Facial Recognition System using a Convolutional Neural Network

Hannah Chookaszian and Nathan Diekema

Introduction

For our final project we designed and implemented a face recognition software. This is an especially interesting project because facial recognition has become an incredibly relevant tool in modern technology. Today, nearly all smartphones on the market use some form of face identification technology. It has become a popular method for unlocking devices, opening sensitive apps, or even enabling payment methods. Face recognition has also, for better or for worse, become an integral tool in law enforcement. These agencies have access to local, state and federal databases in which they can query an image of a potential person-of-interest in hopes of identifying them (Face Recognition, 2020). As helpful as this may be, it relies heavily on the accuracy of the recognition system and the quality of the database: both of which may have internal biases based on race and skin color. This ethically questionable reliance on technology presents a challenging problem given the possibility of false positive results which could ultimately lead to a waste of resources, or worse, the misidentification or even prosecution of innocent people. As a result, we are putting an emphasis on designing our system to be as accurate and unbiased as possible and to gain some insight into the reasons for such biases.

For the sake of comparison, we decided to design and implement two different loss methods for our neural network. The cross-entropy and triplet loss methods were both implemented and compared for accuracy and flexibility. The triplet loss method was chosen for the project because of its flexibility in adjusting to larger datasets and popularity in industry.

In terms of the software architecture, the data flow can be described as follows. First, an image is chosen for analysis with a visible face, or faces. The software then detects every visible face in the image. Each face is transformed to a fixed size and dimension so that every face is identical in size. If the face is rotated in the image, it is transformed such that the detected eyes lie parallel and are vertically centered in the frame. This step is important to maintain continuity when measuring and comparing faces. Next, the transformed face is cropped and passed through the convolutional neural network. For the cross-entropy model, the network will output an array of probabilities that the input image is most similar to each class. In the case of the triplet loss model, each class will be assigned 128-d vector encoding. These vectors are based on measurements taken between key facial features. Finally, each face is classified based on the previously assigned vectors. If the feature vectors of the face are within a certain threshold of similarity to one of the faces in our dataset, the classifier matches the names and will output the person's name accompanied by the percentage confidence of that determination.

Background

Face recognition is a popular and intriguing problem in the world of computer vision which means there are plenty of helpful resources available on the internet. People have tried an array of different methods with varying results. The basic pipeline that most face recognition projects follow is: detecting faces, transforming them, identifying key features, and comparing the features to encodings in a database (Geitgey, 2020). Studies suggest that the most important steps in this process is the face alignment, landmark localization, face frontalization, and facial landmark detection (Le, 2017). These features will be important to focus on to improve the accuracy of our facial recognition project and will assist in the identification of faces in varying situations. Landmark localization will be especially important when building the system because

it assists in consistent identification of faces (Geitgey, 2020). A common method for landmark localization uses ensemble regression trees and was invented by Vahid Kazemi and Josephine Sullivan (Kazemi, 2014). This method uses 68 landmarks on the face to transform the picture so the facial features are in the same place each time. This method has proved to be reliable in transforming faces. Another good reference for our project is from a project called OpenFace which uses a neural network that produces 128 measurements from the face to identify it (Amos, 2016). This project shows a low false-positive rate and has large amounts of information on Github to reference for our project.

In addition to preprocessing, the loss function used is also important to consider. Cross-entropy loss function takes an output list of percentages that all add up to one. The cross-entropy model measures the distances from the truth values and assigns each class to a true or false value. This is less effective with large databases because it is harder to add classes (Koech, 2020). Another loss function that is commonly used is triplet loss. The triplet loss method uses an anchor image, a negative image, and a positive image. This method pushes the encodings of the negative images away from the values of anchor encoding and tends to "cluster" the encodings of the positive images. After doing this hundreds of thousands of times the model should be able to encode any image with an accurate embedding for that person. This loss model is especially reliant on performing a large amount of epochs. Moreover, this method is far more effective on larger datasets because it is flexible with the number of cases (Salgado, 2019).

Data and Methods

Like many computer vision applications, face recognition presents a problem that can be tackled with a wide range of varying approaches. Existing strategies range in complexity and

difficulty. The LFW, or “Labeled faces in the Wild” database, provides thousands of images of people with labeled faces (Face Recognition, 2020). This database was used for the testing and training of our neural network. Two different loss models were tested for this project in order to compare performance and achieve the highest possible accuracy. Each model was accompanied by a different neural network. For the cross-entropy loss model, the neural network was a 7-layer convolutional neural network. The triplet loss method was also implemented and this was accompanied by a 5-layer convolutional neural network.

Our first approach implemented a sparse categorical cross-entropy loss function with a softmax activation. This model returns a multinomial probability distribution applying to all classes in the provided dataset (Brownlee, 2020). This method had plenty of documentation and was quite simple to implement and test and is typically quite accurate for datasets with a manageable number of classes. After countless design iterations we achieved the best testing accuracy at 85.76%. Our final model consisted of seven convolutional layers with four applications of max pooling and periodic batch normalization and dropout functions to reduce overfitting. The issue that was found with the softmax method is the lack of flexibility and the significant decrease in accuracy when dealing with larger datasets. The second loss function method tested is the triplet loss method.

The second approach employed a semi-hard triplet loss function to learn good embeddings for the images. This method was considerably more difficult to implement. Despite tensorflow having a triplet loss function included in its framework, the amount of helpful documentation was limited. The triplet loss method allows for variable numbers of classes. This allowed for more faces to be enrolled in the database while still maintaining high accuracy. Triplet loss uses an anchor photo for each class which is compared to other photos. A positive

match is added to the same class as the anchor photo and a negative is placed in a different class (Moindrot, 2018). This process is all done automatically via online triplet mining performed by the adapted TensorFlow functions.

Evaluation

The evaluation of our project is concerned with the accuracy of our recognition software and the performance difference between the CNN models. We tested the system by splitting a large open-source database of labeled faces into training, testing, and validation datasets with an equal distribution of classes in each. To properly prepare the LFW database for evaluation, we eliminated all classes (or people) from the dataset that had less than 15 face images associated with them. This left us with approximately 3000 images and 62 classes to train and test with. This made it simple to find the accuracy of the implemented systems. For the cross-entropy model the testing was relatively easy. Since each face in the testing database is labeled, we were able to pass many images through and compare the result with the actual result until a concrete number for accuracy was reached.

Evaluation for the triplet-loss function required a few more steps. First, the model was trained using the semi-hard triplet loss function adapted from TensorFlow. Next, we transformed the encodings of each trained image into a 2-d array and plotted them to track the development of the cluster formations. The encodings were then averaged for each class and added to a dictionary to act as the anchor embedding for each particular class. A k-nearest-neighbor classifier was used to predict images from the testing dataset. The accuracy was then determined by manually dividing the number of correct results by the length of the test set.

Results

As discussed earlier, our first approach employed the use of a categorical cross-entropy loss function. This model is very straightforward as the number of outputs directly correlates to the number of classes in the input dataset. Our final design iteration for this model achieved a high accuracy of 85.76%. The design process for this particular model consisted of altering the shape of the neural network, the number of epochs, and the learning rate. In terms of the shape of the neural network we tried all kinds of combinations for the number and size of convolutional layers and played around with batch normalization and dropout to deal with overfitting. The neural network was trained after each design iteration and changes were made based on the trend of the validation and training loss and accuracy. Our final design iteration consisted of 7 convolutional layers with 4 max pooling layers interspersed between to reduce dimensionality. Every pooling layer was followed by batch normalization and dropout to mitigate overfitting of the results. These layers were followed by two dense layers and a final layer with a softmax activation.

Our second approach, as discussed earlier, utilized triplet loss. In this model, triplet loss is used to learn similar embeddings (also known as 'encodings') for faces of the same people. As discussed earlier, the evaluation of this model was slightly different than that of the cross-entropy model. Instead of focusing on the trends of validation accuracy and loss, we mainly determined the success of the triplet loss model based on the visual representation of the embeddings. Figure 1 illustrates the distribution of embeddings created for images in a section of the LFW dataset.

The plot on the below represents the distribution before applying semi-hard triplet loss and the plot on the right is after. Each color represents a different person or class. Before training, the image encodings appear random with no observable pattern. Contrarily, after

applying 500 epochs of semi-hard triplet loss, the embeddings for each image have been clearly grouped together by class. Admittedly, there are still significant overlaps between classes which can be attributed to our less than perfect design. We did our best to mitigate this effect by using a K-neighbors classifier for the predictor but some of these are hard to avoid. That being said, these results were very exciting to see as our first few iterations were not nearly as accurate. To reach the final design iteration we followed a similar process to the cross-entropy model by altering and testing different components of the network.

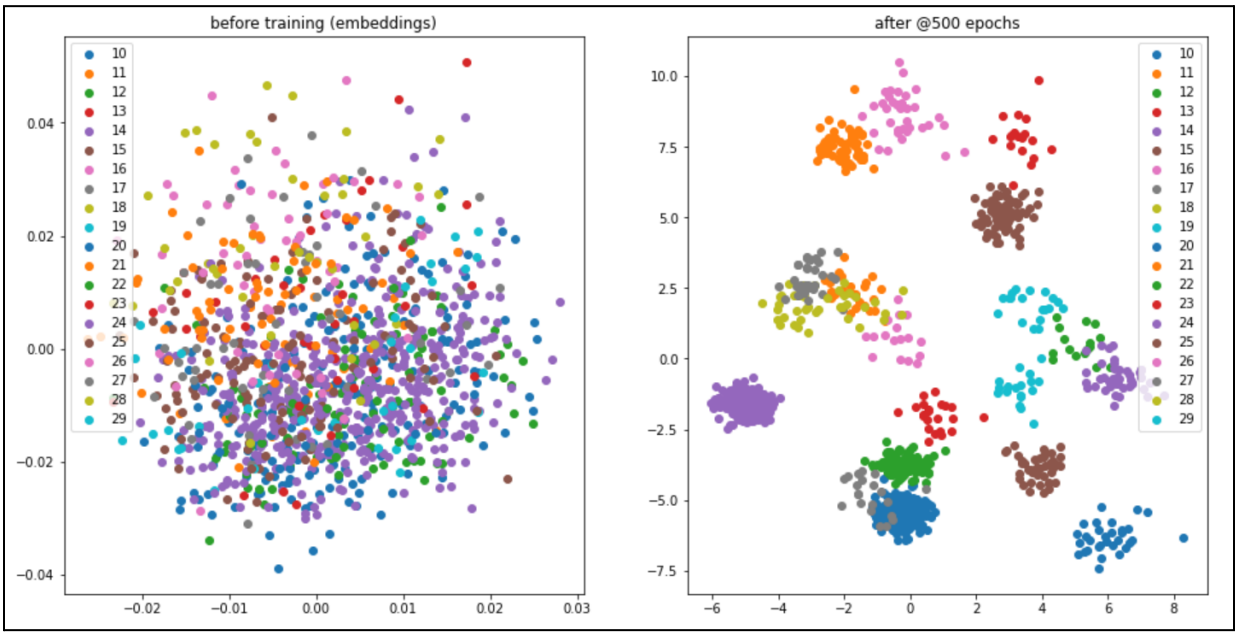

Fig 1: Final design iteration results - dataset distribution before and after triplet-loss training

Outlook

The goal set for this project was originally to achieve a detection accuracy over 95% using the LFW database. Unfortunately, this was not achieved by either loss evaluation method, but after broadening our understanding of the topic, this makes sense. Achieving high accuracies for a large dataset like LFW takes a lot of time and patience. This project made us realize the

complex difficulties behind face recognition algorithms and furthered our understanding of how it could be problematic in law enforcement.

Throughout the duration of the project, we ran into a number of obstacles that in hindsight probably could have been avoided. For instance, the implementation process for the triplet loss function turned out to be much more difficult than expected. The loss function is not only difficult to grasp, but the function included in TensorFlows framework was lacking in proper documentation, especially for our design model. We ended up having to adapt a few of the TensorFlow functions to fit our design. Another challenge was figuring out how to access the webcam from google colab. We were eventually able to solve this by piecing together some code from other sources to achieve acceptable results. Finally, we believe that our accuracy was impacted by the data imbalance in the LFW database is the imbalance in class size but were not able to address this in time.

All in all, we were able to successfully design, evaluate and compare two different loss functions for face recognition, both achieving decent accuracy rates on the LFW dataset. In addition to that we were able to create rudimentary custom datasets by capturing, detecting, and transforming faces in images captured from a webcam. Despite having a few issues here and there, we ultimately learned alot about face detection algorithms, the role of loss functions and overall gained a better understanding of neural networks. Given more time, the opportunities for improvement are endless. First, the accuracy of both models could be improved drastically with more testing and research. Additionally, in terms of custom datasets, we could implement live face recognition from a real-time webcam video. Finally, we could make the code much more user friendly by making it more robust and allowing users to automatically make custom datasets by collecting a set of images from the webcam without needing to understand the code.

References

Abhimanyu1996. (n.d.). Abhimanyu1996/Face-Recognition-using-triplet-loss. Retrieved December 05, 2020, from https://github.com/abhimanyu1996/Face-Recognition-using-triplet-loss/blob/master/Face_Recognition.ipynb

AdrianUng. (n.d.). AdrianUng/keras-triplet-loss-mnist. Retrieved December 05, 2020, from https://github.com/AdrianUng/keras-triplet-loss-mnist/blob/master/Triplet_loss_KERAS_semi_hard_from_TF.ipynb

Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (2016, June). OpenFace: A general-purpose face recognition library with mobile applications. Retrieved 2020.

Brownlee, J. (2020, June 23). Softmax Activation Function with Python. Retrieved December 04, 2020, from https://machinelearningmastery.com/softmax-activation-function-with-python/

Classes¶. (n.d.). Retrieved December 04, 2020, from http://dlib.net/python/index.html

Face Recognition. (2020, August 25). Retrieved December 04, 2020, from https://www.eff.org/pages/face-recognition

Geitgey, A. (2020, September 24). Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning. Retrieved December 04, 2020, from https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. *2014 IEEE Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2014.241

Koech, K. (2020, November 28). Cross-Entropy Loss Function. Retrieved December 05, 2020, from https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

Kortli, Y., Jridi, M., Falou, A. A., & Atri, M. (2020). Face Recognition Systems: A Survey. *Sensors, 20*(2), 342. doi:10.3390/s20020342

Labeled Faces in the Wild Home. (n.d.). Retrieved December 04, 2020, from http://vis-www.cs.umass.edu/lfw/index.html

Le, Y. (2017). The review and results of different methods for facial recognition. *IOP Conference Series: Materials Science and Engineering, 231*, 012021. doi:10.1088/1757-899x/231/1/012021

Moindrot, O. (2018, March 19). Triplet Loss and Online Triplet Mining in TensorFlow. Retrieved December 04, 2020, from https://omoindrot.github.io/triplet-loss

Salgado, F. (2019, July 11). Softmax vs Triplet loss: Fashion-MNIST. Retrieved December 05, 2020, from https://www.franciscosalg.com/triplet-loss/

Singh, G., Rosebrock, A., Domínguez, D., Jain, S., Sanaullah, Kane, G., . . . Kavatkar, A. (2020, April 18). Face recognition with OpenCV, Python, and deep learning. Retrieved December 04, 2020, from

https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

TensorFlow Addons Losses: TripletSemiHardLoss. (n.d.). Retrieved December 05, 2020, from https://www.tensorflow.org/addons/tutorials/losses_triplet

Vaessin, H., Jesudas, Kacha, B., Rosebrock, A., Futami, T., Shah, R., . . . Loyange, T. (2020, April 18). OpenCV Face Recognition. Retrieved December 04, 2020, from https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/