Celebrities Recognition – Neural Network Approach

🧠 Neural Networks for Multi-Class Celebrity Classification (98 Classes)

Objective

Train and evaluate different neural network models to classify images of 98 celebrities. We compare their performance based on accuracy scores using PyTorch and transfer learning.

Dataset Preparation

- Images are loaded and processed into PyTorch `DataLoaders`.
- Each model uses tailored image transformations depending on architecture (e.g. input size).
- Dataset is split into:
  - **Training set**: 70% of the data
  - **Validation set**: 15% of the data
  - **Test set**: 15% of the data

Libraries Used

- `torch`, `torchvision`
- `numpy`, `pandas`
- `seaborn`, `matplotlib`
- `scikit-learn`

Training Pipeline

- Custom PyTorch training loop with:
  - Epoch-level tracking of:
    - Training Accuracy / Loss
    - Validation Accuracy / Loss
  - Batch-level progress tracking
  - Best model checkpointing
  - Visualization of training progress
  - Custom image inference test
  - Model save/load for reuse

Experiments and Results

🔷 TinyVGG (Baseline)

- **Data Used**: 5%
- **Validation Accuracy**: ~1.5%
- **Notes**: Only slightly better than random guessing. Serves as baseline.

🔷 EfficientNetB0 (20% Data)

- **Validation Accuracy**: ~23%
- **Notes**: Big improvement from baseline using transfer learning.

🔷 EfficientNetB0 (50% Data)

- **Training Accuracy**: ~73%
- **Validation Accuracy**: ~37%
- **Notes**: Signs of overfitting. Model performs well on training but less on validation.

🔷 MobileNetV2 (100% Data)

- **Validation Accuracy**: ~50%
- **Notes**: Best generalization so far. Trained on full dataset.

Visualizations

- Training and validation curves:
  - Accuracy
  - Loss
- Sample predictions from custom input images
- Final confusion matrix:
  - Generated using `seaborn`
  - Highlights class imbalance
  - Used to identify which classes may need more samples and show on which classes our model could be improved.

Final Notes

- All trained models are saved and reloaded successfully.
- Overfitting addressed in later models via augmenration techniques and architecture changes.
- Heatmap of results on test set guides data collection improvements.

Future Improvements

- Data augmentation
- Better class balancing

- Regularization techniques (dropout, weight decay)
- Advanced architectures (e.g., ViT, ResNet variants)