Facial Verification in Deep Learning

Erik Kelemen

Yichi Zhang

Daniel Gao

Naga Nageswaran

Ashish Manda

University of Maryland
College Park, MD 20742

1 Introduction

Detecting whether or not the same person is in multiple images is a famous problem in image classification, with implications for security, biometrics, fraud detection, user authentication, and more. The importance of facial recognition to many different sectors of employment has increased ever since it was first developed. However, since it is a constantly evolving technology, state-of-the-art software approaches are far from infallible, leaving much room for improvement. Large-scale attempts, such as FaceNet by Google, have already been mounted to approach this problem, and we think this is a very interesting problem to attempt to solve with deep learning procedures and neural networks. For this research, we study facial verification with the "Same Person or Not" problem – given a set of k images (k >= 2), determine whether or not they are pictures of the same person. For example, given 2 pictures of Barack Obama's face (one smiling and one looking grim), the model must be able to confirm that both of the pictures are of the same person. This is an instance of a one-shot learning, binary image classification problem. The chosen language is Python, and the primary library we develop our network on is PyTorch.

2 Data

For our datasets, we have decided to use the CelebA dataset as the primary dataset for the training and validation phases of our model. This is because the "Learning Faces in the Wild" (LFW) dataset, which we originally proposed to constitute our entire dataset, is typically used as the de facto testing dataset for large-scale deep learning networks. By training on CelebA and testing on LFW, we have the advantages of (1) being able to directly compare our model's performance to that of other state of the art models and (2) obtain better insight on the robustness and learning of our model, as it will be tested on faces it has never seen before (and we can be confident the model has learned similarities, rather than memorizing features). LFW is a popular dataset used for image classification and facial recognition problems which combines several advantages – (1) it contains standardized, 250x250x3 images of many people (13,233 total), and several images for many of these people face (1680 people with k >= 2); (2) it is easy to access and download; (3) it is easy to parse, shuffle, and label to generate pairs or triplets of matching or unmatching faces; and (4) it is the exact dataset used for training, verification, and testing by both Taigman et al. and Schroff et al. in their facial verification projects, in which they used Siamese neural networks to approach the problem (see Baseline and

Relevant Papers section). The authors of the dataset also provide many helpful resources relating to using the faces for a model, so this will help us tremendously during the training/validation/testing phases of the project (Huang et al. 2007). Ultimately, the purpose of LFW is to help with the development of facial verification, which aligns with our goals in consideration to how we want to develop our network.

The CelebA dataset is a large-scale face attributes dataset with upwards of 200,000 celebrity images, each with their own 40 unique attribute annotations. The images in this dataset cover various different large pose variations and include images with varying background clutter. Other advantages of CelebA include a very large diversity of identities, with celebrities from all over the world (not just the United States), large quantities of images with rich labels, including up to 10,177 identities, 202,599 total face images, from more than 5 landmark locations, with about 40 binary attributes annotations per image. The dataset was meant to be employed for the training and test sets for the computer vision tasks such as face attribute recognition, face detection, and landmark (or facial part) localization.

A potential downside of both datasets for our network is that there are very few babies, adolescents, or elderly (over the age of 75) identities included. A majority of the people in this dataset are celebrities, or, in some form, well-known to an audience. This may mean our model may be biased based on age and ineffective at classifying people in these age-ranges. Despite this, we speculate that the narrower age range should make training easier, since babies and the elderly look drastically different from middle-aged people in terms of their facial structures. We also encountered some issues with making the images compatible with each other and the model. First, images from the CelebA dataset are of size 178x218; these are not compatible with the 250x250 dimensions from images of LFW. To resolve this, we resized the CelebA images to 250x250 by augmenting the original images with black borders (in general, we decided that upscaling the smaller images is superior to downscaling the larger images, for preserving information). Another issue we detected when running our model is that some CelebA images are in grayscale (1x250x250) instead of RGB (3x250x250). This conflicts with the channel dimensions of our convolution operations and was dealt with by resizing the grayscale images to RGB (there were approximately 10 images with this property). These are just a few downsides that come with our dataset that our team must consider in analysis. Overall, the CelebA and LFW dataset provide us with a very large number of standardized face images, even though they are only of celebrities, to apply our model on

3 Related Work

To approach this same problem (for k=2), Taigman et al. applied a Siamese network that was trained with standard cross entropy loss and error backpropagation to "directly predict whether the two input images belong to the same person." In this network, the absolute difference between the features was calculated, followed by a fully connected layer mapping these differences into a single logistic unit. They showed that this was able to achieve approximately 96.17% classification accuracy on the Learning Faces in the Wild (LFW) dataset. Considering human accuracy of approximately 97.53%, this was a revolutionizing study and at the time represented a staggering 27% reduction in error over previous state-of-the-art models.

The most crucial paper which guided us through creating our model, designing our training, validation, and testing phases, and implementing our network was from Google deep learning researchers, in a state of the art project called FaceNet. In FaceNet, Schroff et al. attempted facial verification for k=2 with a deep convolutional network that "learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity" (Schroff et al. 2015). On the same widely used Labeled Faces in the Wild dataset that we are using, they achieved a staggering prediction accuracy of 99.63%, outperforming both Taigman et al.'s accuracy (96.16%) and even human accuracy (97.53%) for the same problem. We referred to FaceNet because they achieved a similar goal to our project in their paper while also using the LFW dataset. Because of their success and extraordinary accuracy, much of our model is drawn from their design decisions and high level pipeline; however, we decided against trying to load their model and network weights or perfectly mimic their architecture, due to both the complexity to implement and high total resources required for training our model (with the limited GPU computation that we

have). One major takeaway from the paper is to train our model using triplets of images, consisting of an (anchor, positive, negative), where the anchor is an image of an individual's face, positive is a separate image of the same individual's face, and negative is a third image of a different individual's face. Triplets of this nature are superior to other known inputs because they can be trained with triplet loss, where the network can directly learn how to give similar embeddings for similar images and more distant embeddings for very different images. Another important takeaway from the paper is to train using "hard" triplets, where a hard triplet is defined as a triplet where the distance between (anchor, negative) is less than the distance between (anchor, positive). To obtain this, perform "online triplet mining" – in each minibatch during stochastic gradient descent, choose the p hardest triplets, which is defined as the triplets where d(a,n) - d(a,p) is highest; to obtain batch loss, take the average of only these triplets. The final takeaways from this paper establish a rough starting point for some of our hyperparameters – FaceNet concluded that the ideal threshold t for binary classification is 1.242 and the ideal margin, which is an additional parameter used in classifying "hardness" of triplets, is 0.2.

4 Approach

The pipeline which outlines our approach can be best detailed as follows:

1. Data Processing:
   (a) Load the CelebA and Labelled Faces in the Wild (LFW) dataset.
   (b) Create PyTorch datasets for both CelebA and LFW in order to pass into the DataLoaders. These datasets receive triplets of images, satisfying our (anchor, positive, negative) constraints.
   (c) Divide the CelebA dataset into training, validation, and testing portions of size 1000, 100, 100, respectively. Set aside a subset of the LFW dataset for testing, of size 125 (arbitrary).
   (d) Create DataLoaders for train, validation, and test data.
   (e) Develop Siamese Neural Network:
      i. Define hyperparameters – learning rate=0.001, batch size=8, number of epochs=50, triplet loss margin of 0.2, and threshold=55 (if our distances fall below this threshold, evaluate true; else, false).
      ii. Define model architecture – a Siamese network, consisting of two parallel convolutional networks with tied-together weights (which are updated synchronously). Exact network details can be seen in the model, but, to provide brief overview, each convnet consists of four layers, with the first two using (5x5) kernels and last two being (3x3) kernels, with 2-dimensional batch normalization and MaxPooling, a Rectified Linear Unit (ReLU) as the activation function at the end of each layer, and then a final fully connected layer which compressed the results of the convolution operations to a 1024-dimensional feature vector (embedding), which is the output of the network.
   (f) Train Model:
      i. Forward each batch of triplets through the model to obtain their embeddings.
      ii. Compute loss of the batch by feeding the embeddings into the selected criterion (torch.nn.TripleMarginWithDistanceLoss).
      iii. To implement online hard triplet mining, identify the three "hardest" triplets from the batch, which are the three triplets with the highest distance between (a,p) and (a, n); take the mean of these three losses and declare this the final loss of the mini-batch.
      iv. Propagate gradients of this mean loss backwards to train the weights, which, because of the Siamese network, will update along both convnets the same.
      v. Repeat (i)-(iv) for the specified number of epochs (50).
   (g) Run validation loops as follows:
      i. Obtain a batch of triplets from the model. To implement our base facial verification problem, we must randomly sample two (k=2) images from each triplet and assign a label based on whether the sample is a match or mismatch. To generate this sample,

choose half the input images to be (anchor, positive) with labels (1) for matching, and the other half to be (anchor, negative) with labels (0) for mismatching.

ii. Feed these pairs forward through the network to obtain their embeddings.

iii. Compute the pairwise distances between these embeddings. If the distance is below the threshold, then the model classifies the images are similar enough and are therefore a match. If the distance is above or equal to the threshold, the model classifies the images as too different, and are therefore a mismatch.

iv. Finally, compare this classification to the label; if the classification equals the label, mark it as a correct classification. Track validation accuracy as the number of correct classifications divided by the number of total classifications.

v. Repeat (i)-(iv) for the specified number of epochs (50).

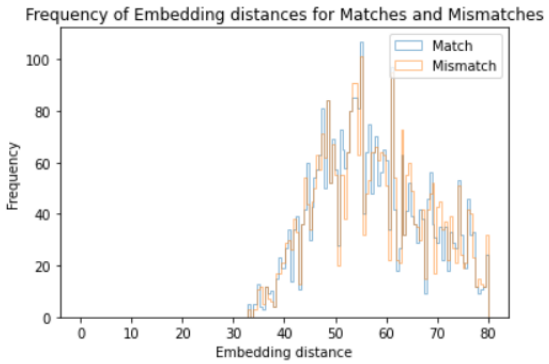
2. Visualize Results:
   (a) Plot loss curves.
   (b) Plot accuracy on both CelebA and LFW datasets.

Having hard triplets can place us in a local minima early on in training while soft triplets don't really benefit our model in any way. The point of using triplet loss is to make sure that the spacial distance between the anchor and the positive embedding minus the spacial distance between the anchor and the negative embedding is less than a defined threshold (FaceNet uses a threshold of 1.242, which we also tried using while running our model). FaceNet provided a lot of framework for us implementing this project, while also guiding us with techniques, such as semi-hard triplet mining, and structural design tips for our model, such as using a convolutional siamese network.

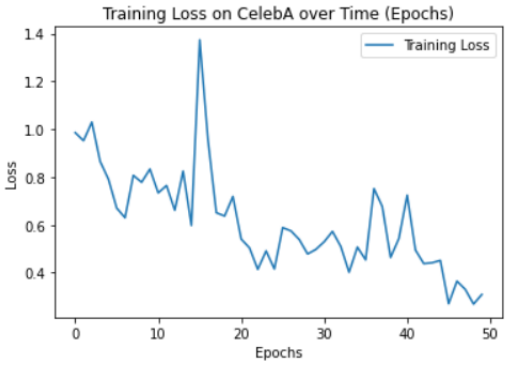
5 Experiments and Results

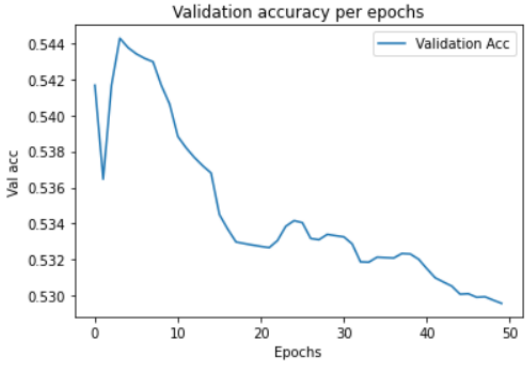
Our model certainly pales in comparison to FaceNet and other superior models, as the state of the art networks were both more architecturally complex and trained on much larger datasets – with Google's computational resources and ubiquitous access to images, they likely trained on datasets 1000x or 10000x larger than the datasets we trained on. However, we did manage to obtain some results, after running for approximately half an hour (50 epochs).

Before analyzing the results, we show our methodology for obtaining our final threshold value. As we mentioned before, FaceNet used a threshold of 1.242 when judging the distance between the anchors with the positives and negatives for the triplets. Unfortunately, we were not able to train or augment our model as extensive as they did, so we could not produce the level of embeddings they could in their paper, forcing us to settle on a less precise threshold of 55. To identify a good threshold, we graphed the frequencies of embedding distances for matches and mismatches, seen below. This allowed us to judge what the threshold size should be – a threshold that has the highest frequency splits the embeddings as best as possible (with limitations of small sample size and high variance). We can see that the highest frequency of embedding distances between matches and mismatches is in the range of 50-60, with an overall range of about 30 to 80. By visualizing the embedding distances of matches and mismatches of pairs of images, we can use a more informed threshold value to help our model form predictions based on the images from CelebA and LFW dataset.



To visualize our results, we plot the loss curve and validation accuracy over time:





It is important to note that validation loss should be included in the loss graph. However, due to GPU memory limitations, we could not compute validation loss during training while also computing validation accuracy.

After running inference on our test datasets for both CelebA and LFW, we obtained these final accuracies:

```
CELEBA TEST ACCURACY:
[0.5625]
LFW TEST ACCURACY:
[0.512]
```

The results are not very promising. There were many limitations to our training, which we will discuss below. Nonetheless, we managed to obtain an accuracy above 50% (default binary prediction accuracy), especially on the CelebA test dataset, which had a final accuracy of 56.25%. It makes sense that inference was more accurate when performed on the CelebA images, since it was the dataset we trained the model on. On the other hand, we were only able to obtain a 51.2% accuracy when running inference on the LFW test dataset, as it comes from a dataset that is completely different than the training set.

6 Conclusion and Discussion

To begin discussion, we will first detail some design decisions we faced as we developed our model. We first tried the FaceNet threshold of 1.242, but this was not effective and caused our model to yield only false classifications, as it was too low and rejected the distances between our embeddings. The likely explanation for this is that the feature vectors our siamese network produced were of a completely different feature space than that of the network produced by FaceNet. We also considered grayscaling our images, from 3x250x250, to 1x250x250, leading to a theoretical 3x speedup in the training, validation, and testing portions; we decided against this, as running for a reasonable number of epochs on the normal RGB data proved feasible and within the capabilities of Google Colab. This almost certainly helped our model, as grayscaling images causes information to be lost and directly leads to the network having greater difficulty identifying patterns and learning.

We encountered multiple issues when trying to develop and run this model. GPU limitations prevented us from using Colab to track validation loss. We also wanted to run our model for much longer (FaceNet was trained for over 1000 hours; for comparison, our model trained 30 minutes). We initially modelled our architecture off of standard Siamese networks we've seen (see Facial Verification.ipynb), but had to make modifications to convolutional filter dimensions, padding, and batch normalization operations to make the network compatible with our images. Aside from these issues, making our data compatible with the model (as discussed in the Data section), finding a good threshold for our network (as discussed in the Results section), and mining for "hard" triplets all constitute some of the problems we had to solve in developing this model.

Why are our results not good enough? One reason may be that our network architecture is not as good as we thought it was. We could have done more experimentation and attempted different designs for our model that we did not have enough time for. Furthermore, our current method for online triplet mining (taking the three hardest in the mini-batch) may likely have issues creating insufficent gradients, as the top 3 triplets out of 8 in the mini-batch may not be "hard" enough. One potential design change would be to make our model deeper, which would probably result in a more accurate feature-space mapping of input images. We could have also experimented with different hyperparameters, especially the threshold and margin (more on this in Future Work). Our LFW test accuracy is worse than test accuracy on CelebA's because the CelebA training images were augmented with black pixels to resize from 178x218 to 250x250; the model was not trained to handle the pixels in LFW in between this discrepancy from upscaling. Lastly, according to the graph of validation accuracy, the accuracy goes down over epochs; we think that our network is underfitting, and that significant training needs to be done before we can see a consistent increase in validation accuracy.

7 Future Work

With extra time, there are many different changes we can make to improve the classification accuracy in our model. With two extra weeks, the first think we would do is train the model for longer and over more data; half an hour (50 epochs) is certainly not enough time for the network to learn to map input images to feature vectors. To improve the model, we can experiment with different hyperparameters, Siamese network architectures, train on more datasets, and apply image augmentation, building robustness in the models. To obtain ideal (or near-ideal) hyperparameters, we could implement grid search, optimizing for learning rate, batch size, epochs, loss margin, and the threshold. With two weeks, we could also generalize the network to work for k>2; to do this, the model will compare every pair of images in the set, returning false if any pair exceeds the threshold and true otherwise; this change does not even require modifying or retraining the model, but instead needs restructuring of the validation and testing phases of the model.

With two extra months, we could look to experiment with deeper convolutional architectures in the Siamese network. By making our model more complex (with more layers, transformations, and overall steps) we could try to achieve a higher classification accuracy. We also can consider mining for semi-hard triplets, instead of hard triplets, where a semi-hard triplet is defined as $[0 < \|f(x_i^a, x_i^p)\|_2^2 - \|f(x_i^a, x_p^i)\|_2^2 < \alpha]$. According to the FaceNet paper, "selecting the hardest negatives can in practice lead to bad local minima early on in training, specifically it can result in a collapsed model (i.e. f(x) = 0)" (Shroff 2015). An alternative triplet mining procedure to consider would be the "offline triplet generation" method, also outlined in the FaceNet paper, which computes hard triplets every "checkpoint," which occurs less frequently than every mini-batch. It would be interesting to compare results between offline and online triplet mining.

One final application for potential future work would be to implement the FaceNet architecture or download their exact model (and subsequent weights/biases). Representing current state-of-the-art performance accuracy on this problem (99.63%), being able to modify their network and experiment with techniques like dropout or learning rate decay could potentially improve the success of their model. By taking note of these prospective improvements, we can improve our model, achieving better results and increased accuracy.

8 Link to Final Presentation

https://docs.google.com/presentation/d/1KYd3I4pU3F1Unc7-QRprWS7JMMVhQFW5jqZT88ygz14/edit?usp=sharing

References

[1] Computer Vision Machine Learning Team. (2017, November) An on-device deep neural network for face detection. Retrieved March 31, 2021, from
https://machinelearning.apple.com/research/face-detection

[2] Huang, G., Ramesh M., Berg T., & Beeman, D. (1995) Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. *University of Massachusetts, Amherst,* Technical Report 07-49, http://vis-www.cs.umass.edu/lfw/

[3] Schroff, F., Kalenichenko, D. & Philbin, J., 2015. Facenet: A unified embedding for face recognition and clustering. In Proceedings of the IEEE conference on computer vision and pattern recognition.

[4] Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification", *Conference on Computer Vision and Pattern Recognition*, 2014. 2.