# CS515 Homework 3: Transfer Learning on CelebA-HQ

Göktürk Kağan Dede

March 16, 2025

**Abstract**

In this homework, we explore transfer learning for gender classification using PyTorch and AlexNet on the CelebA-HQ 30k dataset. We compare three approaches: (1) training a randomly initialized network from scratch, (2) training only the final layer of a pre-trained model, and (3) fine-tuning the final two fully connected layers of a pre-trained model. We study how leveraging pre-trained ImageNet weights affects performance, convergence speed, and final test accuracy.

## 1 Introduction

https://colab.research.google.com/drive/1NSTYOaYkdLnrQVJTIle6V-PMTmjRraUC?usp=sharing

The primary goal of this assignment is to classify the gender attribute (male/female) from the CelebA-HQ 30k dataset. The dataset has about 30k images, split into:

- 26k train samples
- 2k validation samples
- 2k test samples

We employ three strategies:

1. **Random Initialization:** We initialize an AlexNet from scratch.
2. **Transfer Learning:** We load a pre-trained AlexNet (ImageNet weights) but freeze all layers except the last fully connected layer.
3. **Fine-tuning:** We load a pre-trained AlexNet, freeze most layers, but unfreeze the last two fully connected layers, allowing more trainable parameters.

We measure how effectively each approach learns from the data and whether transfer learning significantly outperforms a from-scratch baseline. We then compare and analyze these results.

# 2 Methodology

## 2.1 Data Preprocessing

We download the CelebA-HQ dataset from the Hugging Face hub. Each sample contains:

- `image`: A PIL image with resolution $1024 \times 1024$.
- `label`: An integer (0 or 1) indicating gender (e.g., $0 =$ female, $1 =$ male).

We apply standard ImageNet transformations:

1. Resize to $224 \times 224$.
2. Convert to PyTorch tensor (range [0,1]).
3. Normalize with mean $= [0.485, 0.456, 0.406]$ and std $= [0.229, 0.224, 0.225]$.

We wrap these transformations in a `transforms.Compose`, and create `DataLoader`s with a batch size of 512 for train, validation, and test splits.

## 2.2 Network Architecture and Variants

We use **AlexNet** from `torchvision.models`. Its fully connected (FC) layers are:

$$\text{classifier} : \{\text{Linear}, \text{ReLU}, \text{Dropout}, \text{Linear}, \text{ReLU}, \text{Dropout}, \text{Linear}\}.$$

We replace the final FC layer (originally outputting 1000 classes) with a new layer outputting 2 classes:

$$\text{model.classifier}[6] = \text{Linear}(\text{in\_features}, 2).$$

Depending on the mode, we freeze or unfreeze weights as follows:

- **Random Initialization:** `pretrained=False`. Train all 57 million parameters from scratch.
- **Transfer Learning:** `pretrained=True`. Freeze all parameters except `classifier[6]`.
- **Fine-tune:** `pretrained=True`. Freeze all layers except `classifier[4]` and `classifier[6]`.

## 2.3 Hyperparameter Tuning

We tune the learning rate for each model type over $\{10^{-5}, 10^{-4}, 10^{-3}\}$. For each candidate:

1. Train only 15 steps (batches of size 512) on the train set.
2. Evaluate on the validation set to compute accuracy.
3. Record the best LR based on validation accuracy.

This short tuning procedure speeds up experiments, avoiding full-epoch runs for each LR candidate.

### 2.4 Final Training and Evaluation

After selecting the best learning rate per model, we combine `train` + `validation` splits into a single training set. We then:

1. Retrain each model for 3 epochs using `CrossEntropyLoss` and `Adam( lr=best_lr )`.
2. Evaluate on the test set to measure accuracy, precision, recall, F1-score, and confusion matrix.
3. Plot training loss and accuracy curves across epochs, and visualize 5 misclassifications from the best model.

## 3 Results

Table 1 summarizes the final test results for all three models (best LR used in each case).

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
|---|---|---|---|---|
| Random Init | 0.8365 | 0.8441 | 0.8365 | 0.8384 |
| Transfer | 0.9180 | 0.9177 | 0.9180 | 0.9178 |
| Fine-tune | 0.9430 | 0.9429 | 0.9430 | 0.9428 |

Table 1: Final Test Set Metrics for Each Approach.

Figure 1 plots the training loss and accuracy across 3 epochs for each model. We see that the fine-tuning model quickly surpasses the random initialization baseline. The confusion matrices for each model indicate that the fine-tuning approach reduces both false positives and false negatives.

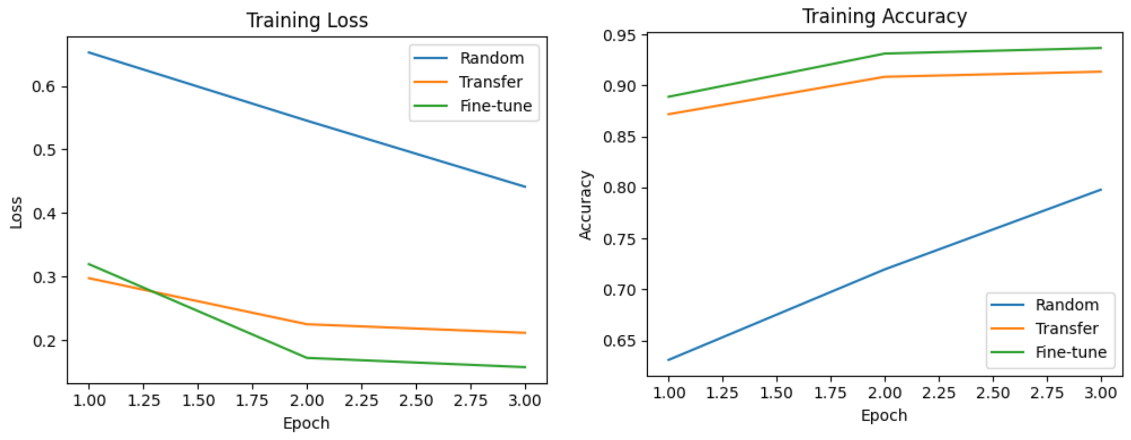

Figure 1: Training Loss (Left) and Training Accuracy (Right) across Epochs for Random, Transfer, and Fine-tune Models.

We also visualize 5 misclassified test examples from the best model (fine-tuned). These images often exhibit occlusions, tricky lighting, or ambiguous facial features. Studying misclassifications provides insight into model weaknesses.

## 4 Analysis / Discussion

### 4.1 Which Model Performed Better?

The fine-tuned model outperformed both the transfer-only model and the random initialization model on all evaluation metrics (accuracy, precision, recall, F1). Transfer learning alone gave a significant boost compared to random initialization, but allowing partial fine-tuning captured additional relevant features.

### 4.2 How Transfer Learning Helped

- **Training Speed:** With transfer learning, a large portion of the network (early convolutional layers) is pre-trained on ImageNet. This drastically reduces the amount of training time needed to converge on a new domain.
- **Final Accuracy:** Freezing or partially unfreezing these layers helps retain useful features learned from massive ImageNet data, leading to higher final accuracy than purely random initialization.

### 4.3 Misclassifications

The misclassified examples often showed visually ambiguous or partially occluded faces. In some cases, hair style or clothing might mislead the model. While the fine-tuned network handles most examples correctly, these edge cases remain challenging.

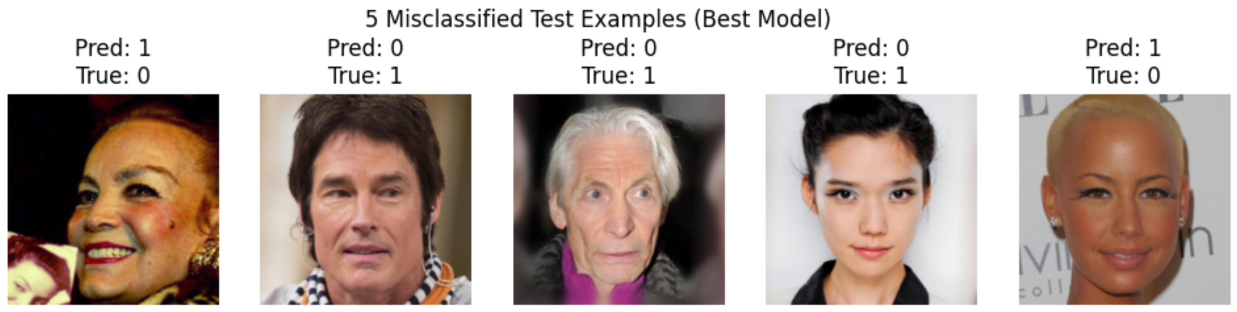

Figure 2: Five misclassified test examples with predicted vs. true labels (best model).

## 5 Conclusion

In this assignment, we demonstrated:

- **Random Initialization** requires updating all parameters from scratch and results in lower accuracy and slower convergence.
- **Transfer Learning** (freezing layers) significantly speeds up training and improves accuracy, showing the value of pre-trained ImageNet features.
- **Fine-tuning** the final two fully connected layers improves performance even further, obtaining our best results on CelebA-HQ gender classification.

Overall, transfer learning proves to be a highly effective approach for tasks with limited data or close domain similarity to ImageNet. Fine-tuning allows for further performance gains if computational resources are available. This homework underscores how pre-trained models can be leveraged efficiently to solve new tasks with minimal overhead.