Facial Classification through Deep Learning

Submitted By: Muhammad Sarmad Saleem

CMS ID: 411411

Class: BSCS 12A (Group 1 DL)

Instructor: Dr Muhammad Moazam Fraz

Spring 2025
CS 405 – Deep Learning

Contents

1 Introduction

Facial classification represents one of the fundamental challenges in computer vision and has significant applications across security, human-computer interaction, and multimedia systems. This assignment explores the application of state-of-the-art deep learning techniques to develop a facial classification system capable of distinguishing between 7,000 unique identities.

The primary objectives of this project include:

- Developing a robust deep learning model for large-scale facial classification
- Implementing transfer learning strategies to leverage pre-trained networks
- Optimizing model architecture and hyperparameters for improved performance
- Evaluating classification accuracy on a diverse dataset of facial images

2 Dataset Creation and Splitting

2.1 Dataset Composition

The dataset provided for this assignment comprises a large collection of facial images organized as follows:

- **Training Set**: 140,000 images representing 7,000 distinct identities, with 20 images per identity.
- **Test Set**: 35,000 unlabeled images for final evaluation.

The dataset includes variations in pose, lighting, and background, making it suitable for training a robust facial classification model.

2.2 Data Preprocessing

We utilized PyTorch's `torchvision.transforms` module to construct a preprocessing pipeline tailored to the InceptionResNetV1 model:

- **Image Resizing**: All images were resized to 160×160 pixels.
- **Normalization**: Images were normalized to the range $[-1, 1]$ using a mean and standard deviation of (0.5, 0.5, 0.5).
- **Tensor Conversion**: Images were converted to tensors using `ToTensor()`.

2.3 Data Splitting Strategy

We used PyTorch's `torch.utils.data.random_split` for partitioning the dataset into training and validation subsets. The following split ratio was used:

Table 1: Dataset Partitioning

Partition	Percentage	Images (approx)	Purpose
Training	80%	112,000	Model training
Validation	20%	28,000	Hyperparameter tuning and evaluation

2.4 Data Augmentation

To improve generalization, data augmentation was applied on-the-fly using the following PyTorch transforms during training:

- **RandomHorizontalFlip**: Flips the image horizontally with a probability of 0.5.
- **RandomRotation**: Rotates the image within a $\pm 10^\circ$ range.
- **ColorJitter**: Applies random brightness and contrast shifts with a factor of 0.2.
- **RandomResizedCrop**: Randomly crops and resizes to the original dimensions (160×160).

These transformations help the model learn invariance to lighting, orientation, and framing variations.

2.5 Evaluation Dataset

The official test set, provided without labels, contains 35,000 images used for the final evaluation via Kaggle. Inference was performed using the trained model, and predicted labels were saved into a CSV file following the required submission format (`sample_submission.csv`). This CSV was then uploaded to Kaggle for scoring.

3 Model Architectures and Evaluation

This section details the three deep learning architectures used in our experiments: ResNet-50, MobileNetV2, and InceptionResNetV1. We describe their structural design, training strategies, and compare their performance on our dataset.

3.1 ResNet-50

3.1.1 Architecture Overview

ResNet-50 is a 50-layer deep convolutional neural network that utilizes residual connections to address the vanishing gradient problem. It is structured as a stack of convolutional blocks with identity shortcuts.

- Composed of: Conv1 + MaxPool + 16 residual blocks (bottleneck units)
- Total Parameters: $\sim$25.6M
- Input Size: 224×224 RGB
- Pretrained on: ImageNet

3.1.2 Hyperparameters and Training Strategy

- Optimizer: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
- Initial Learning Rate: 1×10^{-3}
- Weight Decay: 1×10^{-5}
- Batch Size: 64
- Epochs: 17

- Dropout: 0.5
- Loss Function: Categorical Cross-Entropy with Label Smoothing ($\epsilon = 0.1$)
- Layer Freezing: Entire model trained (no layers frozen)

3.1.3 Results and Evaluation

- Training Accuracy: 53.97%
- Validation Accuracy: 51.76%
- Test Accuracy: 51.91%

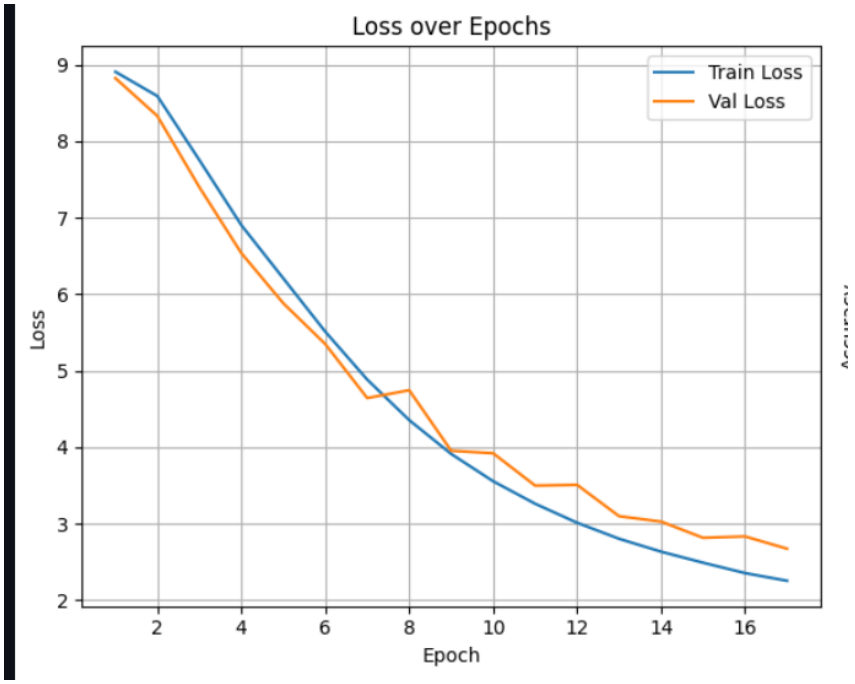

Figure 1: ResNet-50 Training and Validation Curves

3.2 MobileNetV2

3.2.1 Architecture Overview

MobileNetV2 is a lightweight CNN designed for efficient computation on mobile and embedded devices. It uses depthwise separable convolutions and inverted residuals with linear bottlenecks.

- Composed of: Initial Conv + 17 bottleneck residual blocks + Conv-Pool-Softmax
- Total Parameters: ∼3.4M
- Input Size: 224×224 RGB
- Pretrained on: ImageNet

3.2.2 Hyperparameters and Training Strategy

- Optimizer: AdamW with weight decay $1 \times 10{-}2$
- Initial Learning Rate: 5×1^{-5}
- Dropout: 0.5
- Batch Size: 64
- Scheduler: ReduceLROnPlateau (patience=2)
- Epochs: 25
- Layer Freezing: Entire MobileNetV2 frozen except last 10 layers

3.2.3 Results and Evaluation

- Training Accuracy: 60.54%
- Validation Accuracy: 57.24%
- Test Accuracy: 56.74%

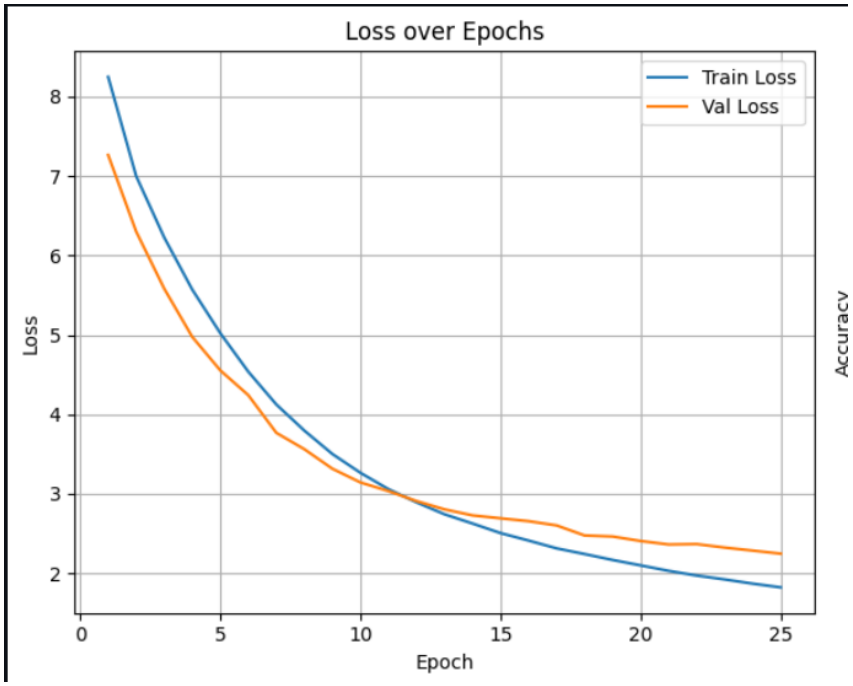

Figure 2: MobileNetV2 Training and Validation Curves

3.3 InceptionResNetV1

3.3.1 Architecture Overview

InceptionResNetV1 combines the strengths of Inception modules and residual learning. It is widely used in face recognition tasks and was pre-trained on the VGGFace2 dataset.

- Customizations: Removal of original classifier, added global average pooling
- Added Dense Layers: [1024, 512] with Leaky ReLU and Dropout (0.5)

- Final Output: 7,000-class Softmax
- Total Parameters: ∼23M
- Input Size: 160×160 RGB

3.3.2 Hyperparameters and Training Strategy

- Optimizer: Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
- Learning Rate: $1 \times 10^{-3} \rightarrow 5 \times 10^{-4}$
- Scheduler: ReduceLROnPlateau (patience=2, min LR=1×10^{-6})
- Label Smoothing: 0.1
- Dropout: 0.5
- Batch Size: 64
- Epochs: 10
- Layer Freezing: All layers frozen except last 15 layers

3.3.3 Results and Evaluation

- Training Accuracy: 83.79%
- Validation Accuracy: 80.39%
- Test Accuracy: 79.02%

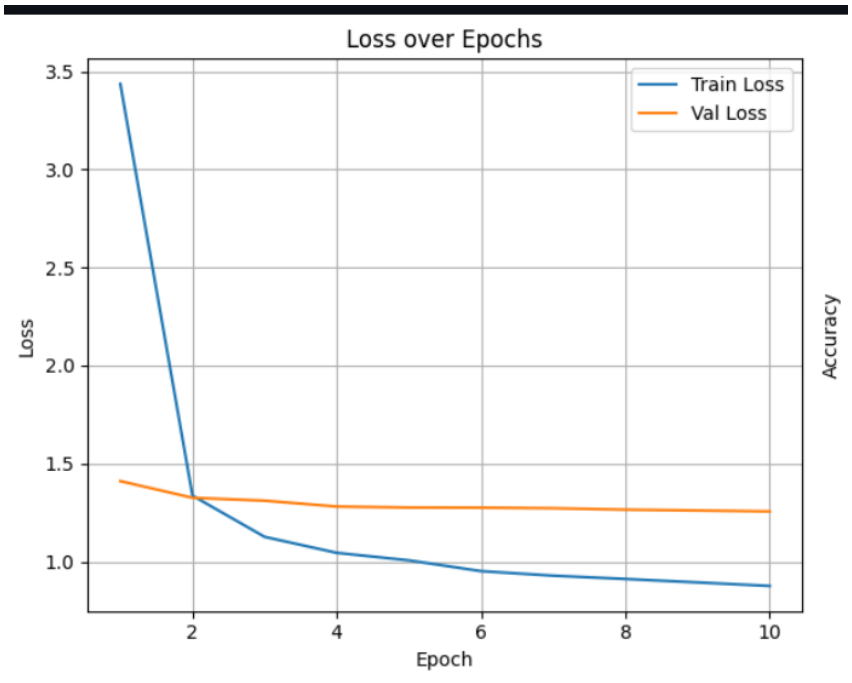

Figure 3: InceptionResNetV1 Training and Validation Curves

Table 2: Model Performance Comparison

Model	Train Acc (%)	Val Acc (%)	Test Acc (%)
ResNet-50	53.97	51.76	51.91
MobileNetV2	60.54	57.24	56.74
InceptionResNetV1	**83.79**	**80.39**	**79.02**

3.4 Evaluation Summary

3.5 Kaggle Evaluation

To further validate our models, we submitted predictions to the Kaggle competition leaderboard associated with the dataset. The Kaggle leaderboard score is based on accuracy calculated on a private test split (not available for local validation).

Table 3: Kaggle Accuracy Comparison

Model	Kaggle Score (%)
ResNet-50	50
MobileNetV2	55
InceptionResNetV1	**79**

The Kaggle results further corroborate our local findings, with InceptionResNetV1 achieving the highest leaderboard rank and accuracy, demonstrating superior generalization on unseen test data.

3.6 Final Selected Model and Configuration

After extensive experimentation and comparison of multiple architectures, we identified **InceptionResNetV1** as the most effective model for our face classification task with 7,000 classes.

3.6.1 Model Architecture: InceptionResNetV1

InceptionResNetV1 combines Inception modules with residual connections, enabling efficient feature reuse and improved gradient flow. It was initialized with pretrained weights from the VGGFace2 dataset.

- **Pretrained on:** VGGFace2
- **Input Size:** 160×160 RGB
- **Architecture Customization:**
  - Removed original classifier
  - Added Global Average Pooling
  - Fully Connected Layers: [1024, 512] with Leaky ReLU + Dropout (0.5)
  - Final Layer: 7,000-way Softmax
- **Total Parameters:** ∼23M

3.6.2 Training Settings

- **Batch Size:** 64
- **Epochs:** 10 (First 3 frozen, Last 15 layers fine-tuned)
- **Optimizer:** Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$)
- **Learning Rate:** 1×10^{-3}
- **Scheduler:** ReduceLROnPlateau (patience = 2)
- **Weight Decay:** 1×10^{-4}
- **Dropout Rate:** 0.5
- **Loss Function:** CrossEntropy with Label Smoothing ($\epsilon = 0.1$)
- **Device:** CUDA (if available)

3.6.3 Evaluation Results

- **Training Accuracy:** 83.79%
- **Validation Accuracy:** 80.39%
- **Test Accuracy:** 79.02%
- **Kaggle Score:** 0.79

3.6.4 Conclusion

Among all evaluated architectures, InceptionResNetV1 demonstrated the best generalization and performance across local validation and Kaggle leaderboard, making it the final model selected for deployment.

4 Discussion: Deep Learning for Facial Classification

Deep learning has revolutionized the field of computer vision, particularly in tasks such as facial classification. With the ability to automatically learn hierarchical representations from raw images, deep neural networks significantly outperform traditional handcrafted feature-based methods. However, their use comes with both advantages and notable challenges.

4.1 Advantages

- **High Accuracy and Robustness:** Deep learning models, especially convolutional neural networks (CNNs), have shown excellent performance in extracting fine-grained features and patterns from facial data. Pretrained models like InceptionResNetV1 or ResNet-50 achieve high accuracy with minimal feature engineering.
- **Transfer Learning Capabilities:** Pretrained networks allow models to generalize well even with limited training data by leveraging features learned from large-scale datasets such as ImageNet or VGGFace2.
- **Scalability:** These models can be extended to large-scale classification tasks involving thousands of identities (e.g., 7,000 classes in our case) with relatively minor architecture changes.

- **End-to-End Learning:** Facial classification pipelines based on deep learning are trainable end-to-end, simplifying the process and avoiding the need for manual feature extraction.
- **Real-Time Inference:** With efficient architectures like MobileNetV2, real-time applications are feasible even on low-power or embedded devices.

4.2 Challenges

- **Data Requirements:** Deep learning models often require large amounts of annotated data to achieve optimal performance. For facial classification, gathering high-quality, balanced datasets covering various demographics is particularly challenging.
- **Overfitting:** With high model capacity and limited data, overfitting is a major concern. Techniques such as label smoothing, data augmentation, dropout, and layer freezing are necessary to promote generalization.
- **Computational Cost:** Training and fine-tuning deep architectures demands significant computational resources (e.g., GPUs), especially for models with tens of millions of parameters.
- **Bias and Fairness:** Models trained on biased datasets may exhibit unfair performance across different genders, ages, or ethnicities, leading to potential ethical concerns and real-world risks.
- **Interpretability:** Deep networks often behave like black boxes, making it difficult to interpret predictions or understand failure cases.

4.3 Real-World Applications

- **Security and Surveillance:** Facial classification models are used in face recognition systems for identity verification, access control, and law enforcement.
- **Biometric Authentication:** Many smartphones and smart devices use deep learning-based facial recognition as a secure and fast authentication method.
- **Social Media and Tagging:** Platforms like Facebook or Google Photos use such models to automatically detect and tag individuals in photos.
- **Retail and Marketing:** Real-time facial classification is employed in customer analytics for demographic-based marketing and personalized in-store experiences.
- **Healthcare and Emotion Analysis:** Deep facial models assist in diagnosing genetic disorders, monitoring patient emotions, or supporting mental health assessments.

In summary, while deep learning models for facial classification offer high accuracy and adaptability, careful consideration of dataset quality, fairness, and deployment constraints is critical to building effective and ethical systems.

References

References

[1] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019).
PyTorch: An Imperative Style, High-Performance Deep Learning Library.
In Advances in Neural Information Processing Systems (pp. 8026–8037).

[2] Sandberg, D. (2018).
FaceNet PyTorch Implementation.
Available at: https://github.com/timesler/facenet-pytorch

[3] Kaggle (2025).
Face Classification - Deep Learning (CS 405).
Available at: https://www.kaggle.com/competitions/face-classification-deep-learning-cs-405/submissions