Face Recognition in Deep Learning with Minimal Resources

Annika Basch abasch3@gatech.edu
Stephen Lyons slyons39@gatech.edu
Khaled Abdelaziz kabdelaziz3@gatech.edu
Michelle Adea madea3@gatech.edu

Georgia Institute of Technology

Abstract

Today's facial recognition models can achieve accuracies over 99% with deep learning models that contain over 100 million parameters. In this paper, we explore the performance of three current deep learning models used for facial recognition: Inception ResNet v3, MobiFace, and DeepFace. We investigate training the models with a smaller dataset and reduced sizes. By analyzing the performance of these models with minimal resources, we observe how facial recognition may be accomplished in environments with limited computational resources, such as mobile environments. Ultimately, we are unable to implement a more lightweight model. We do leverage transfer learning from object recognition to facial recognition and observe models with reduced sizes are able to learn on smaller datasets.

1. Introduction

Our objective is to replicate the success of three facial recognition models, Inception ResNet v3, MobiFace, and DeepFace, using a smaller dataset and reduced model sizes to solve the problem of facial recognition models being inconveniently large. Efficient models are vital for today's mobile-first software because mobile software has tight constraints on available computational resources. Currently, standard facial recognition models are too large to fit onto mobile devices.

In the paper "Deep Face Recognition: A Survey," the authors state that deep learning "dramatically boosted [the accuracy] to above 99.80%" on face recognition since 2014 [12]. Most state-of-the-art models have significantly over 100 million parameters and are computationally expensive. One relatively efficient model that achieves comparable accuracy is Inception ResNet v3, which has around 25 million parameters [9]. One standout model is MobiFace [2], which achieves over 90% accuracy on the "Labeled Faces in the Wild" benchmark with fewer than 4 million parameters [4].

Image classification models are generally not run on mobile devices today, but rather on the cloud or on larger devices with GPUs. Google Photos and Apple Photos both leverage facial recognition software to allow users to search through their photo libraries by person on their mobile devices. However, one major limitation of the technology is that the photos must be cloud-based. Even under ideal conditions with a fast internet connection, it can take hours for faces to be identified in a new photograph and grouped by the same face or direct label. Additionally, if users prefer to store their photos locally and not sync them to the cloud, they may not be able to leverage the technology.

The current state-of-the-art facial recognition model, DeepFace, which was developed by Facebook, has achieved impressive results in face recognition tasks with over 120 million parameters. This model is computationally expensive and requires a vast amount of storage space. Note, DeepFace has also been criticized for its lack of transparency and potential privacy concerns [11].

Another approach to optimize facial recognition models involves using embeddings instead of traditional classification layers. Embeddings can represent complex relationships between identities in a more compact form. This can be particularly effective when there are only a few examples of a class. Embeddings can simply be the output of a linear layer. Another benefit to this is the compact size of the embedding could be easily uploaded, since it would be much smaller then the image itself, from the mobile device to a database making it easy to reference on other datasets.

To address the constraints on time and computing resources, we will explore two approaches. First, we will explore using a pretrained model like Inception ResNet v3 with pretrained weights on the ImageNet dataset. This model has already been trained on a large dataset for the task of object recognition and can be fine-tuned further for specific tasks with fewer parameters. This can significantly reduce the computational requirements and storage space needed to train a model from scratch. Second, we will explore implementing a more efficient model like MobiFace. MobiFace is a lightweight architecture that has fewer than 4 million parameters and can run on mobile devices with limited com-

putational resources [2]. While MobiFace may not achieve the same level of accuracy as DeepFace, it may be sufficient for certain applications where efficiency is a priority.

If we succeed in creating compact, well-performing facial recognition models on a smaller dataset, it will provide a proof-of-concept for running models locally on mobile devices and personal computers. This would enable users to classify pictures without uploading them to cloud or waiting hours for cloud-based classification.

The optimized model will have additional applications in security, law enforcement, and e-commerce. In home security, for example, smart doorbells could identify visitors and notify the residents who is visiting. Also, home security cameras could reduce notifications for people it recognizes as inhabitants.

We use the CelebA (CelebFaces Attributes) dataset to derive our data, specifically the aligned dataset that contains processed images that align the individual's face in the photo [6]. It contains a total of 202,599 images with 10,177 individual identities/labels. This dataset was created by the Multimedia Lab (MMLAB) of the Chinese University of Hong Kong to specifically solve the following tasks in computer vision: face detection, landmark (facial part) localization), face editing and synthesis, face attribute recognition, and face recognition [6]. We use a subset of the CelebA dataset with a total of 20,000 images. We use 15,000 images for training and 5,000 for testing with about 7,000 individual identities/labels. This means there are approximately 2 to 3 samples per identity/label.

We hypothesize that results may be further improved with transfer learning, so we introduce a second dataset, CIFAR-10, for pretraining. We first train the models with the CIFAR-10 dataset to get pretrained weights for each of the models. We then freeze the feature weights and train the classification layer on the CelebA dataset. The CIFAR-10 dataset does not contain faces and is used for object detection rather than facial recognition. It contains a total of 60 thousand images with 10 classes; this means 6000 color images of size 32x32 per class [5].

We hypothesize that higher-quality data than CIFAR-10 for pretraining would even further improve the learning and performance of the models. In the case of Inception ResNet v3, pretrained weights from the ImageNet dataset are readily available. We download the pretrained weights for Inception ResNet v3 on the ImageNet dataset from the pretrained-models library [1]. The ImageNet dataset was created with the purpose of providing data to advance training for large-scale object recognition models. It contains 14,197,122 images with 21,841 synsets, and about 1000 images per synset. These synsets represent synonym sets, a representation of multiple words or word phrases, in WordNet, which is a database of semantic relationships between English words [8].

2. Approach

All three of the models are deep neural networks. We imported Inception ResNet v3 from PyTorch models. We implement MobiFace and DeepFace using PyTorch by following the guidance of the papers as they are not publicly available. We plan to implement these deep neural networks to solve the facial recognition task with a sparse dataset, meaning fewer examples per batch. We think this approach will be successful because of the previous research and high performance of these models. Something new, or more of a continuing effort in this field, is that we will attempt to make these models more lightweight to improve upon constraints in time and computing resources. The details of our plans to solve this problem are provided below.

Inception ResNet v3 consists of a combination of convolution, average pooling, max pooling, concatenation, dropout and fully connected layers with batch normalization. The concatenation within the inception modules allows for parallelization, meaning multiple feature maps can be created with various filter sizes and concatenated to produce a singular output [10].

MobiFace is a lightweight implementation consisting of convolution and depthwise convolution layers, alternating bottleneck and residual bottleneck blocks, a convolution layer, and fully connected layer. It uses batch normalization and non-linear activations with PReLU, which is a modification of ReLU with a slope for negative values. This architecture facilitates fast downsampling. MobiFace seeks to maximize the information in the final feature embedding while keeping the computational cost to a minimum [2]. In our implementation, we reduce the number of parameters and number of layers by implementing only one instance of each layer rather than repeating layers multiple times. We also eliminate skip connections to decrease complexity and memory requirements.

DeepFace is a Facebook proprietary model which is not publicly available. We recreated DeepFace's nine layer deep neural network based on Taigman et al.'s paper that outlined its architecture, consisting of the alignment pipeline layer, a convolutional layer, a max-pooling layer, and another convolutional layer, followed by three locally connected layers and two fully connected layers [11]. We are using aligned input images from the CelebA dataset, so our DeepFace architecture does not have the alignment pipeline layer and consists of the remaining eight layers.

All the models use the stochastic gradient descent (SGD) optimizer. SGD is historically a good optimizer that may take many steps to update and converges to the optimal weights of the model. Hyperparameters include epochs, batch size, learning rate, and momentum. We use a batch size of 128 since it is the largest size that will fit onto VRAM. As there are fewer examples per class, a higher learning rate is desirable so that the model can learn as much as possible

from each example. However, too high of a value may cause instability as the model will overfit to the examples, and it may cause the model to converge too quickly to a suboptimal solution. We choose a learning rate of 0.3, which is as high as we can go before noticing a loss of stability in training. We choose a momentum of 0.9 without tuning as it is a standard value. Finally, we train for 500 epochs with early stopping if a high accuracy is achieved or no improvements are observed.

For more information on the architecture behind the models you may find additional details here:

- Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning [9]
- MobiFace: A Lightweight Deep Learning Face Recognition on Mobile Devices [2]l
- DeepFace: Closing the Gap to Human-Level Performance in Face Verification [11]

We anticipated problems with maintaining a consistent computing environment to train and test the models. Our teammate Stephen volunteered to use his personal NVIDIA RTX 3070 GPU, so we have him run training and testing to ensure consistent and comparable results. We implement the architecture for our models in Python with the PyTorch library and use GitHub to store and update our code [3]. For more details on additional Python packages, please refer to the file requirements.txt in the Supplemental Material.

One issue we run into using the NVIDIA RTX 3070 GPU is that it cannot fit the entire MobiFace model for training with the CelebA dataset. The implementation of MobiFace in the paper has an order of magnitude fewer parameters 2. This is because the creators of MobiFace were able to leverage embeddings for the classification layer [2], whereas we have to rely on a fully connected layer to give us probabilities for each class. We make an attempt to implement embeddings, but due to time and resource constraints, we are unable to produce useful results, leading to decreased learning compared to the results without embeddings. By pretraining MobiFace on CIFAR-10 and then freezing the feature layers, we are able to fit MobiFace for training the classification layer with the CelebA dataset.

Our target benchmark is at least 90% accuracy in all models on the CelebA dataset. Originally, we thought we would only train the three models on the smaller subset of the aligned CelebA dataset. However, due to limitations in computing resources, we are not able to train the entire MobiFace model with the CelebA dataset. In addition, we hypothesize that pretraining may improve results. Revisiting how the models were originally trained, we notice they had many more images per label. So we investigate pretraining the models on the CIFAR-10 dataset. Note the Inception ResNet v3 model has existing pretrained weights from ImageNet via the pretrainedmodels PyTorch library, which we investigate for use in pretraining as well.

3. Experiments and Results

In our study, we aim to train a facial recognition model using a dataset of 20,000 images with 7,000 classifications. Our dataset is split into 15,000 images for training and 5,000 for testing, with only two samples on average per training loop, making learning slow and difficult. We target a training loop of 500 epochs with early stopping if high accuracy is achieved or if no learning is occurring. The loss and the accuracy on the test data are recorded for each epoch, and the weights are saved for potential future research. To evaluate the model's performance and to determine if the model overfits and generalizes well, we compare the training accuracy to the test accuracy over time. We do not see any issues with overfitting, and you can see in our test accuracy that it never had a net decrease 1.

We implement a replica of the Facebook DeepFace model using the PyTorch framework and train it on the CelebA dataset. While the training loss shows steady improvement, the testing accuracy does not noticeably improve. The original model does include locally connected layers that PyTorch does not have, which may account for the lack of success. We make various modifications to the architecture, like substituting PyTorch's local response norm modules to build a custom locally connected module, but these changes do not lead to any improvements in the testing accuracy. Additionally, we tune hyperparameters such as the learning rate and batch size, with an increase in the learning rate resulting in an improvement in the training accuracy but not the testing accuracy. We are unable to observe the DeepFace model learning effectively. We hypothesize the following reasons for why it performs poorly. One, the DeepFace architecture is rather large and we need to train the model from scratch. This training requires a significantly larger number of samples than we are able to provide as input. Two, we may have benefited from scheduling the learning rate with something like Reduce-On-Plateau. Lastly, the original DeepFace model implementation relied on preprocessed data from its alignment pipeline layer and the aligned data from the CelebA dataset may not be adequate to be evaluated by the model.

Initially, the Inception ResNet v3 model does not learn at all when starting with randomly initialized weights and using only the CelebA dataset. To address this issue, we decide to perform pretraining on the CIFAR-10 dataset, using its feature layers and freezing them before training on CelebA. This approach enables the models to start learning on the CelebA dataset. However, the learning is slow with the CIFAR-10 training set due to its low resolution (32x32) and lack of facial features as classification tasks. Therefore,

Model	Parameters
Inception v3 without classification layer	27 Million
Inception v3 with classification layer	40.1 Million
MobiFace without classification layer	0.77 Million
MobiFace with classification layer	738 Million
DeepFace without classification layer	127 Million
DeepFace with classification layer	157 Million

Table 1. Number of Parameters per Model

the features learned are likely not very complex. We then decide to try and learn the CelebA dataset on the Inception ResNet v3 model from scratch again but allow it to train for many epochs, so instead of 50 we allow it to train to 300. 50 was chosen initially based on the reference model, Inception ResNet v3 with ImageNet pretraining, which only took 7 epochs to achieve nearly 100% accuracy. The high performance of Inception ResNet v3 with ImageNet pretraining is expected due to the high-quality and large size of the ImageNet data as well as the first-rate weights available for download. What we see in the Inception ResNet v3 model without pretraining is that the model does eventually start to learn and we achieve a better learning rate by epoch 60 and by epoch 90 it is at 100% accuracy on the test data 1. We can see that the pretraining gave the model an initial boost compared to starting from random initialization, demonstrating that pretraining is an effective method of weight initialization. The higher accuracy eventually achieved on the Inception ResNet v3 model when using the CelebA dataset compared to pretraining with CIFAR-10 is most likely due to freezing the feature layers during the transfer learning. It would have likely been better to not freeze those layers to allow them to learn the more complex features that are in facial features.

MobiFace is able to learn, but not as effectively as Inception ResNet v3. MobiFace fails to achieve greater than around 20% accuracy 1. This is likely due to the large size of the classification layer making training slow and difficult, plus the frozen feature layers being unable to learn. We hypothesize that using effective embeddings and training without frozen feature layers would significantly improve MobiFace performance, and this is left as a direction of future research. Using effective embeddings would allow us to significantly reduce the number of parameters, making learning easier and faster. Allowing updates to the feature layers would allow the model to improve and tune its learned features to the target dataset rather than relying on the weights from pretraining.

There is potential to perform pretraining on even small datasets to initialize the weights of a new model. This serves two purposes. One is to have a comparison on a standard dataset in the event your application has custom data. You can then compare it to other types of models. Secondly, you can save this model after pretraining and use it as a starting point for multiple unique problems. It also shows how pretraining on a model and then re-training on new tasks works well, such in the case of ImageNet with the inception model. Having these weights available dramatically decreases the time to achieve high accuracy on new tasks. This shows how feature layers, or convolutional layers, can generalize well in image tasks.

We also explore the idea of using embeddings instead of a hard-coded linear layer with an output for each unique face. By leveraging the robust feature set from ImageNet training, we anticipated more accurate results in classifications with few examples and other benefits such as reduced model size and faster inference. However, implementing embeddings with the Inception model and ImageNet only achieves a 15% accuracy rate. This is not accurate enough and lower than the accuracy achieved with CIFAR-10 dataset transfer learning. Further testing should be done as embeddings is the standard approach for facial recognition. This is due to some of the negative aspects of a linear layer for classification, such as having to retrain for each new face label, and how large the linear layer can become. MobiFace, although initially the smallest model, has a large output set of features and quickly grew and became too resource intensive due to relying on a linear layer rather than embeddings.

Our experiments indicate that learning complex features with low sample count and high label count does not work well. We are able to find success, however, in separating feature learning from facial classification learning. Additionally, we showed that transfer learning works even on unrelated image categories. The CIFAR-10 dataset has only ten categories and does not include people.

4. Conclusion

In this report, three deep neural networks - MobiFace, Inception ResNet v3, and DeepFace - are implemented using PyTorch. DeepFace is a standard, larger model. Inception ResNet v3 is smaller than DeepFace, and MobiFace is the most lightweight model with a minimal computational cost. All three models are trained using the Stochastic Gradient Descent optimizer with momentum and tuned hyperparameters.

During the experiments, the DeepFace model shows poor

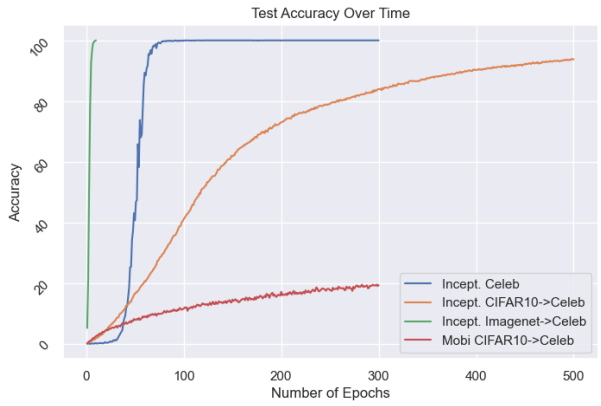

Figure 1. Training results

learning even after several modifications are made to the architecture and hyperparameters are tuned due to its size, and lack of scheduled learning rate and lack of highly preprocessed data. The Inception ResNet v3 model initially does not seem to learn when trained on the CelebA dataset only, but the initial learning performance improves after about 50 epochs and increases rapidly, outperforming CIFAR-10 pretraining. We hypothesize that the pretrained model on CIFAR-10 would perform at least as well as the model without pretraining if the feature layers were not frozen during training. Unsurprisingly, the Inception ResNet v3 with ImageNet pretraining performed the best achieving nearly 100% accuracy in only 7 epochs due to high quality of the ImageNet data and the weights available for download. MobiFace is too large with the linear classification layer to train on available computational resources without our pretraining method. MobiFace is able to learn, but not as effectively as Inception ResNet v3. We hypothesize that using effective embeddings and training without frozen feature layers would improve MobiFace performance, and this is left as a direction of future research.

Pretraining can be very useful when dealing with a sparse or unlabeled dataset. It can reduce the number of epochs needed to train on your custom dataset and reduce the chance of no learning occurring. Furthermore, the idea of using embeddings instead of a hard-coded linear layer was explored, but it did not produce satisfactory results.

The experiments showed that learning complex features with both a low sample count and high label count does not work well. Instead, separating feature learning from facial classification learning and using transfer learning from unrelated image categories can lead to improved results.

In the end, we were unable to implement a more lightweight model. We did accomplish a high accuracy with the Inception ResNet v3 model pretrained on ImageNet to evaluate the small CelebA dataset.

There are many areas of potential work in the future. First, we can explore using embeddings. Towards the end of the project, we attempted to use embeddings to map the images to vectors. We had limited resources and time to produce embeddings and were unsuccessful in producing satisfactory results. We hypothesize that effective embeddings will produce improved accuracies in a faster amount of time. Second, we could perform transfer learning without freezing the feature layers. This should improve the learning and performance of the models. Third, we could use video data instead of static image data in our models. In the future, it will be interesting to explore processing video data as input from the YouTube Faces Database (YTF) [13]. Fourth, we can explore preprocessing image data by cropping and aligning the images to specifically contain the face, while only removing noisy backgrounds and extra information. We hypothesize that the increased focus on the target area would improve the accuracy of face recognition. Fifth, the transformer architecture has advanced performance in the domain of natural language processing. We can specifically explore how using this architecture in Vision Transformers will preserve overall global and local specialization of the original features with use of skip connections and attention mechanisms [7]. This could allow for faster facial recognition.

5. Work Division

The individual responsibilities of each team member are outlined in Table 2.

Student Name	Contributed Aspects	Details
Annika Basch	MobiFace Model	Wrote the code for the MobiFace model. Debugged the model with Michelle. Transferred the draft to LaTeX Overleaf template with Michelle.
Stephen Lyons	Utility functions (training/testing loop, data loading)	Created the training and testing loop code and supporting utility functions, such as data loading, pre-processing, and saving metrics.
Khaled Abdelaziz	DeepFace recreation	Created an approximation for DeepFace model and attempted to train it from scratch. Collaborated with the team on the project proposal and final report.
Michelle Adea	MobiFace, DeepFace Models	Debugged MobiFace with Annika. Worked on implementing DeepFace with Khaled. Transferred the draft to LaTeX Overleaf template with Annika.
All Team Members	Planning, Meeting, Writing	Contributed to project scoping, met on a weekly basis, wrote up and edited proposal and final report and troubleshot training/testing issues during training of the models.

Table 2. Contributions of team members.

References

[1] Remi Cadene. Pytorch pretrained library. https://github.com/Cadene/pretrained-models.pytorch/tree/master/pretrainedmodels/models. 2

[2] Chi Nhan Duong, Kha Gia Quach, Ngan Le, Nghia Nguyen, and Khoa Luu. Mobiface: A lightweight deep learning face recognition on mobile devices. *CoRR*, abs/1811.11080, 2018. 1, 2, 3

[3] The Linux Foundation. Pytorch documentation. https://pytorch.org/docs/stable/index.html. 3

[4] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007. http://vis-www.cs.umass.edu/lfw/. 1

[5] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). http://www.cs.toronto.edu/~kriz/cifar.html. 2

[6] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015. https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html. 2

[7] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *CoRR*, abs/2108.08810, 2021. 5

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. https://www.image-net.org/. 2

[9] Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. 1, 3

[10] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. 2

[11] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014. 1, 2, 3

[12] Mei Wang and Weihong Deng. Deep face recognition: A survey. *CoRR*, abs/1804.06655, 2018. 1

[13] Lior Wolf, Tal Hassner, and Itay Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR 2011*, pages 529–534, 2011. 5