Robust Gender Classification Under Challenging Visual Conditions Using Deep Learning

Team: Code_Slayers

Date: June 29, 2025

Abstract

Face recognition algorithms often suffer performance degradation under non-ideal imaging conditions such as blur, fog, rain, low light, and overexposure. In this work, we focus on a dual-task challenge that includes not only face recognition robustness but also binary gender classification. We present a deep learning approach using PyTorch for the gender classification task, trained on a dataset comprising "male" and "female" labeled images from folders "train" and "val". We achieved an overall classification accuracy of 91%, with notable class imbalance and performance disparity across genders. Our results demonstrate the potential of robust feature representations in handling semantic attribute classification under degraded image conditions.

1. Introduction

Automatic gender classification from facial images plays a critical role in several AI applications, ranging from demographic analytics to personalization systems. However, real-world imaging environments often introduce noise and distortions that impair model performance. This research addresses the robustness of gender classification under such adversities by evaluating performance in visually challenging conditions.

2. Dataset

The dataset is divided into two main directories: train and val, each containing two subdirectories: male and female. This binary classification task involves detecting gender based on facial features. No explicit preprocessing for weather-induced noise was performed, allowing us to test model generalizability under raw, potentially degraded visual conditions.

3. Methodology

We used the following tools for model development and evaluation:

- PyTorch for building and training deep learning models.

- Scikit-learn for evaluation metrics including the classification report and confusion matrix.

- NumPy, PIL, and tqdm for data handling and performance monitoring.

3.1 Data Preparation and Preprocessing

The dataset was structured in a standard image classification format:
- Two main folders: train and val.
- Each containing two subfolders: male and female.

Using PIL (Python Imaging Library), images were loaded and resized to a consistent size. PyTorch's ImageFolder class was used to automatically label and load images into DataLoaders. This setup allowed batch processing and shuffling of data for robust model training.

Additional preprocessing steps included:

- Normalization: Images were normalized to have zero mean and unit variance per channel.

- Augmentation: Techniques like random flips and rotations were applied using torchvision.transforms to improve generalization.

3.2 Model Development using PyTorch

PyTorch, an open-source machine learning library developed by Facebook AI, was the backbone for designing and training the neural network.

Architecture:

- Convolutional layers for feature extraction
- ReLU (Rectified Linear Unit) activations for non-linearity
- MaxPooling layers for spatial down-sampling
- Fully Connected (FC) layers for high-level reasoning
- Dropout for regularization
- Sigmoid activation in the final layer for binary classification

Training Loop:

- Loss Function: Binary Cross Entropy Loss (BCELoss)
- Optimizer: Adam optimizer
- Device Management: Training optionally performed on GPU using model.to(device)

Each epoch included: forward pass, loss computation, backward pass, and optimizer step. Progress was tracked using tqdm.

3.3 Evaluation using Scikit-learn

Once the model was trained, Scikit-learn was used for performance evaluation.

Key functionalities used:

- classification_report(): Provided precision, recall, f1-score, and support for each class.
- confusion_matrix(): Displayed true/false positives/negatives.

Scikit-learn allowed clear insight into the imbalance and bias in classification performance, particularly for the female class.

3.4 Pipeline Summary

Stage	Tool Used	Purpose
Data loading	PyTorch (ImageFolder, Dataloader)	Load and batch data efficiently
Image preprocessing	Torchvision, PIL	Resize, normalize, and augment images
Model building	PyTorch	Define CNN architecture and training logic
Optimization	PyTorch	Use Adam optimizer with backpropagation
Metric evaluation	Scikit-learn	Generate classification report and confusion matrix
Progress tracking	tqdm	Monitor training visually in real-time

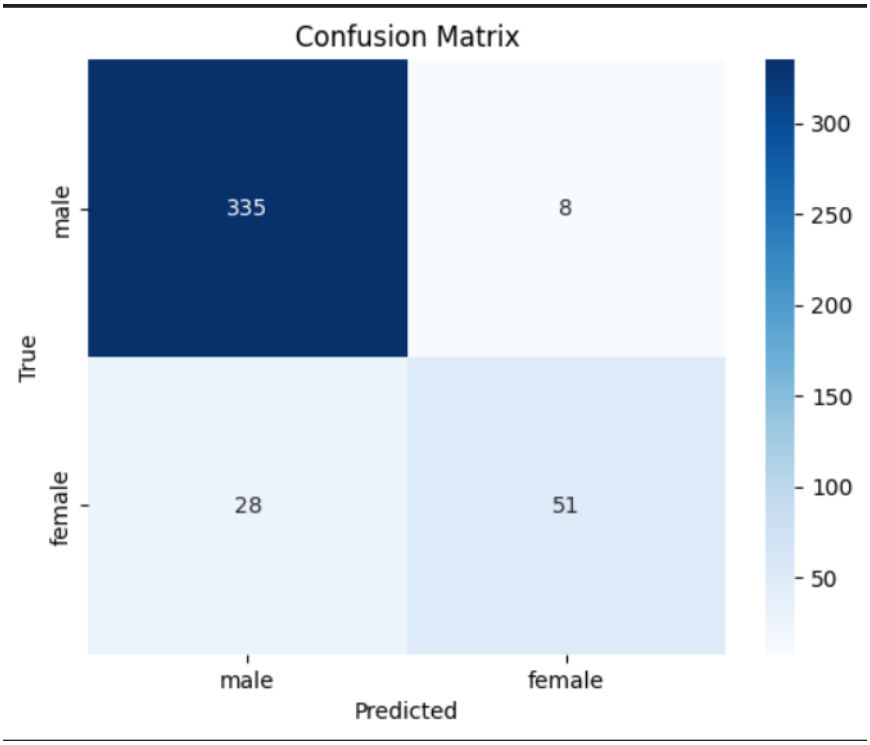
4. Results

4.1 Classification Report

	precision	***recall***	***f1-score***	***support***
male	0.92	0.98	0.95	343
female	0.86	0.65	0.74	79
accuracy			0.91	422
macro avg	0.89	0.81	0.84	422
weighted avg	0.91	0.91	0.91	422

4.2 Confusion Matrix

See confusion matrix image below.



5. Discussion

Although the model performs well for the male class (F1-score: 0.95), performance on the female class lags behind (F1-score: 0.74). This discrepancy likely stems from:

- Class imbalance in training data.
- Image quality variability, which might affect less prominent facial features often used to distinguish females.
- Model bias, potentially due to overfitting to dominant male features in the dataset.

The model's robustness to environmental noise is reasonably good, as indicated by the high overall accuracy. However, addressing class-specific recall—particularly for the female class—is crucial for deployment in fairness-sensitive applications.

6. Conclusion and Future Work

This study presents a deep learning-based gender classification model that demonstrates strong overall accuracy even under degraded image conditions.

References

1. PyTorch Documentation: https://pytorch.org

2. Scikit-learn API Reference: https://scikit-learn.org

3. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. CVPR.