Use OpenCV to extract mouth m_i from image f_i
  Rescale the size of the extracted mouth m_i to be 300*300
  Save extracted mouth m_i into a new folder in gray scale
  if OpenCV cannot detect mouth from f_i **then**
    Manually extract mouth from f_i and replace the non-mouth image
  end if
end for

2.3. Different Neural Networks

For this project, we have trained LeNet, ResNet and AlexNet and compared the accuracy under different networks.

LeNet, a pioneering work by Yann LeCun, is one of the first proven working CNNs designed to recognize handwritten letters and digits. The architecture of LeNet is as shown below:



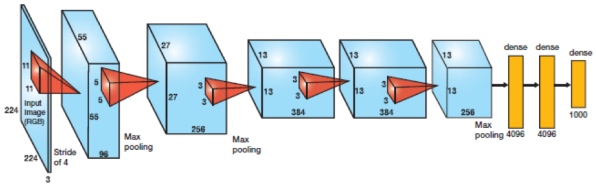
From the nature of its purpose, it is not hard to realize that the most important features to learn are the edges and corners. Therefore, it is reasonable to use only two convolutional layers since the first convolutional layer

Algorithm 2 Training different CNN

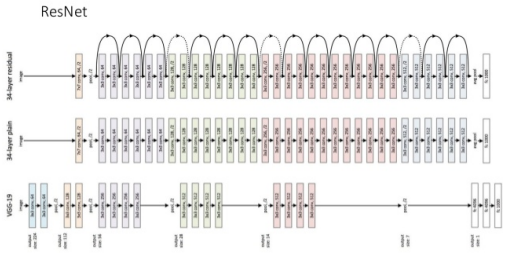
Input: data f_i, e_i, m_i, size $300 * 300$
Do $train\_test\_split$ to split training data and test data
for training data f_i in face folder **do**
  Use tensorflow to train LeNet, ResNet, and AlexNet, get training accuracy and predicted training labels under different CNNs respectively
end for
for training data e_i in eyes folder **do**
  Use tensorflow to train LeNet, ResNet, and AlexNet, get training accuracy and predicted training labels under different CNNs respectively
end for
for training data m_i in mouth folder **do**
  Use tensorflow to train LeNet, ResNet, and AlexNet, get training accuracy and predicted training labels under different CNNs respectively
end for
Do weighted majority vote on predicted training labels of face, eyes and mouth all under LeNet, ResNet, and AlexNet, get training accuracy and predicted training labels respectively
for test data f_j in face folder **do**
  Use trained LeNet, ResNet, and AlexNet, get test accuracy and predicted test labels under different CNNs respectively
end for
for test data e_j in eyes folder **do**
  Use trained LeNet, ResNet, and AlexNet, get test accuracy and predicted test labels under different CNNs respectively
end for
for test data m_j in mouth folder **do**
  Use trained LeNet, ResNet, and AlexNet, get test accuracy and predicted test labels under different CNNs respectively
end for
Do weighted majority vote on predicted test labels of face, eyes and mouth all under LeNet, ResNet, and AlexNet, get test accuracy and predicted test labels respectively

activates on edges, and the second layer should recognize combinations of edges, such as corners.

To classify objects other than letters and digits, a more complicated structure is required. An example is the AlexNet, with the architecture shown below:



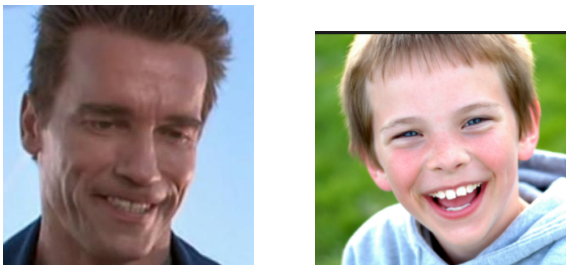
A major difference of AlexNet from LeNet is that more convolutional and fully connected layers were used. This means that AlexNet is more capable of learning higher levels of features, such as shapes, formed by combinations of corners and edges. In addition, by introducing ReLu as activation function, AlexNet is able to lessen the vanishing gradient problem when constructing deep networks. However, incorporating ReLu does not diminish the vanishing gradient issue. He et al. proposed the concept of residual module (hence the name). By constructing a direct connection between the input and output inside a residual module, ResNet is able to stack deeper layers without encountering severe vanishing gradient difficulties. In practice, a 34 layered network is used and performs generally better than LeNet and AlexNet. The architecture of ResNet is as shown below:



3. Data sets and Evaluation

3.1. Dataset

To the knowledge of authors, there is no open source database of labeled real and fake smiles on-line. Therefore, the 210 images that we currently have were collected and labeled by hand. These images mainly come from Facebook, Google image search, Twitter and Imgur (among some other available websites). An image is categorized into *fake* if all the authors agree that the smile is not genuine. Otherwise, it is labeled *true*. The authors will also take additional information of the image into account, such as corresponding posts, hashtags, or comments, to



(a) Fake Smile (b) True Smile

Figure 2. An example of labeled smile

help distinguish the smile. An example of a fake and real smile we labeled is shown in Figure 2. In general, pictures taken formally (passport photos, student ID photos) tend to be associated with fake smiles, as well as those of politicians, while some other pictures are harder to be classified and hence we authors need to make more efforts identifying.

We are using the methods introduced in Feature Extraction section above to load in the images from our compiled dataset, and there were ample instances where OpenCV-Python module is unable to detect a face, an eye or a mouth from an image. In these cases, we would first approximate the average position of the desired feature (face, or eye or mouth) across all source images and then use those positions to manually extract the most likely positions of the feature.

3.2. Evaluation

For AlexNet and ResNet, due to our group members' personal computer's computational power limitations, following the original number of neurons in the last fully-connected layers was not feasible. The proposed 1000 and the two 2048 followed by 1000 activation units proposed by the authors of ResNet and AlexNet, respectively, were required to be scaled down. Therefore, we used cross-validation to find the best combination of the scaled-downed parameters, and then use our optimal classifier to predict on our training and test data to get training accuracy and testing accuracy. After cross-validation, the number of neurons for ResNet's activation layer were 500 and for AlexNet were 64-64-32.

From the results run locally through our machines, we found that separating the training data into 5 batches provide reasonable predictive powers. All the networks were trained with different number of epochs until their training and testing accuracy are stable. We also discovered that the testing accuracy was the highest when using the whole face as the input data. A summary of the three models' performances, with the whole face as input, is shown in Table 1. As can be expected, the LeNet achieved the lowest testing accuracy due to its structure of having least number of convolutional layers and kernels. AlexNet, with an increased number of convolution operations, more kernels for each convolutional layers, and two more fully connected layers, managed to achieve a three percent improvement over LeNet. Finally, with the most complicated and deepest structure, ResNet is the best performing CNN architecture of the three. With 32 convolutional layers and up to 512 kernels per convolutional layer, it has an over ten percent improvement when compared to LeNet and eight percent to AlexNet. As a result, we conclude that our optimal model is ResNet with the whole face as the input data.

Table 1. Evaluation on CNNs for face

		precision	recall	F1	training accuracy	testing accracy
LeNet	True	0.55	0.63	0.59	1.0	0.60
	Fake	0.66	0.58	0.61		
ResNet	True	0.75	0.75	0.75	1.0	0.71
	Fake	0.71	0.71	0.71		
AlexNet	True	0.65	0.52	0.58	1.0	0.63
	Fake	0.62	0.74	0.68		

4. Conclusions and Impact

By using our compiled dataset of images with added labels, we successfully detected faces, eyes and mouths which we then use individually to train numerous CNN models, and then determined the final predicted label of an input image by majority voting (among three feature models). Different network architectures of CNN were investigated as well, including LeNet, ResNet and AlexNet. We found out using ResNet and the whole face as input data leads to the most accurate result. The test accuracy is not very high because the difference between fake smile and real smile is small, we need more complex CNN to solve this problem, which is unfortunately unable to be carried out by our own laptops.

CNNs with appropriate network structure are proven to be capable of differentiating fake smiles from true ones when the faces, eyes and mouths features are extracted from the original input pictures, and this is significant because it signifies a further advance on the capability of machines in the already rapidly evolving machine learning oriented world. From our experimentation and results, it helps to prove that there's really no limit to what the machines can do if they are given enough good quality data for training purposes, and the society will continue thriving with more discovered functionalities of machines.

5. Citations and References

[1] Zeiler M.D., Fergus R. (2014) Visualizing and Understanding Convolutional Networks. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision ECCV 2014. ECCV 2014. Lecture Notes in Computer Science, vol 8689. Springer, Cham

[2] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[3] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[4] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[5] Michael Nielsen, Neural Networks and Deep Learning, Chapter 6.

Acknowledgments

The authors would like to thank Prof. Eric Eaton for providing valuable insights regarding the approaches to analyze different CNN structures. While completing this project, the authors also consulted the following resources:

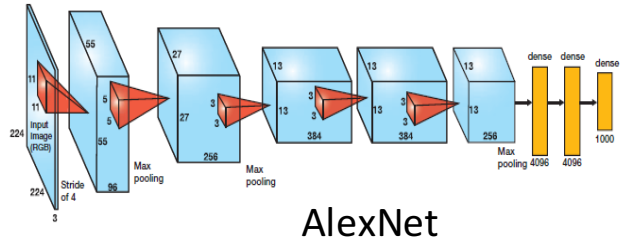
Introduction to Convolutional Neural Networks for Visual Recognition. A Stanford course on CNN.
https://www.youtube.com/watch?v=vT1JzLTH4G4&index=1&list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv

TensorFlow tutorial:
http://cv-tricks.com/tensorflow-tutorial/training-convolutional-neural-network\-for-image-classification/

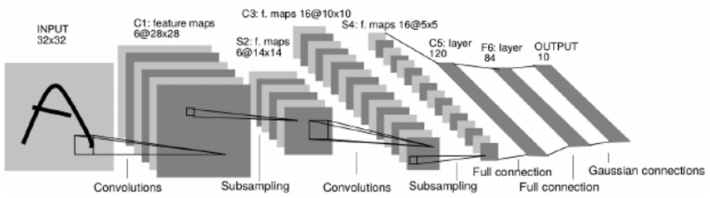
Fake Smile Detection

Goal

- **Our data:**
- Since there is no existing dataset for fake smile and real smile, we collect data (images) from social website such as Facebook, Twitter and Google, and label them manually. The distribution of our data is even, as we have almost 50% true smile images and 50% fake smile images, all of which have identifiable facial features.
- **Technologies:**
- Convolutional Neural Network, Tensor Flow, Keras, OpenCV, LeNet, ResNet, AlexNet, Weighted Majority Vote
- **Evaluation:**
- Training accuracy, test accuracy, precision, recall and F-1 score under different CNNs.
- Test accuracy for LeNet ~0.6, ResNet ~0.71, AlexNet ~0.63.
- Optimal model: ResNet with whole faces, training accuracy ~1.0, test accuracy ~0.71.
- Performance is unfortunately restricted by our laptops which are unable to run more complex CNNs.



AlexNet



LeNet

ResNet

Fake Smile Detection