

Celebrity Face Classification

Comparative analysis of four different
models

Team Members

Roaa Mohamed	20220169
Zeinab Sayed	20220193
Aliaa Ali	20220302
Mariam Hany	20220473
Menna Ahmed	20220506
Mona Yousef	20220512

Introduction

This documentation describes a project for celebrity face classification: building and evaluating deep-learning models that identify which celebrity appears in a given face image. The project uses the VGGFace2 dataset (with references to LFW where applicable) and explores both from-scratch and transfer-learning approaches. Four architectures were implemented and compared: VGG-19 (implemented from scratch), ResNet (pretrained; transfer learning on our dataset), Inception-V1 (pretrained; transfer learning on our dataset), MobileNet.

The primary goals are to (1) document each selected architecture with step-by-step implementation details, (2) show model behavior and performance on VGGFace2, and (3) provide a clear comparative analysis explaining why certain architecture works better (or worse) for celebrity face recognition.

VGG-19 Convolutional Neural Network

Overview

The VGG-19 convolutional neural network (CNN) is a deep learning architecture notable for its depth, comprising 19 layers, and its straightforward design, which employs a consistent use of small 3×3 convolutional filters. Developed as an extension of the VGG-16 model, VGG-19 primarily differs in the number of layers. This architecture achieved runner-up status in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2014.

Architecture

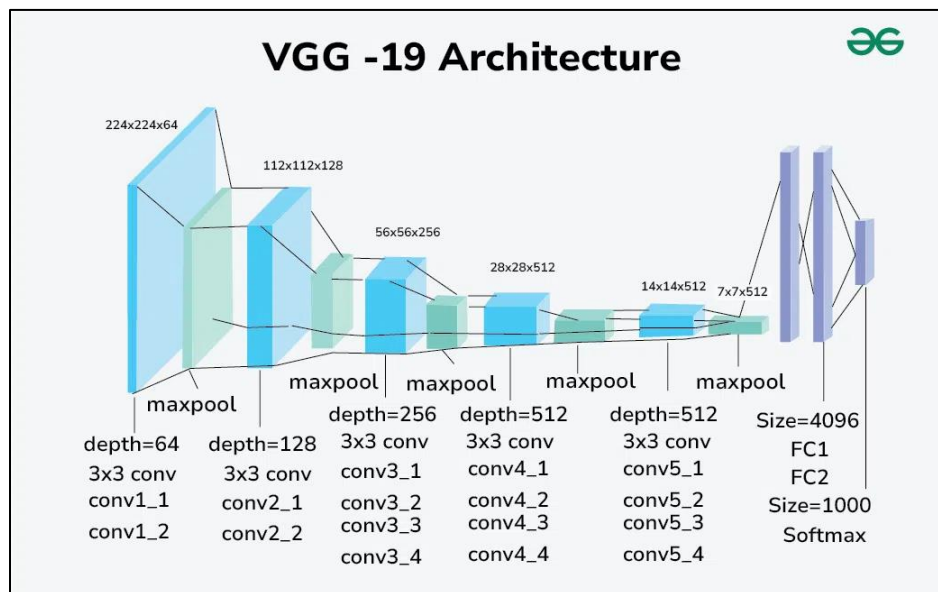
The VGG-19 network comprises 19 weight layers, including 16 convolutional layers and 3 fully connected layers, along with 5 max-pooling layers and a final softmax layer for classification. The convolutional layers employ small 3×3 filters with a stride of 1 and same padding, which preserves the spatial resolution of the input image. Max-pooling is performed using 2×2 windows with a stride of 2 between convolutional blocks to reduce the spatial dimensions of feature maps.

The network architecture consists of two convolutional blocks with two convolutional layers each, followed by three blocks with four convolutional layers each. After these layers, the output is flattened and passed through two fully connected layers of 4096 channels each, then a final fully connected layer with 1000 channels for classification via a softmax activation function.

ReLU activations are applied to all convolutional and fully connected layers to introduce nonlinearity and enhance computational efficiency. Stacking small 3×3 filters increase the network's capacity for hierarchical feature extraction and representation learning while maintaining parameter efficiency. Architectural variants may incorporate global average pooling and dropout layers for improved regularization, such as a global average pooling

2D layer followed by a dropout layer with a rate of 0.5, connected to a fully connected layer with a sigmoid activation function.

Since the model is very large with 2 dense layers and more than 4000 neurons, it needs a lot of computational power and RAM usage. Instead of that, we implemented just one Max-pooling layer with 512 neurons, which decreases the RAM usage with 50%, making it lighter, faster, and more suitable for our local devices.



Advancements

- ❖ Deep architecture enables strong feature extraction, capturing both low-level and high-level features.
- ❖ Works effectively as feature extractor in transfer-learning pipelines.
- ❖ Produces fused feature representations with reduced noise and artifacts.
- ❖ Techniques such as dropout help improve generalization during training.
- ❖ Batch normalization normalizes activations, improving convergence speed and generalization performance.

Limitations

- ❖ Very large model size (>550 MB) makes deployment challenging on limited-resource devices.
- ❖ Long inference time due to depth and number of parameters.
- ❖ High computational cost, especially compared to newer architectures.
- ❖ Suffers from challenges such as the vanishing gradient problem in deep networks.

Residual Neural Networks

Overview

The Residual Blocks idea was created to address the issue of the vanishing/evaporating gradient. We apply a method known as skip connections. The skip connection bypasses some levels in between link-layer activations to subsequent layers. This creates a leftover block. These leftover blocks are stacked to create resnets.

The idea behind this network is to let the network fit the residual mapping rather than have layers learn the underlying mapping. Thus, let the network fit rather than using the initial mapping.

The benefit is that regularization will skip any layer that degrades the architecture performance. As a result, training an extremely deep neural network is possible without encountering problems with vanishing or expanding gradients.

Architecture

Unlike traditional CNNs that stack convolutional layers sequentially, ResNet introduces residual learning, where layers explicitly learn a residual function with reference to the layer inputs.

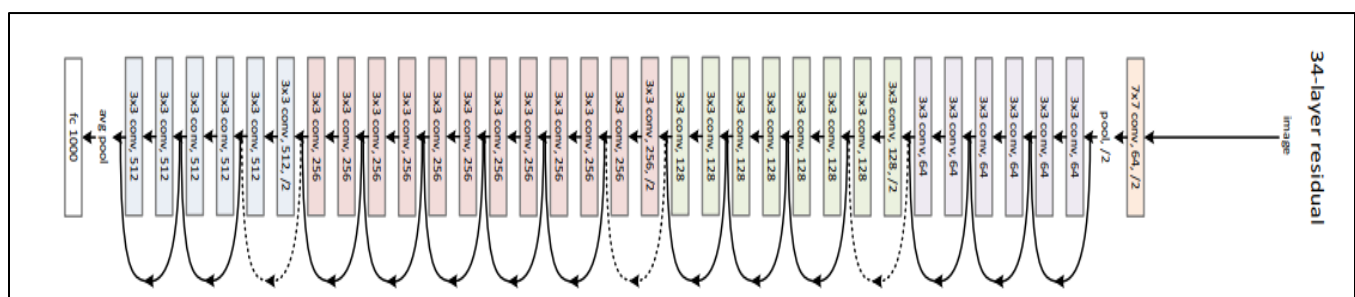
The key structural element of ResNet is the residual block, which consists of two or three stacked convolutional layers (typically 3×3 filters), each followed by batch normalization and ReLU activation, and a shortcut connection that bypasses these layers and adds the input directly to the block's output.

This shortcut connection enables the model to train much deeper networks such as ResNet-18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, without suffering from vanishing gradients.

Shallow variants use basic blocks (2-layer residual units), while deeper variants like ResNet-50 and above use bottleneck blocks consisting of 1×1 convolution for dimensionality reduction, 3×3 convolution for feature extraction, and 1×1 convolution for dimensionality restoration.

The architecture begins with a 7×7 convolution with stride 2, followed by Max-pooling, then multiple stages of residual blocks where the number of filters increases at each stage. After the residual stages, ResNet ends with global average pooling layer, then a fully connected (dense) layer for classification.

This architectural design allows gradients to flow easily through many layers and enables stable training of extremely deep models.



Advancements

- ❖ Very deep models can train effectively without vanishing gradients.
- ❖ Easier optimization, as the network learns residual mappings rather than full transformations.
- ❖ Improved accuracy on large-scale datasets.
- ❖ Highly efficient representation learning, allowing the model to extract strong hierarchical features.
- ❖ Better generalization when used as a feature extractor in transfer learning tasks.
- ❖ Bottleneck blocks significantly reduce computational cost while maintaining performance.
- ❖ Global average pooling reduces parameter count and prevents overfitting compared to large fully connected layers.

Limitations

- ❖ Deep variants are computationally heavy.
- ❖ Large number of layers increases training time and memory usage.
- ❖ Shortcut connections restrict flexibility, the input and output dimensions must match or require extra projection layers.
- ❖ May overfit on small datasets without strong regularization or data augmentation.
- ❖ Bottleneck blocks add architectural complexity compared to simpler CNNs.

Inception V1

Overview

The Inception architecture is designed to approximate an optimal local sparse structure in convolutional networks using dense, readily available components. Since the model assumes translation invariance, it is constructed from convolutional building blocks that are repeated spatially across the network.

It combines multiple convolutional filter sizes, 1×1 , 3×3 , and 5×5 , in parallel, along with a pooling path, and concatenates their outputs to form the input to the next stage. To control computational cost, the architecture uses 1×1 convolutions for dimension reduction before expensive operations. Stacked Inception modules process visual information at multiple scales efficiently while preventing computational blow-up.

Architecture

The architecture is based on analyzing correlation statistics in each layer and grouping highly correlated units into clusters. These clusters correspond to filter banks in the next layer. In lower layers, correlated units tend to concentrate in local image regions, which can be covered using 1×1 convolutions. At the same time, more spatially spread-out clusters require convolutions over larger regions, motivating the use of 3×3 and 5×5 filters. To avoid alignment issues, the Inception architecture restricts filter sizes to 1×1 , 3×3 , and 5×5 .

Each Inception module combines parallel convolutional paths with different filter sizes, and their outputs are concatenated into a single output vector that becomes the input to the next stage. Because pooling is essential in convolutional networks, an additional parallel pooling path is included within the module.

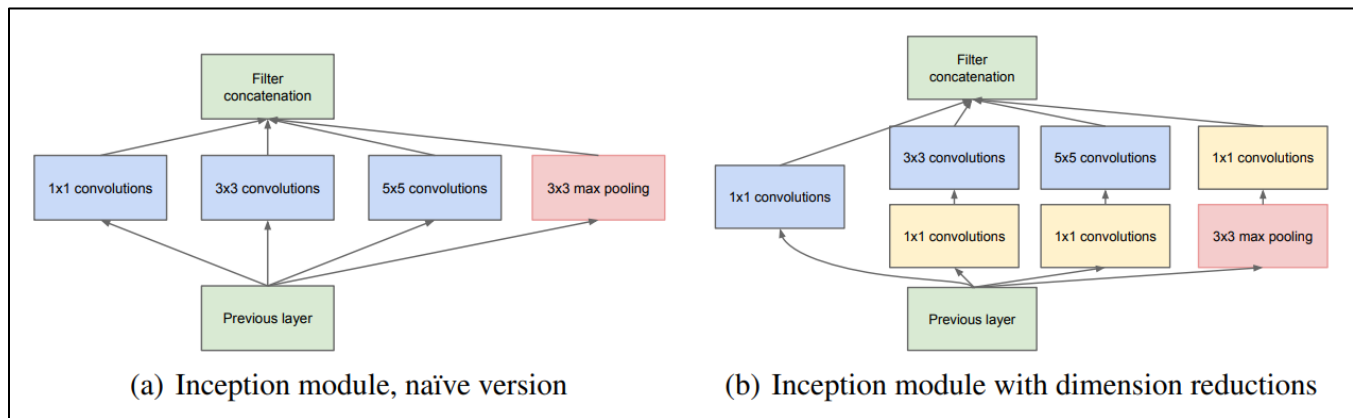
As these modules are stacked, higher layers capture features with lower spatial concentration, so the architecture uses a higher ratio of 3×3 and 5×5 convolutions in deeper layers.

To prevent computational explosion caused by stacking modules with many filters, especially due to expensive 5×5 convolutions and pooling output, Inception introduces dimension reduction. This is achieved by applying 1×1 convolutions before the 3×3 and 5×5 convolutions, reducing the number of input channels and lowering computational cost. These 1×1 convolutions also include ReLU activations, making them dual-purpose for both feature embedding and nonlinearity.

An Inception network consists of multiple such modules stacked together, with occasional max-pooling layers (stride 2) to halve spatial resolution. For memory-efficiency reasons, Inception modules are typically applied starting from higher layers, while lower layers may remain in traditional convolutional form.

The architecture allows a significant increase in the number of units at each stage without uncontrolled computational growth due to its extensive use of dimension reduction. This design supports multi-scale processing by aggregating information from different filter sizes simultaneously, enabling the next stage to abstract features captured at different spatial scales.

Additionally, the modular design allows computational trade-offs by adjusting components within the Inception module, enabling cheaper variants that operate 2–3 \times faster with similar performance, though this requires manual tuning.



Advancements

- ❖ Significant improvement in classification and detection performance, achieving state-of-the-art results in ILSVRC 2014.
- ❖ Improved utilization of computational resources, allowing the network to increase depth and width while keeping computation roughly constant.
- ❖ Uses multi-scale processing, combining 1×1 , 3×3 , and 5×5 filters and pooling in parallel to capture features at different scales.
- ❖ Dimension reduction (1×1 convolutions) greatly reduces computation and prevents computational blow-up in deeper layers.
- ❖ Allows many more units per stage without uncontrolled growth in computational complexity.
- ❖ Enables both deeper and wider architectures to be trained efficiently.
- ❖ Efficient enough for real-world use, designed to stay within a feasible computational budget ($\approx 1.5\text{B}$ operations) and suitable for devices with limited resources.
- ❖ Supports creating cheaper, faster variants ($2\text{--}3\times$ faster) by adjusting module components.
- ❖ Auxiliary classifiers improve gradient flow, providing additional regularization and helping deeper layers train effectively.
- ❖ Better computational distribution helps prevent inefficiencies that occur when model capacity is increased uniformly.

Limitation

- ❖ Naïve Inception modules are computationally expensive, especially the 5×5 convolutions and pooled feature concatenation.
- ❖ Pooling outputs increase filter counts, leading to rapid growth in the number of output channels without careful dimension reduction.
- ❖ Compressed (embedded) representations are harder to model, making training more complex when strong dimensionality reduction is used.
- ❖ Manual design is required to balance computational cost and accuracy for different module variants; automated design is not yet available.
- ❖ Training is memory-intensive, motivating the decision to use Inception modules only in higher layers for practical reasons.
- ❖ Architecture decisions are complex, requiring careful tuning of reduction layers and filter sizes.
- ❖ Theoretical motivation is not fully verified, and it is unclear whether the guiding principles guarantee optimal designs in general.

MobileNet V2

Overview

MobileNetV2 is a convolutional neural network architecture designed to improve the trade-off between accuracy, efficiency, and computational cost for mobile and resource-constrained environments. It builds upon the principles of MobileNet by introducing new architectural modules that enable efficient feature representation while reducing information loss. The architecture is specifically optimized for low-latency and low-power applications, making it suitable for mobile vision tasks.

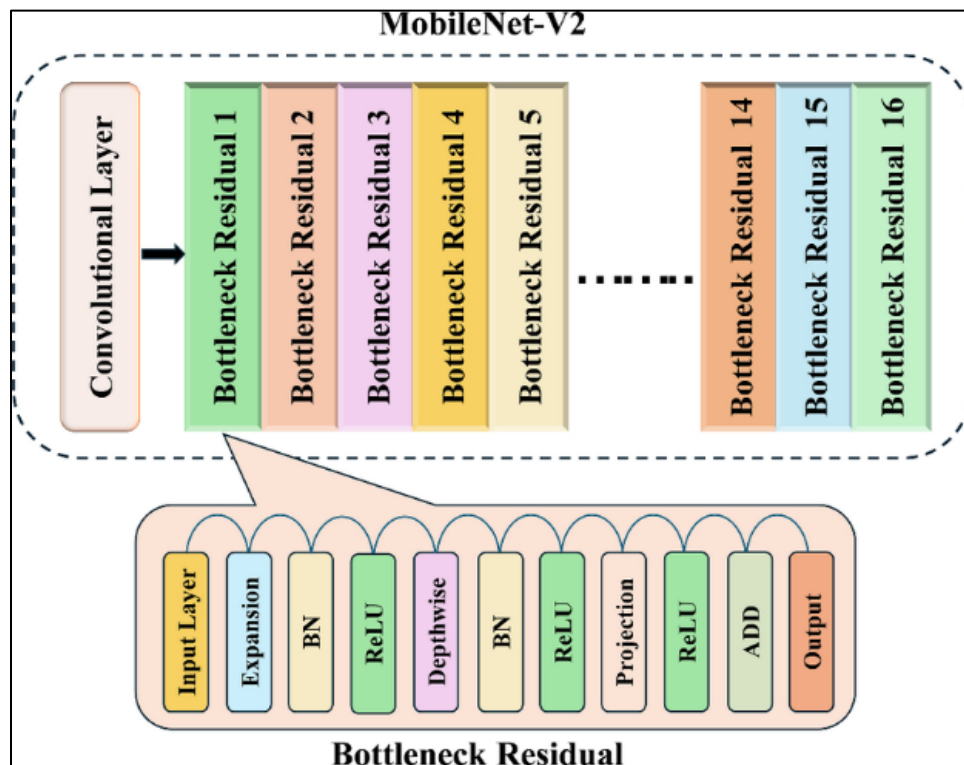
Architecture

MobileNetV2 introduces two key architectural innovations; inverted residuals and linear bottlenecks, which together enable efficient feature representation while reducing computational cost.

The core building unit of the network is the inverted residual block, which differs from traditional residual blocks by connecting residual shortcuts between thin bottleneck layers rather than wide feature representations. Each block begins with a 1×1 pointwise convolution that expands the number of channels, creating a higher-dimensional feature space where richer feature extraction can occur. This expansion is followed by a 3×3 depthwise convolution, which operates independently on each channel and is responsible for capturing spatial information efficiently. The expanded features are then projected back into a lower-dimensional space using a final 1×1 pointwise convolution, forming what is known as a linear bottleneck.

Unlike earlier architectures, this projection layer does not apply a non-linear activation function, which helps preserve important information that might otherwise be lost. Residual (skip) connections are incorporated when the input and output dimensions of a block match, allowing gradients to flow more effectively through the network during training. Most of the network's non-linear activations use ReLU6, which is better suited for low-precision computation.

Downsampling within the architecture is achieved through stride-2 depthwise convolutions in specific bottleneck blocks. Overall, the design maintains high efficiency by minimizing memory access costs while preserving strong representational power.



Advancements

- ❖ Introduces inverted residuals, improving information flow while reducing memory usage.
- ❖ Linear bottlenecks prevent information loss, especially in low-dimensional feature spaces.
- ❖ Improves accuracy compared to MobileNetV1 at similar or lower computational cost.
- ❖ Highly efficient for mobile and embedded devices, with reduced latency and power consumption.
- ❖ Optimized for low-precision arithmetic, making it suitable for real-world deployment on mobile hardware.
- ❖ Enables better performance across multiple vision tasks while maintaining a compact model size.

Limitations

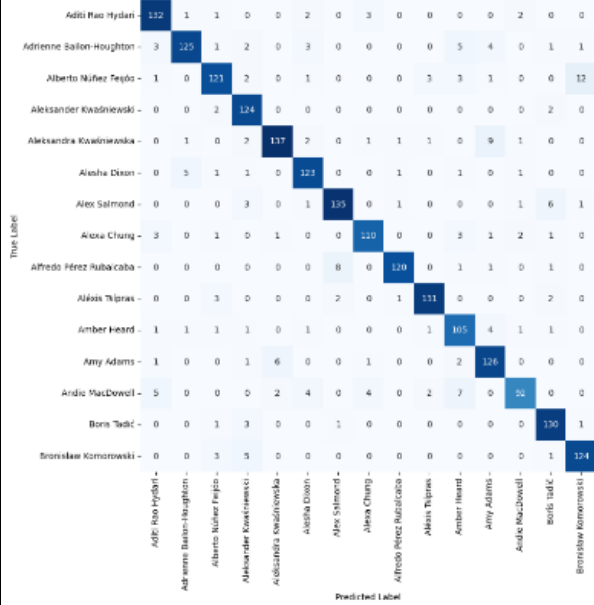
- ❖ Performance depends on careful block design, including expansion factors and bottleneck sizes.
- ❖ Linear bottlenecks restrict non-linearity, which may limit expressiveness if misused.
- ❖ Residual connections are only applicable when dimensions match, limiting their use across all layers.
- ❖ While efficient, the architecture prioritizes resource efficiency over maximum possible accuracy.

Comparisons

	VGG-19	ResNet	Inception V1	MobileNet
Accuracy	0.91	0.84	0.75	0.95
Recall	0.91	0.84	0.75	0.95
Precision	0.91	0.84	0.75	0.95
F-Score	0.91	0.84	0.74	0.95

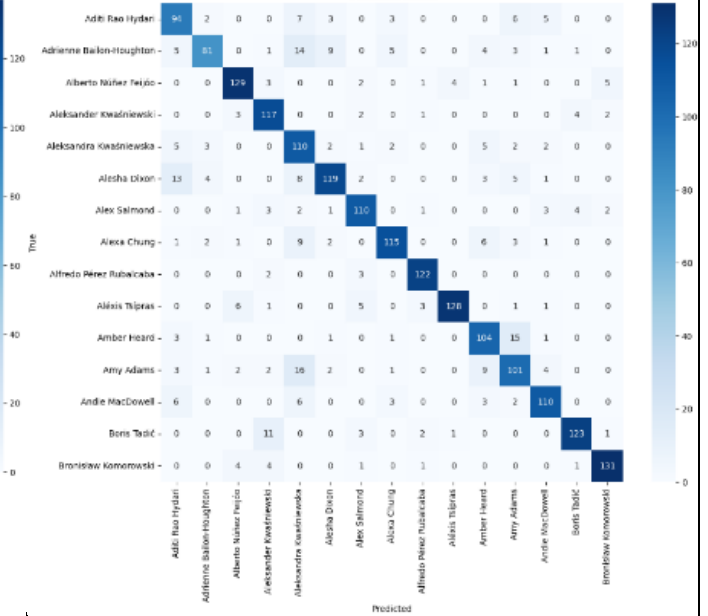
VGG-19

Confusion Matrix: Who is getting confused with whom?



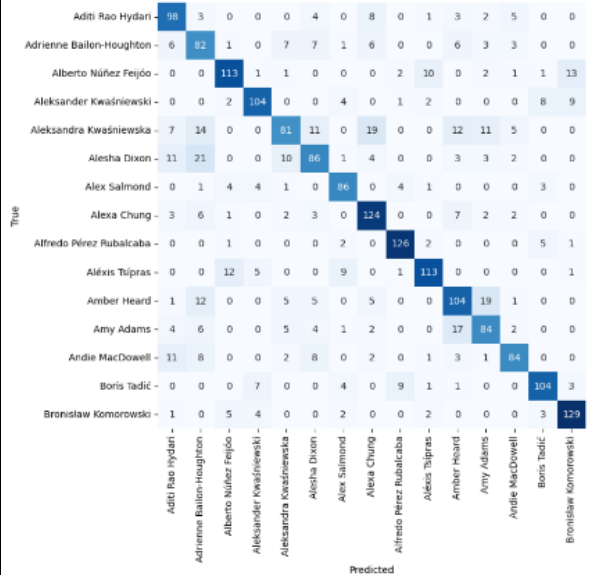
ResNet

Confusion Matrix



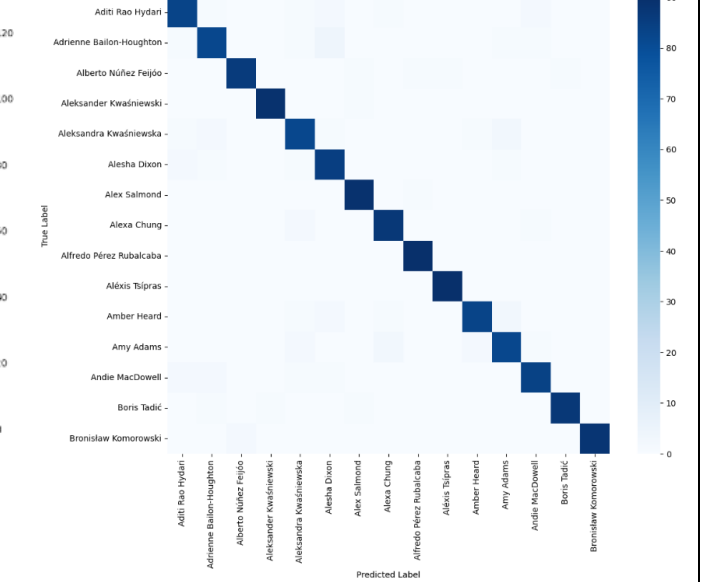
Inception V1

Confusion Matrix

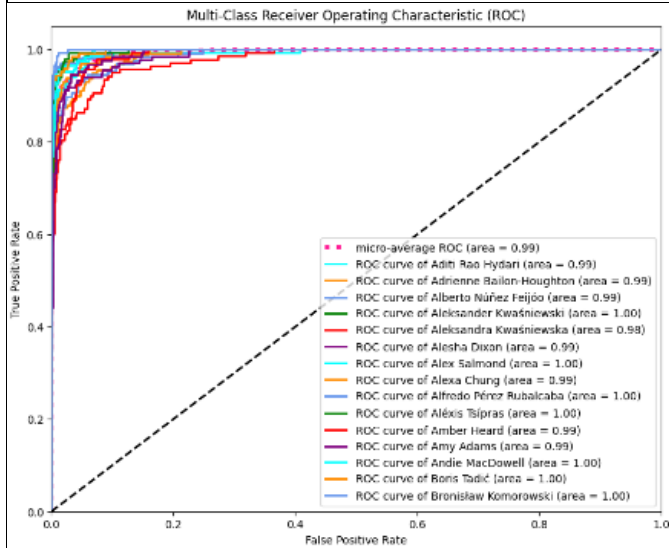


MobileNet V2

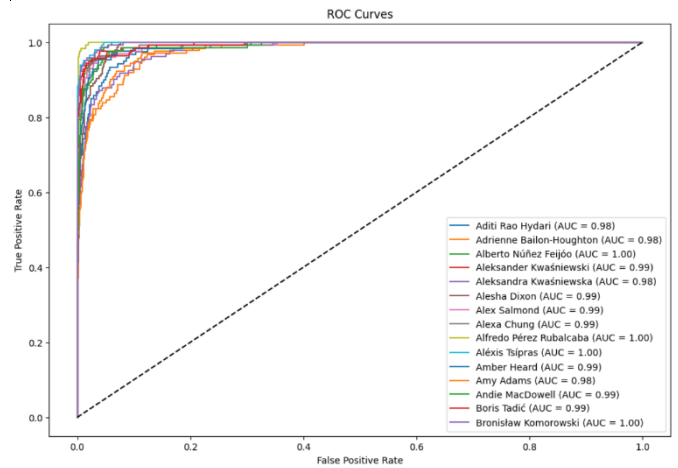
Confusion Matrix



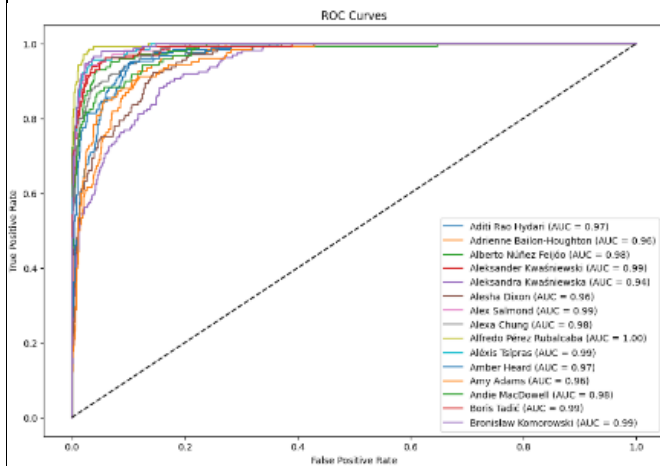
VGG-19



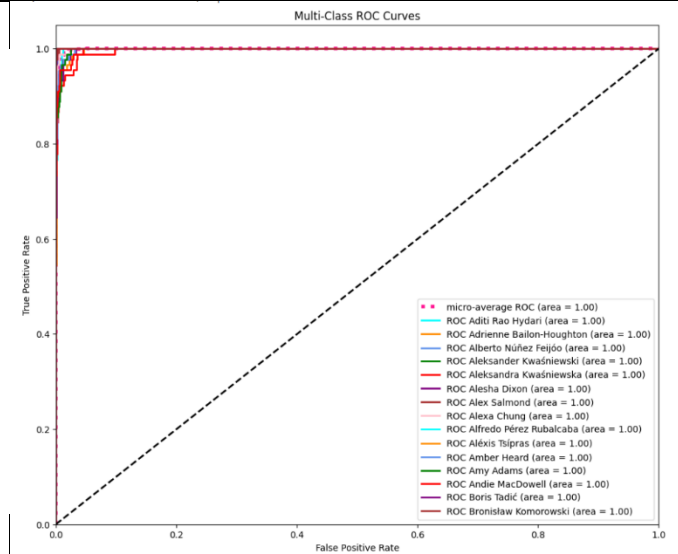
ResNet



Inception V1



MobileNet V2



Why MobileNetV2 Performs Better for This Task and Dataset

MobileNetV2 achieves the best performance in this project, outperforming VGG-19, ResNet, and Inception V1. This superior performance can be explained by how well MobileNetV2's architecture aligns with the nature of face recognition, the characteristics of the VGGFace2 dataset, and the practical training constraints the project.

❖ Better Balance Between Model Capacity and Efficiency

Celebrity face classification requires the model to focus on fine-grained facial details (eyes, nose, mouth, facial proportions) rather than highly complex scene-level features. MobileNetV2 is specifically designed to preserve informative features while avoiding unnecessary complexity.

VGG-19 is very deep and parameter-heavy, which makes it prone to overfitting and high memory usage, especially on limited hardware. ResNet and Inception V1 are powerful but still relatively heavy and optimized for large-scale, highly diverse datasets. MobileNetV2, on the other hand, maintains strong representational power with fewer parameters, allowing it to generalize better on face data.

This balance explains why MobileNetV2 achieves higher accuracy while remaining computationally efficient.

❖ Inverted Residuals and Linear Bottlenecks Preserve Facial Information

MobileNetV2 introduces inverted residual blocks with linear bottlenecks, which play a crucial role in face recognition. Linear bottlenecks prevent information loss, especially in low-dimensional feature spaces. Unlike architectures that apply non-linear activations everywhere, MobileNetV2 preserves essential identity-related features during projection.

As a result, MobileNetV2 retains discriminative facial features more effectively than deeper, more aggressive architectures.

❖ Efficient Feature Extraction with Depthwise Separable Convolutions

MobileNetV2 uses depthwise separable convolutions, which extract spatial features efficiently, reduce redundant computation, and focus learning on meaningful facial patterns instead of noise.

This is well-suited to the VGGFace2 dataset, where faces are already aligned and centered, meaning the model benefits more from efficient local feature extraction than from extremely deep global representations.

Better Generalization Under Hardware and Training Constraints:

MobileNetV2 is designed for low memory access cost, optimized for low-latency and low-power environments, less sensitive to training instability under constrained resources. This allows MobileNetV2 to train more consistently and effectively, leading to better convergence and higher evaluation metrics.

❖ Transfer Learning Works More Effectively with MobileNetV2

MobileNetV2's compact architecture adapts more easily to a new domain, requires fewer parameter updates, and reduces the risk of overfitting compared to larger pretrained networks. This makes MobileNetV2 particularly effective when fine-tuned on face-centric datasets like VGGFace2.

In conclusion, MobileNetV2 performs better for celebrity face classification on the VGGFace2 dataset because it provides an optimal balance between efficiency and representational power. Its inverted residuals and linear bottlenecks preserve critical facial information while avoiding unnecessary complexity. Combined with depthwise separable convolutions and efficient memory usage, MobileNetV2 generalizes better under practical training constraints and adapts more effectively through transfer learning. These architectural advantages directly explain its superior experimental performance compared to VGG-19, ResNet, and Inception V1.

Reference

- ❖ <https://www.sciencedirect.com/topics/computer-science/vgg-19-convolutional-neural-network>
- ❖ <https://www.geeksforgeeks.org/computer-vision/vgg-net-architecture-explained/>
- ❖ <https://arxiv.org/pdf/1812.07857>
- ❖ <https://medium.com/@siddheshb008/resnet-architecture-explained-47309ea9283d>
- ❖ <https://www.geeksforgeeks.org/deep-learning/residual-networks-resnet-deep-learning/>
- ❖ <https://arxiv.org/pdf/1409.4842>
- ❖ https://d1wqtxts1xzle7.cloudfront.net/100487595/pdf.pdf?1680252010=&response-content-disposition=inline%3B+filename%3DMasked_Face_Recognition_Using_MobileNet.pdf&Expires=1765813349&Signature=gDlpStuLimvdP0ZR1RIOOIuuFGrtEYMuOJKXSDD-U3hS~vegb5-PY9nDlaf5j7w9Ti0U6Rsl1VfPKvIefcLDhYcsYCWYlhjDr3-bSCP0Jauvm1ezfIDEtKJ1iEsqw2h9m9csLPP2LLaXKwoSVAY07JctE8ow3yBDKk89IxDuDFBcdz7lsUvOraQRxEiSL1cAaysT~8--t2knYvCPireV8m9e5Vf1YysyeuAvBsnbMQPLZ-9w4MJqUmXmY3bso0OuBngNx-yL9ish1sxW85fCLxDwvixTxPAuolM~dE226y9oZFuAZAd1E1KZ63VTXFaLSDxURr7B43XOJWQInyctBQ__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
- ❖ <https://www.geeksforgeeks.org/computer-vision/mobilenet-v2-architecture-in-computer-vision/>