Implementing a Facial Recognition Login Method

Deemal Patel
Northwestern University
5/2/21

Abstract

Our company is losing out on potential sales. Many of our customers abandon their shopping carts because they cannot remember their login password and refuse to reset it. Using neural networks, we can implement a face recognition login model using neural networks to solve our issue of missing out on sales. CNNs are the most appropriate model for image classification and detection, and performing various tests on different nodes and optimizers will help create the most accurate model. Based on the recommended model, a few tweaks can be made to get the accuracy we want, which will lead to more customers making purchases from our company.

Introduction

According to a study done on 500 respondents, about 78% of these people had to reset the password for one personal account in the last 90 days. 57% of the 500 claimed that they had to do a password reset for work account (Malik 2019). Users are creating so many accounts that it is becoming difficult to keep track of their password for a specific account. Many of these users sometimes do not have the patience to reset their password and move on from whatever they were trying to log in to. Businesses have been known to suffer from e-commerce sites because customers will add stuff to their shopping carts, and then finally, when it is time to check out, they cannot seem to remember their password and begin wondering if they need that product. Roughly 19% of customers end up abandoning their shopping carts because they do not feel the need to reset their password and purchase what they want (Johnson 2017). With smartphones as the most ordinary phone nowadays and the rise of machine learning,

companies are now implementing facial recognition login, making it easier for customer usage and satisfaction.

Due to many users abandoning their shopping carts and not wanting to reset passwords, management has asked me to develop a neural network model that will recognize the user's face and let them log in that way. The company has noticed that about 20% of shopping carts have been abandoned each month, resulting in missing out on potential sales for the company. We have been generating less revenue by missing out on these sales, preventing us from reaching our new sales goals.

Literature Review

Recent work by Agrawal and Mittal (2019) on using CNN for facial expression recognition is similar to the research I am conducting. The duo used a CNN architecture to determine an individual's facial expressions. They wanted to create a strong model to develop a human-like accuracy for detecting faces. They believed that to create a significant impact on accuracy, they needed to manipulate the kernel size and the number of filters each layer has. The FER-2013 dataset was used for their study. There was a kernel size list and a number of filters list, and the duo performed each permutation and compared the results to see which one performed the best. They used an ADAM optimizer which proved to be the best choice, and saw that the best combinations from their testing were 16 filters with a kernel size of 4, 32 filters with a kernel size of 8, 4 filters with a kernel size of 16, and 2 filters with a kernel size of 32. They were able to achieve a “human-like accuracy of 65%.”

Coskun et al. (2017) also performed face recognition based on convolutional neural networks, but their focus was on adding normalization operations to two of the layers for

architecture. Their main contribution was to obtain a powerful recognition algorithm that was available to develop a high accuracy score. Since they wanted to focus on the normalization of two layers, they decided to implement Batch Normalization. The dataset came from the Georgia Tech face database that contains images of 50 individuals taken in two or three sessions between June 1st, 1999, to November 15th, 1999. They implemented an eight-layer CNN consisting of 5 convolutional layers and three pooling layers and used different sized images to perform their test. They had a top-1 and a top-5, where top-1 accuracy checks if the top class is the same as the prediction label. They were able to generate a 94.8% accuracy with an image size of 64x64x3. That same image was able to get 98.8% for the top-5 result.

Finally, Yan et al. (2017) conducted their face recognition algorithm based on Convolution Neural Networks. Their network consisted of three convolution layers, two pooling layers, two fully-connected layers, and one Softmax layer. They ended up using a stochastic gradient descent algorithm to train the model. In order to solve the issue of overfitting, they implemented a dropout method as well. The team used the ORL face database and the AR face database to train their model, and they got incredible accuracy scores. They used a 5x5 kernel size along with a Xavier weight initialization. Their pooling layer had a 2x2 size, and the step size was 2. They resized the images to be a 32x32 pixel and scaled their images from 0-255 to 0-1. Their experiments and computations were able to get a testing accuracy of 98.95% for the ORL database and 98.30% for the AR database. For all the data for ORL and AR, they got 99.82% and 99.78%, respectively.

Methods

To create a strong face detection model, I decided to use the CIFAR-10 data set, which contains about 60,000 rows of data that pertain to different classes of real-world objects. In order to thoroughly conduct my research on developing the proper model, I decided to create multiple models to compare and contrast the results so that I can use the best model to create our face detection model. The model used for my research consists of two types of neural networks. The first network is a deep neural network, or DNN, and convolutional neural network, or CNN. There is some difference between these two types of neural networks. The first is that DNN encompasses any form of deep learning. It is a general use neural network that can be applied to almost any type. CNN is designed specifically for computer vision, such as image detection and classification, which we perform here. These models are developed from a multilayer perceptron, or MLP, which uses a forward feed method. Forward feed means that the data will be passed on to the next layer for training. These models train by using backpropagation, making a neural network the best choice for creating a face detection program. I performed a total of 13 experiments to see what worked and what did not thoroughly. Four out of the 13 models used two layers, another four used three layers, and the last five used four layers. The reasoning behind this is that I can perform a thorough analysis for each layer and determine which one performed the best to perform hyper tuning on that model and get better accuracy results. Each model uses a ReLU activation which is the most commonly used activation function in neural networks and can be applied to any architecture.

The biggest tool that helped me perform my research is TensorFlow, a free, open-source software used for machine learning. A package that comes with TensorFlow is the Keras API, which contains tools to help build our MLP. One of the biggest packages from Keras we use is the Dense class which is used in both DNN and CNN, and the Conv2D class, which is used for

our CNN models. The Conv2D will help us choose our filters, kernel size, and strides for each of our layers in the CNN to get better accuracy scores. A filter is a matrix of weights slid over an image to produce a filtered output. Kernel size is the height and width of the filter mask. In my case, I used a 2x2 kernel, so then our filter will shift by that size. Stride is a filter parameter that modifies the amount of movement over an image. I set my stride to be a 1x1 so that the filter will move one pixel at a time. Finally, my output layer is set up to use a SoftMax activation which is essentially the go-to activation when working with multiclass classification problems.

Another package that I used is the seaborn package and matplotlib.pyplot package. These packages allow me to look at the dataset and see how the images look for training. Finally, the last package that I used is the sci-kit learn package. Sci-kit learn contains a metrics package that allowed me to develop a confusion matrix to check the true positives, false positives, etc. The other function from the metrics package I used is accuracy score which allowed me to see how well my model worked on test data, essentially new data that my model has not seen. From that, I can tell whether the model is overfitting or underfitting.

The first step to performing any task is to perform exploratory data analysis or EDA. Performing this task will help me get insight into what my data contains and if I need to do any preprocessing before running my models. Each column of the data is a pixel value of the image ranging from 0 to 255. With this wide scale, it makes it a little difficult for our model to learn, so I needed to prepare my data so that all the values for each column are in a reasonable range, such as from 0 to 1. Once the model is complete, I plan to revise the model over time and get more and more users using the face detection system, meaning more data to train the models making it more accurate over time. I plan to revise the model every month so that the accuracies do not drop due to drift.

Results

The first two experiments I conducted used the DNN architecture to see how well that model would work with image classification. The first experiment uses the Adam optimizer with two layers. Each layer consists of 256 nodes because I felt that advanced data such as images and a large number of nodes would be needed to perform good classification. From the summary table in Appendix A, I got a training accuracy of 70%, a testing accuracy of 53%, and a validation accuracy of 56%. The model ran very fast by completing in about two minutes. However, the model overfitted because the test and validation accuracy are not close to the training. Therefore this model could not be used unless I perform some hyper tuning. The confusion matrix shows numbers all over the place, meaning there was not one specific area that trouble classifies. The second experiment was also a DNN, but I decided to use three layers, each with 256 nodes. This model also took two minutes, but the accuracies are about the same as the first experiment, meaning adding some layers did not improve performance. My initial hypothesis is that DNN will not be useful in image classification. Experiments three and four used the CNN architecture with two and three layers, respectively. I again used the Adam optimizer, and the accuracy scores were a lot better in this case. There was a training accuracy of 87% and 83%, a validation accuracy of 75% and 80%, and test accuracy of 74% and 78%. Both took almost an hour to run but judging by the accuracies, and experiment four performed better because there is less overfitting and should be the model for me to do more in-depth research with.

I decided to rerun experiments one through four, except this time, I added L2 regularization to the model to see if that would change the accuracy. Regularization is a technique that makes slight modifications to the learning algorithm such that the model

generalizes better. This, in turn, improves the model's performance on the unseen data. I decided to label these experiments 5a to 5d. The accuracy ended up dropping for experiment 5a compared to experiment one, but the overfitting reduced greatly from 14% to 8%. The same thing was witnessed in experiment 5b. After seeing these results, I was able to completely rule out DNN as the choice of architecture for our face detection model. 5c and 5d showed a drop in the training accuracy, but the overfitting reduced greatly. These two models may be explored a little more to see if we can increase the accuracy scores.

In the final five experiments, I decided to focus on CNN architecture because it was proven that DNN did not get the job done. For three of the experiments, I used the SGD optimizer, and for the other two, I used Adam. Every one of the experiments had regularization in the model. From what I saw in experiments 5a to 5d, the overfitting reduced greatly, and I believed it would help get better accuracy scores for future experiments. For experiment 6, I had four convolutional layers with 32, 64, 128, and 256 strides in the respective layer. I also had a max-pooling layer of 0.3 after each. Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter. Thus, the output after the max-pooling layer would be a feature map containing the most prominent features of the previous feature map. I used a learning rate of 0.01 and a momentum of 0.9. The learning rate of 0.01 lets my model train a little faster because the computational heavy program and momentum would help my model escape any local minima. This experiment gave me a training accuracy of 69.90%, a testing accuracy of 68.84%, and a validation accuracy of 70.83%. These accuracy are not what we want, but some hyper tuning can be improved. Experiment seven used 32, 32, 64, and 128 filters on their respective layer, and this time I used the Adam optimizer as I did in the first five experiments. This experiment gave me an 89.93% training accuracy, 81.89% testing

accuracy, and 82.23% validation accuracy. These are many better numbers, but there is some overfitting going, but with regularization and other techniques, the overfitting can be improved. Experiment eight was exactly like experiment seven, but I used an SGD optimizer, and my fully connected layer had 1024 and 512 nodes in their respective Dense layers. This experiment gave me 88.03%, 79.42%, and 80.53% in the train, test, and validation accuracy. Some improvement could have been made, such as more regularization or some other technique. Experiment nine used an SGD, but this time I had 64, 64, 128, and 256 filters for the respective convolutional layers, and the overfitting increased immensely. This experiment's accuracy was 93% train, 78.595 test, and 79.5% validation. Finally, experiment ten ran the same way, except I used 512 node and 128 nodes for the Dense layers with an Adam optimizer and got 91.70% train, 82.32% test, and 83.27% validation. I noticed that when I lowered the nodes of my Dense layer, the overfitting dropped, and in most of the experiments, SGD had less overfitting than Adam.

Conclusion

The best model of all the experiments was experiment 7 because of the high accuracy score with some overfitting. I recommend this model to management for our face detection model because of the good ratio of accuracy and overfitting. Based on my research, one of the best ways to reduce overfitting is to apply more training examples, add regularization, and a few other methods. Because this model is already high in accuracy, it will not need much tweaking in the parameters. This means that if we can fine-tune this model just a little bit, we will easily be able to have our users login with just face detection, and we will not miss out on potential sales. These potential sales will help our company grow and meet our new sales goals.

References

Agrawal, Abhinav, and Namita Mittal. “Using CNN for Facial Expression Recognition: a Study of the Effects of Kernel Size and Number of Filters on Accuracy.” *The Visual Computer* 36, no. 2 (2019): 405–12. https://doi.org/10.1007/s00371-019-01630-9.

Coskun, Musab, Aysegul Ucar, Ozal Yildirim, and Yakup Demir. “Face Recognition Based on Convolutional Neural Network.” *2017 International Conference on Modern Electrical and Energy Systems (MEES)*, November 15th, 2017. https://doi.org/10.1109/mees.2017.8248937.

Johnson, Tim. "When Customers Forget Their Passwords, Business Suffers." Phys.org. Phys.org, June 20th, 2017. https://phys.org/news/2017-06-customers-passwords-business.html#:~:text=%22Twenty%2Done%20percent%20of%20users,day%2C%22%20the%20study%20found.

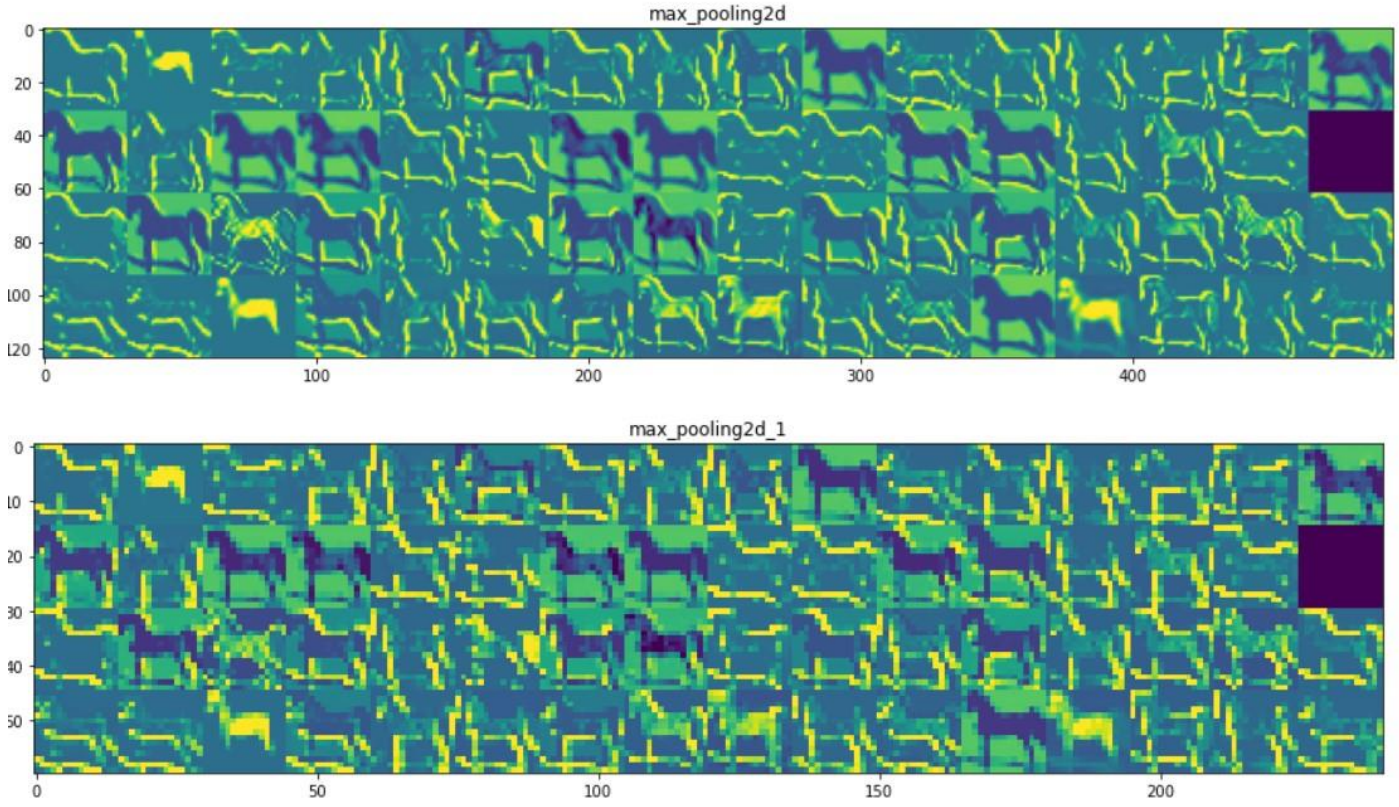
Malik, Daniyal. "Study: 78 Percent People Forget Their Passwords And Then Go For Reset!" Digital Information World, December 23rd, 2019. https://www.digitalinformationworld.com/2019/12/new-password-study-finds-78-of-people-had-to-reset-a-password-they-forgot-in-past-90-days.html.

Yan, Kewen, Shaohui Huang, Yaoxian Song, Wei Liu, and Neng Fan. “Face Recognition Based on Convolution Neural Network.” *2017 36th Chinese Control Conference (CCC)*, July 26th, 2017. https://doi.org/10.23919/chicc.2017.8027997.

APPENDIX A:

Experiment	Architecture	Optimizer	Layers	Train Accuracy	Validation Accuracy	Test Accuracy	Regularized?	Computation Time
Experiments 1	DNN	Adam	2	70.0%	56.6%	53.03%	No	0H:01M:56.462S
Experiments 2	DNN	Adam	3	70.80%	53.80%	53.13%	No	0H:02M:0.888S
Experiments 3	CNN	Adam	2	87.35%	75.67%	74.19%	No	0H:53M:36.442S
Experiments 4	CNN	Adam	3	83.71%	80.70%	78.6%	No	1H:03M:00.127S
Experiments 5a	DNN	Adam	2	62.26%	54.07%	52.94%	Yes	H:01M:49.875S
Experiments 5b	DNN	Adam	3	63.98%	55.50%	54.12%	Yes	0H:02M:11.632S
Experiments 5c	CNN	Adam	2	80.97%	75.03%	74.83%	Yes	0H:42M:59.543S
Experiments 5d	CNN	Adam	3	82.02%	77.87%	76.12%	Yes	1H:00M:39.080S
Experiments 6	CNN	SGD	4	69.90%	70.83%	68.84%	Yes	0H:17M:53.535S
Experiments 7	CNN	Adam	4	89.93%	82.23%	81.89%	Yes	1H:07M:57.852S
Experiments 8	CNN	SGD	4	88.03%	80.53%	79.42%	Yes	2H:29M:02.813S
Experiments 9	CNN	SGD	4	93.01%	79.5%	78.69%	Yes	4H:39M:48.266S
Experiments 10	CNN	Adam	4	91.70%	83.27%	82.32%	Yes	3H:38M:15.426S

Summary Table of all the experiments with computation times



Outputs from 2 max-pooling filters