Efficient Attendance Management Using Face Classification: A Fine-Tuning Approach

Vidya Bharti, Shadab Raza and Animesh Awasthi

Abstract

This paper outlines the development of an attendance management system based on face classification. By leveraging transfer learning and fine-tuning on the pretrained InceptionResNetV1 model, we optimized the system for a small dataset of 11 individuals. The methodology includes dataset preparation, image preprocessing, architectural adjustments, and fine-tuning processes. The results demonstrate the efficiency of fine-tuning over retraining from scratch for small datasets. The system achieves robust face classification while maintaining computational efficiency.

1. Introduction

Attendance tracking is essential in academic and corporate environments but remains prone to errors and inefficiencies when performed manually. Automating this process using computer vision techniques, specifically face recognition, provides a scalable and non-intrusive solution.

Traditional machine learning approaches require extensive labeled data and significant computational resources for model training. In contrast, transfer learning and fine-tuning pre-trained models enable faster deployment and improved accuracy, particularly when working with smaller datasets. This paper highlights the fine-tuning process applied to InceptionResNetV1, leveraging its robust feature extraction capabilities for efficient attendance management.

2. Why Fine-Tuning Instead of Retraining

2.1 Challenges of Retraining from Scratch

Collecting a large dataset of student images was challenging due to logistical and privacy issues. With limited data, retraining a model from scratch posed the risk of overfitting, high computational cost, and inefficiency, as training on small datasets is prone to poor generalization.

2.2 Why Fine-Tuning?

Fine-tuning allowed us to leverage pre-trained models, such as VGGFace2, that already capture essential facial features. By adding classification layers, we adapted the model to our task with minimal data, reducing overfitting and computational cost. Fine-tuning proved to be an efficient and scalable solution for our small dataset.

3. Dataset Preparation

3.1 Data Collection

We collected images of 11 students, each contributing 40 images with varying conditions:

- **Expressions**: Smiling, neutral, serious, etc.
- **Lighting**: Bright, dim, side-lit, and backlit.
- **Angles**: Frontal, slight left/right turns, and slight tilts.
- **Accessories/Hairstyles**: With/without glasses, hairstyle variations.

3.2 Image Preprocessing

3.2.1 Initial Face Cropping: Haar Cascade

We initially used Haar Cascade for face detection but faced issues:

- **False Positives**: Incorrect detections (e.g., non-faces).
- **Inconsistent Results**: Struggled with low light or tilted faces.
- **Limited Robustness**: Failed with occlusions.

3.2.2 Improved Approach: Deep Learning-Based Detection

We switched to OpenCV's DNN module with a pre-trained ResNet SSD model:

- **Higher Accuracy**: Reliable detection even under challenging conditions.
- **Dynamic Padding**: Ensures better framing.
- **Scalability**: Detects multiple faces if necessary.

3.3 Dataset Structure

- **Training Set**: 35 images per student for training.
- **Validation Set**: 5 images per student for validation.

The directory structure is as follows:

```
/dataset/
    train/Student1/img1.jpg ...
    val/Student1/img1.jpg ...
```

3.4 Rationale Behind Data Split

A 7:1 ratio (training:validation) was used to ensure:

- **Adequate Training Data**: 35 images for model learning.
- **Reliable Validation**: 5 images for performance evaluation.

3.5 Quality Assurance

We manually reviewed the cropped faces to ensure data quality before model training.

4. Model Architecture

The model was built on **InceptionResNetV1**, pre-trained on the VGGFace2 dataset, known for its robust face feature extraction capabilities.

4.1 Base Model

The architecture retained the original InceptionResNetV1 layers to extract high-level facial features. To adapt the model for our classification task, additional task-specific layers were introduced:

1. **Convolutional Layers**:
    - **Conv1**: 512 → 256 filters with kernel size 3.
    - **Conv2**: 256 → 128 filters with kernel size 3.
2. **Fully Connected Layers**:
    - **FC1**: Dynamically initialized based on input size.
    - **FC2**: 256 → num_classes for classification.

4.2 Dynamic Layer Initialization

To handle variable input sizes, **initialize_fc1** dynamically calculates the flattened size after convolutional layers. This ensured seamless compatibility between feature maps and the fully connected layer.

5. Fine-Tuning Process

5.1 Model Freezing

The pretrained InceptionResNetV1 layers were frozen to retain the generic facial features learned during training on the VGGFace2 dataset. This step reduced computational costs and focused training efforts on the newly added layers.

5.2 Training Procedure

The fine-tuning process involved training the new layers while leveraging the frozen base model for feature extraction.

Loss Function and Optimization

- **Loss Function**: Cross-Entropy Loss was used, suitable for multi-class classification.
- **Optimizer**: Adam Optimizer provided adaptive learning rates, ensuring stable convergence.

Training Loop

The training loop included:

1. **Forward Pass**: Input images passed through the frozen base and fine-tuned layers.
2. **Backward Pass**: Gradients computed and weights updated for fine-tuned layers.
3. **Validation**: Model performance evaluated after each epoch.

6 Results Analysis

Below is the summarized table for the experimental results of varying batch sizes, epochs, and learning rates:

Batch Size	Epochs	Learning Rate (LR)	Train Loss	Validation Loss	Validation Accuracy (%)	Observation
32	10	0.001	0.0949	0.0792	96.36	High accuracy with low train and validation loss; the large batch size and optimal LR balance training well.
16	10	0.001	0.1228	0.2070	94.55	Slight drop in accuracy and increase in validation loss; smaller batch size reduces stability.

4	10	0.001	0.2506	0.2514	90.91	Smaller batch size causes unstable gradients, higher losses, and lower accuracy.
4	10	0.0001	0.8609	0.7267	81.82	Lower learning rate with small batch size causes slow convergence, leading to higher losses.
4	50	0.0001	0.2882	0.2148	92.73	More epochs help the model converge better despite a small batch size and low LR.
32	50	0.0001	0.1252	0.1101	94.55	Large batch size stabilizes training, and more epochs allow for lower validation loss.
32	50	0.00001	2.3612	2.3716	9.09	Extremely low learning rate prevents the model from learning effectively, resulting in poor performance.

The above analysis of hyperparameters revealed that larger batch sizes (e.g., 32) provide smoother convergence, higher validation accuracy (96.36%), and lower losses, but require more memory. Smaller batch sizes (e.g., 4) lead to noisier gradient updates, slower convergence, and higher losses. An optimal learning rate of 0.001 balances training speed and stability, achieving high accuracy in fewer epochs, while lower rates (e.g., 0.0001) slow convergence, requiring more epochs to perform similarly. Extremely low learning rates (0.00001) hinder learning. Fewer epochs (10) work well with optimal settings, while more epochs (50) benefit low learning rates and small batch sizes but show diminishing returns with higher settings. The best configuration was batch size 32, epochs 10, and learning rate 0.001, offering the best balance of accuracy (96.36%) and convergence.

Sample Test Data Results

After fine-tuning the model with the optimal configuration (batch size 32, epochs 10, learning rate 0.001), the model achieved a validation accuracy of 96.36%. To demonstrate its effectiveness, here are the results from applying the trained model to a set of test images:

Test Image	Predicted Class	Actual Class	Annotated Image
img1.jpg	Ankita	Ankita	img
img2.jpg	Animesh	Animesh	img
img3.jpg	Anmol	Anmol	img
img4.jpg	Deependra	Deependra	img
img5.jpg	Vidya	Vidya	img
img6.jpg	Sonal	Sonal	img

We tested about 25 images and the model classified 20 out of 25 correctly.