# CIS 5810 Extra Project Final Report – Celebrity Facial Recognition

Angela Wang (shuw119@seas.upenn.edu)

Zhengtao Hu (huzhengt@seas.upenn.edu)

Spring 2023

# Abstract

In this project, we aimed to develop a facial recognition algorithm capable of accurately detecting Brad Pitt's face from a dataset of videos and images. The algorithm is built based on a pre-trained VGG16 deep learning model, which we selected as the base model for this task, given its widespread adoption and effectiveness in facial recognition tasks. We fine-tuned the VGG16 model on a training dataset consisting of 5,000 Brad Pitt images and 10,000 non-Brad Pitt images using PyTorch, a powerful deep learning framework. Throughout the development process, we encountered challenges such as dataset collection and overfitting. We implemented various strategies, including video capturing using OpenCV and hyperparameter fine-tuning, to overcome these issues. As a result, the final model achieved a high level of accuracy (97%) in recognizing the target celebrity on the validation dataset, compared to xx% demonstrated by the benchmark model, highlighting the effectiveness of our model in celebrity face recognition tasks. This report provides an in-depth analysis of the project’s objectives, methodology, results, conclusion, and related works.

# Introduction

Facial recognition technologies have experienced rapid advancements across various industries over the past decade. In this extra project, our main goal is to develop an algorithm capable of accurately detecting a given celebrity’s face from sets of videos or images.

To achieve this goal, our objectives include:

* Assembling an impactful dataset for training, validation, and testing purposes.
* Evaluating pre-trained facial recognition models and choosing the base model
* Applying transfer learning to fine-tune the base model and developing a functional facial recognition algorithm capable of detecting faces in videos or images.
* Evaluating the algorithm's performance and comparing it with benchmark models.

Our approach to this problem sets up apart from others in several ways. First, our final algorithm is designed to detect human faces and recognize our target celebrity from both locally saved videos and Youtube live stream videos, showcasing its versatility. Second, we evaluated, selected, and leveraged existing pre-trained models, allowing us to effectively solve the task while building a comprehensive understanding of the most widely adopted models in the field. Third, we trained the model using various benchmark settings with different hyperparameters and dataset sizes, which enables us to identify the factors that have influence on the model’s performance and understand the model’s limitations.

# Related Works

Our celebrity face recognition project was built upon a variety of existing research papers and leveraged well-established tools and frameworks such as PyTorch and OpenCV.

1. The VGGFace model [1] is a deep learning-based face recognition model that achieved superior results on various benchmarks and is widely adopted for various facial recognition tasks. The VGGFace architecture is a deep Convolutional Neural Network (CNN) that uses multiple convolutional layers to learn various features for facial recognition. In our project, we adopted and fine-tuned the VGGFace16 model to effectively recognize the target celebrity (Brad Pitt) from images and videos.
2. FaceNet [2] is another well-known work in the field of facial recognition. This model uses a deep CNN to learn a unified embedding for face recognition and clustering. It leverages the concept of triplet loss to enforce a distance metric that allows for easy comparison of facial embeddings. Although we eventually chose to use the VGGFace model, we studied FaceNet and acquired insights from FaceNet to deepen our understanding of deep learning-based face recognition techniques.
3. Yin et al. [3] proposed a center-based feature transfer framework to address a common challenge in facial recognition that deals with under-represented subjects who have few samples. The framework augments the feature space of under-represented subjects by transferring information from regular subjects with sufficiently diverse samples. Although we eventually did not apply this approach, the paper provided valuable guidance for handling potential data imbalance in our celebrity dataset.
4. Bansal et al. [4] offered practical guidelines for designing and training CNN-based face recognition models. This paper highlights important architectural choices, training strategies, and loss functions that can significantly impact model performance. We referred to this work when training our model and fine-tuning the hyperparameters.

In terms of implementation, we used two primary tools and frameworks:

1. PyTorch [5] is a flexible and high-performance deep learning library that allowed us to build and train our VGGFace-based model efficiently. PyTorch provides a wide range of functions and modules, making it convenient to develop complex neural network architectures and experiment with different settings. Its support for GPU acceleration further allowed us to train models quickly.
2. OpenCV [6] is a comprehensive open-source library for computer vision and machine learning. It provides an extensive set of tools and algorithms for image and video processing. In our project, we relied on OpenCV for capturing batches of images of both Brad Pitt and non-Brad Pitt people from videos, which helped us solve the problem of dataset size. Also, we used OpenCV in the implementation of the algorithm so that our program can automatically detect human faces from videos, and then recognize if it’s the target celebrity’s face.

# Methodology

Our methodology for the facial recognition project consists of the following steps:

1. Data Collection and Preprocessing:

We chose Brad Pitt as our target celebrity for facial recognition due to the ease of collecting images. We initially collected 5000 Brad Pitt through web scrapping and video frame extraction techniques. After thorough data cleaning, the final collection of training dataset consisting of 3000 Brad Pitt images and 8500 non-Brad Pitt images. We allocated 9300 images as the training set and 2200 as the validation set. The images were mostly single-person shots, so we separated them into "brad" photos and "non-brad" photos. We preprocessed the training set images using various data augmentation techniques, such as random cropping, horizontal flipping, rotation, affine transformations, and color jittering to increase the model’s robustness. We used the “*torchvision”* library to apply these transformations.

1. Model Selection and Modification:

We selected the VGG16 model as our base model due to its proven performance and popularity in facial recognition tasks. The VGG16 model is built with a simple architecture with a series of convolutional layers, which enables the model to learn a hierarchical representation of features and makes it suitable for facial recognition problems. We utilized the “*torchvision.models*” library to load a pre-trained VGG16 model. To adapt the model for our binary classification problem, we modified the last layer of the classifier to have two output nodes corresponding to our two classes (“brad\_pitt” and “ppl”). We froze the early layers of the model, which detect basic features like edges, corners, and textures, to preserve the pre-trained weights. We added three fully connected layers followed by ReLU activation functions and a Softmax layer at the end for fine-tuning.

1. Training and Validation:

We set up the training and validation data loaders using the “*torch.utils.data.DataLoader*” class to load and process the training and validation datasets. Our final model uses the CrossEntropyLoss loss function and the Adam optimizer with a learning rate of 1e-4 and weight decay of 5e-4 for 100 epochs.

1. Model Evaluation and Visualization:

After training, we evaluated our model's performance with a batch of validation images. We visualized the images along with their predicted and true labels, highlighting correct predictions in green and incorrect predictions in red. We plotted the training and validation loss over the epochs to visualize the model's performance. If the performance turned out to be unsatisfactory, we would fine-tune the hyperparameters and repeat Step 3. To encounter the overfitting issues, we fine-tuned the hyperparameters and applied the L2 regularization. Once we were satisfied with the model’s performance, we used OpenCV library along with a pre-trained Haar Cascade classifier in the implementation of the algorithm to best visualize the results. Our program can automatically detect human faces from locally saved videos or Youtube live stream videos, and then recognize if it’s the target celebrity’s face.

# Experiments and Results

In this section, we present the various experiments performed throughout the process and their results for the facial recognition task. We evaluate the impact of design choices by comparing the visually observed results and the loss functions produced.

The design choices we made include:

1. Model Layer Structure

The first experiment involved modifying the VGG16 model's layer structure. We froze the early layers of VGG16, which detect basic features such as edges, corners, and textures, and only modified the last classifier layer for the facial recognition task. We tried adding additional fully connected layers to the original VGG16 architecture and also tried removing fully connected layers from it. The final model uses a modified VGG16 architecture with the following additional layers for fine-tuning:

1. A linear fully connected layer with 512 neurons
2. A linear fully connected layer with 256 neurons
3. A linear fully connected layer with neurons equal to the number of classes (2)

These layers, along with ReLU activation and Softmax functions, were appended to the VGG16 model's classifier in our final model.

1. Size of the Dataset

We experimented with different dataset sizes, initially using a smaller number of images (1000 for Brad Pitt, 4500 for non-Brad Pitt) for the training set. Later, we expanded the training set to include more images (5000 for Brad Pitt, 8500 for non-Brad Pitt) by extracting the video frame from Brad-related videos. To guarantee data integrity, we performed data cleaning. Finally, the data set has 3000 for Brad Pitt, 8500 for non-Brad Pitt, which expects to improve the model's performance after training.

1. Hyperparameters

We experimented with different hyperparameters, such as the number of training epochs. Initially, we set the number of epochs to 35, and then increased it to 100.

With different combinations of the above design choices, we selected three trained models to compare their performances:

1. Final model: our final model used the modified VGG16 architecture with 3 additional fully connected layers as shown above, a larger dataset size, and 100 epochs.
2. Alternative Model : this model has the same number of images and epochs as the final model but used a different layer structure. Instead of adding additional fully connected layers, this model removed the last 3 layers from the VGG16 classifier.
3. Benchmark 1(mid-term model): the model we built and submitted in the mid-term report used the VGG structure with only last layer modified, a smaller dataset size, and 35 epochs.

Table 1 summarizes the different choices utilized in each model.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Final Model | Alternative Model | Benchmark 1 |
| Model Layer Structure | VGG16 with 3 FC layers added to classifier | VGG16 with 3 FC layers removed from classifier | VGG16 with last FC layers modified |
| Size of Dataset | 3000 for Brad Pitt  8500 for non-Brad Pitt | 3000 for Brad Pitt  8500 for non-Brad Pitt | 1000 for Brad Pitt  4500 for non-Brad Pitt |
| Epochs | 100 | 100 | 35 |
| Learning Rate | 0.0001 | | |
| Loss Function | CrossEntropyLoss | | |
| Optimizer | Adam | | |

*Table 1. Design Choices of Models*

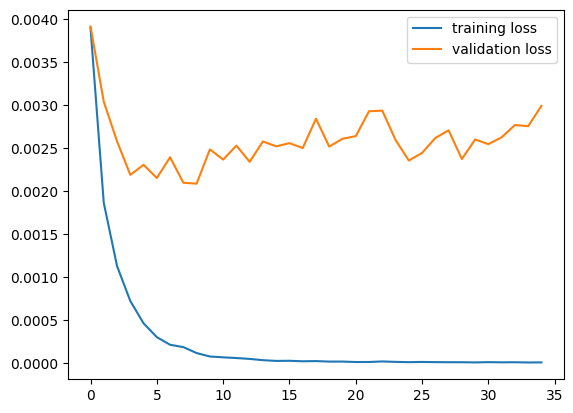
To compare the performances of the three models and analyze the optimization strategy, we started with the Benchmark 1 model, the one we built and submitted in the mid-term report. The main challenge our team faced at that time was that the training loss and validation loss did not seem to converge to the same minimal value. Specifically, the validation loss stopped decreasing and fluctuated around a certain value, even with an increased number of training epochs (refer to Figure 1). We quickly realized that a more comprehensive and diverse dataset might help the model learn more nuanced features and potentially increase the model’s accuracy.

Figure Training Loss and Validation Loss of Benchmark 1

Chart, histogram

Description automatically generatedTo reduce overfitting and improve the model performance, we expanded the size of the training dataset to 3000 Brad Pitt photos and 8500 non-Brad Pitt photos utilizing OpenCV’s video capturing functions. The improvement on the model performance was obvious. As shown in Figure 2, the training loss and validation loss was able to converge to a similar minimal value. Another change we made is to increase the number of epochs from 35 to 100, which extended the model training time. Through the visualization of the results (refer to Figure 3), although the algorithm was able to recognize most of the Brad Pitt photos correctly, there is still room for improvement.

Figure Training Loss and Validation Loss of Final Model

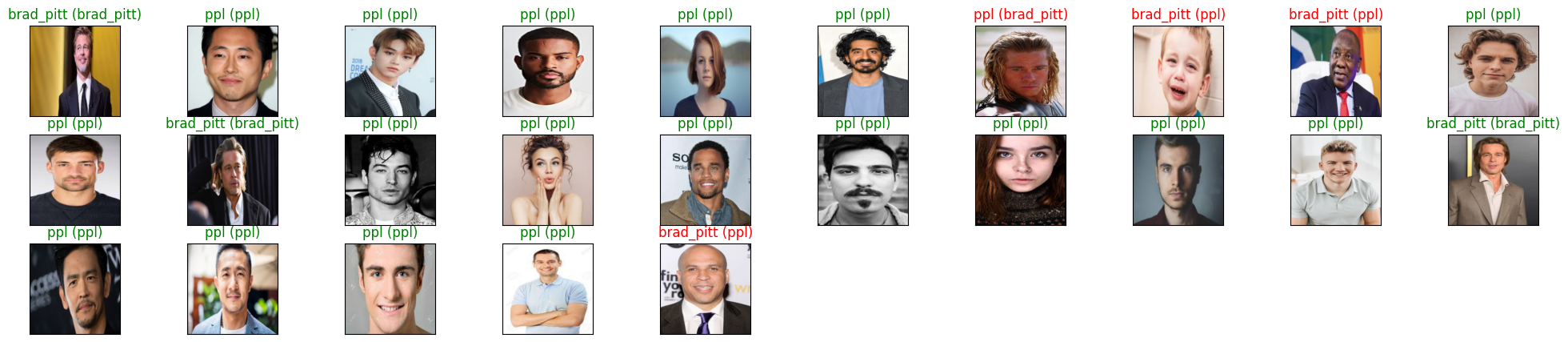


Figure Visualization Result of Final Model

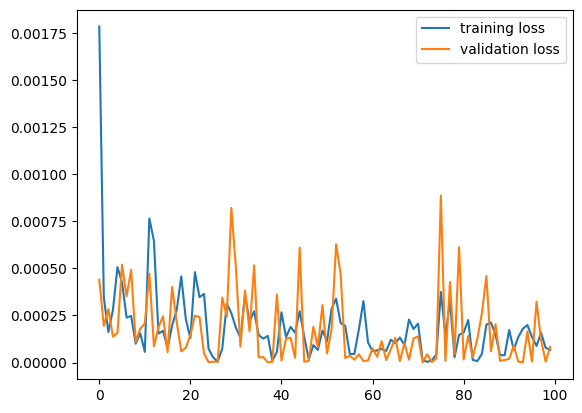
After having a functional model with converging loss functions, we kept on experimenting different model layer structures to find the optimal model. Alternative model utilizes a modified VGG16 structure with the last 3 fully connected layers removed from the classifier. We kept all the other parameters the same to see the impact. The results show that the modified layer structure produces a similar satisfactory result with our final model. The visualization results of Alternative model can be found in Figure 4.

Figure 4 Training Loss and Validation Loss of Alternative Model



Figure 5 Visualization Result of Alternative

# Conclusion

By comparing those three benchmarks, we demonstrated the impact of design choices, such as model architecture, dataset size, and hyperparameters, on the visual quality of the results. By expanding the dataset and modifying the architecture, our final model and alternative model improve significantly regarding overfitting issue, showcasing their ability to accurately recognize a celebrity's face from images and videos.

Our team has identified several areas for improvement that can potentially further enhance the accuracy and performance of our celebrity facial recognition algorithm.

1. Expanding the dataset: We will continue to expand our dataset by adding more images to both “brad” and “non-brad” groups with photos taken in various lighting conditions, poses, and expressions. Ideally, we would like to expand the size of the dataset to 10,000 for “brad” and 20,000 for “non-brad”.

2. Implementing a more sophisticated augmentation strategy: We will look into developing a more sophisticated data augmentation strategy to improve the model's ability to generalize to new data. This may include additional transformations, such as random rotations, zooming, and changes in brightness and contrast.

3. Incorporating face alignment: We will investigate incorporating face alignment techniques during the preprocessing stage to ensure that the faces in the dataset are consistently aligned. This could help improve the algorithm's performance by making it easier for the model to extract meaningful features and recognize the celebrity.

By addressing these potential improvements, we hope to further enhance the performance of our celebrity facial recognition algorithm and make it more robust and versatile for real-world applications.

# Reference

*[1] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556. Retrieved from https://arxiv.org/abs/1409.1556*

*[2] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Retrieved from https://openaccess.thecvf.com/content\_cvpr\_2015/html/Schroff\_FaceNet\_A\_Unified\_2015\_CVPR\_paper.html*

*[3] Yin, X., Yu, X., Sohn, K., Liu, X., & Chandraker, M. (2019). Feature Transfer Learning for Face Recognition with Under-Represented Data. Michigan State University, NEC Laboratories America, University of California, San Diego. Retrieved from https://openaccess.thecvf.com/content\_CVPR\_2019/papers/Yin\_Feature\_Transfer\_Learning\_for\_Face\_Recognition\_With\_Under-Represented\_Data\_CVPR\_2019\_paper.pdf*

*[4] Bansal, A., Castillo, C. D., Ranjan, R., & Chellappa, R. (2017). The Do's and Don'ts for CNN-Based Face Verification. Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW). Retrieved from https://openaccess.thecvf.com/content\_ICCV\_2017\_workshops/papers/w37/Bansal\_The\_Dos\_and\_ICCV\_2017\_paper.pdf*

*[5] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems (NeurIPS). Retrieved from https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf*

*[6] OpenCV. (n.d.). OpenCV: Open Source Computer Vision Library. Retrieved from https://opencv.org/*