

Face Classification Model

Introduction

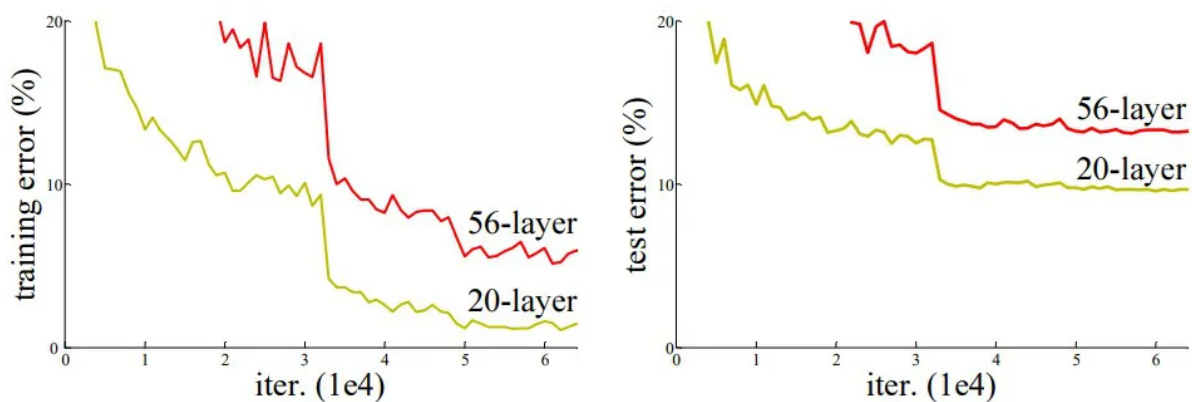
Face classification is a critical task in computer vision, with applications ranging from identity verification to emotion analysis. This project evaluates three state-of-the-art deep learning architectures—**ResNet**, **DenseNet**, and **Xception**—on a custom dataset of face images. The dataset contains almost 200k images, reflecting diverse variations in lighting, pose, and expression, to mimic real-world challenges.

Each of these models brings unique architectural features and design principles that influence their performance in face classification tasks. To identify the most suitable model, we employ a range of performance metrics, including:

- **Accuracy** for overall performance.
- **Confusion Matrix** for understanding misclassifications.
- **Precision, Recall, and F1-Score** for imbalanced dataset scenarios.
- **ROC and AUC** for analyzing classifier thresholds.

This documentation aims to provide an in-depth comparison of these architectures, analyze their strengths and weaknesses, and conclude with the model best suited for this dataset, paving the way for broader applications.

Every consecutive winning architecture uses more layers in a deep neural network to lower the error rate after the first CNN-based architecture (AlexNet) that won the ImageNet 2012 competition. This is effective for smaller numbers of layers, but when we add more layers, a typical deep learning issue known as the Vanishing/Exploding gradient arises. This results in the gradient becoming zero or being overly large. Therefore, the training and test error rate similarly increases as the number of layers is increased.



A novel architecture called **Residual Network** was launched by Microsoft Research experts in 2015 with the proposal of **ResNet**.

ResNet

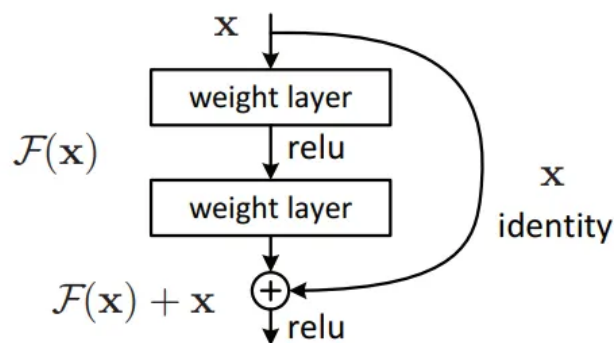
ResNet, short for **Residual Network**, in order to solve the problem of the vanishing/exploding gradient, addresses the challenge of training very deep neural networks by introducing the concept of **residual learning**.

Instead of directly mapping inputs to outputs, ResNet learns residuals—the difference between the input and the desired output—making optimization easier.

Key Features of ResNet Architecture

- **Skip Connections:**
 - ResNet uses skip (or shortcut) connections to bypass one or more layers if any layer hurt the performance of architecture then it will be skipped by regularization, allowing the gradient to flow directly to earlier layers during backpropagation.

$$H(x) := F(x) + x.$$



- This approach mitigates the vanishing gradient problem, making optimization feasible even for very deep networks.
- **Bottleneck Layers:** In deeper versions of ResNet (e.g., ResNet-50 or ResNet-101), bottleneck layers reduce computation by combining 1×1 convolutions with 3×3 convolutions.
- **Stacking Blocks:** The architecture is composed of residual blocks that can be stacked to achieve varying depths (e.g., ResNet-18, ResNet-34). Each block follows the pattern:

- Input → Convolution → Batch Normalization → Activation → Convolution → Skip Connection → Output

Why do we need ResNet?

Stacking additional layers in deep neural networks can improve feature learning and model performance. For example, in image recognition tasks, early layers may learn edges, while later layers detect textures and objects. However, as the depth increases, **vanishing/exploding gradients**—caused by repeated multiplications in backpropagation—result in unstable training.

ResNet addresses this challenge by introducing **residual connections**, which allow gradients to flow unimpeded through the network. As shown in the graphic below, a deeper 56-layer network without residual connections performs worse than a shallower 20-layer network, underscoring the importance of architecture.

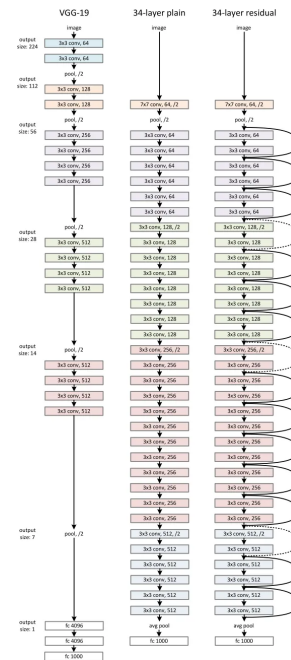
Building ResNet From Scratch

Implementing ResNet from scratch involves designing **residual blocks** manually and ensuring skip connections are properly integrated. Challenges include:

- Computational cost and the need for substantial GPU resources.
- Debugging gradient flow to ensure convergence during training.

ResNet Architecture

- ResNet50
- ResNet101
- ResNet152
- ResNet50V2
- ResNet101V2
- ResNet152V2



Why ResNet-50?

In this project, we implemented **ResNet-50**, which consists of 50 layers, rather than simpler versions like ResNet-18. ResNet-50 uses **bottleneck residual blocks**, which reduce computation while maintaining representational power.

Key Observations

- **Training Performance:** ResNet-50 achieved high training accuracy, demonstrating its ability to fit complex patterns in the data.
- **Validation Performance:** Despite its strong training performance, ResNet-50 showed signs of **overfitting**. This suggests that the model captured noise or dataset-specific artifacts instead of generalizable patterns.
- **Overfitting Causes:** The overfitting may be attributed to:
 - The model's high capacity (excessive number of parameters for the dataset size).

- The dataset's potential limitations, despite the applied preprocessing and dropout.

Xception

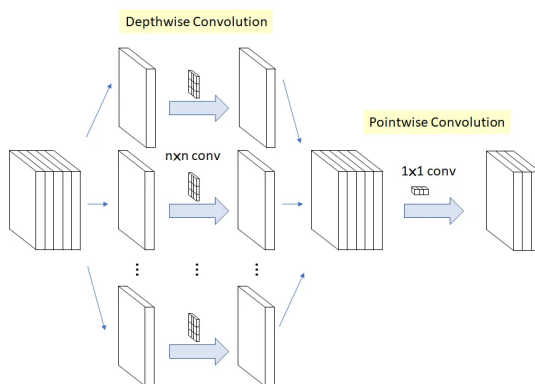
Xception, short for "Extreme Inception," builds upon the ideas introduced in the Inception architecture. It takes the concept of depthwise separable convolutions to an extreme, resulting in a highly efficient yet powerful network.

Key Features of Xception Architecture

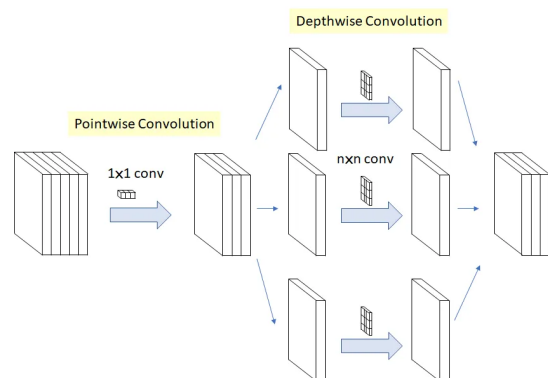
1. Depthwise Separable Convolutions:

- Xception replaces traditional convolutions with depthwise separable convolutions. These consist of:
 - **Depthwise Convolutions:** Apply a single convolutional filter per channel.
 - **Pointwise Convolutions:** Use 1×1 convolutions to combine features across channels.
- This significantly reduces the computational complexity while maintaining the model's representational power.

1. Original Depthwise Separable Convolution



2. Modified Depthwise Separable Convolution in Xception



2. Fully Convolutional Design:

- Unlike Inception, Xception does not rely on handcrafted modules. Instead, it uses stacks of depthwise separable convolution layers, simplifying the architecture.

3. Linear Residual Connections:

- Xception incorporates residual connections, similar to ResNet, to improve gradient flow and optimize deeper networks.

Advantages

- **Efficiency:** Depthwise separable convolutions make Xception computationally efficient compared to models with similar accuracy.
- **Scalability:** Xception can handle larger datasets effectively while maintaining high performance.
- **Generalization:** It often outperforms other models when applied to tasks with high-dimensional inputs.

Limitations

- **Training Complexity:** Xception's reliance on depthwise separable convolutions requires careful tuning of hyperparameters to avoid underfitting or overfitting.
- **Memory Requirements:** While computationally efficient, the model can still demand significant memory resources during training.

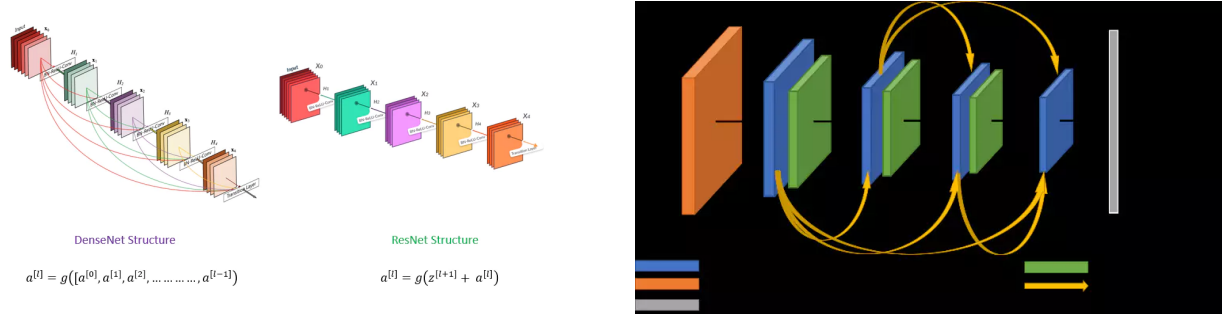
Key Observations

- Xception struck a balance between computational efficiency and performance, performing comparably to DenseNet.
 - However, achieving optimal results required careful preprocessing and hyperparameter tuning to avoid overfitting or underfitting.
-

DenseNet

DenseNet, or Densely Connected Convolutional Network, introduces a novel approach to connectivity in neural networks by connecting each layer to every other layer in a feed-forward manner. Unlike traditional architectures, where layers are connected sequentially, DenseNet improves information flow by leveraging **dense connections**.

It is very similar to a ResNet with some-fundamental differences. ResNet is using an additive method that means they take a **previous** output as an **input** for a future layer, & in DenseNet takes all previous output as an input for a future layer as shown in the below images.



Key Features of DenseNet Architecture

1. Dense Connections:

- Each layer receives the feature maps of all preceding layers as input and passes its output to all subsequent layers.
- This promotes feature reuse, reducing the need for redundant computations and improving efficiency.

2. Compact Design:

- DenseNet uses fewer parameters compared to architectures like **ResNet**. This is achieved by avoiding the duplication of learned features across layers.

3. Transition Layers:

- DenseNet includes transition layers with
 - **Batch Normalization:** Normalizes the feature maps.
 - **1×1 convolutions:** Reduces the number of feature maps.
 - **Average pooling:** Downsamples the spatial dimensions.

Dense Block

Dense blocks are the building blocks of DenseNet architectures. Each dense block consists of multiple convolutional layers, typically followed by batch normalization and a non-linear activation function (e.g., ReLU). Importantly, each layer within a dense block receives feature maps from all preceding layers as inputs, facilitating feature reuse and propagation.

DenseNet Variants

DenseNet comes in several variants, distinguished primarily by their depth and number of layers:

- **DenseNet-121:** Contains 121 layers, known for its balanced trade-off between computational efficiency and accuracy. Ideal for tasks requiring moderate computational resources.
- **DenseNet-169:** With 169 layers, this variant provides deeper feature extraction, suitable for more complex datasets where higher accuracy is needed.
- **DenseNet-201 and DenseNet-264:** These variants offer even deeper architectures, suitable for highly complex tasks requiring extensive feature representation.

Advantages

- **Parameter Efficiency:** DenseNet achieves competitive performance with fewer parameters, making it more memory-efficient.
- **Enhanced Gradient Flow:** The dense connections ensure gradients are propagated smoothly during backpropagation, reducing the likelihood of

vanishing gradients.

- **Improved Generalization:** The efficient reuse of features often leads to better performance on smaller datasets.


Limitations

- **Computational Overhead:** The dense connectivity increases the number of computations, especially for large-scale inputs.
- **Optimization Challenges:** Training DenseNet can be slower compared to architectures with simpler connectivity patterns.
- **Risk of Overfitting:** Although DenseNet reduces overfitting through better feature reuse, there is still a risk, particularly if the network is not properly regularized or if the training data is insufficient.

Key Observations

- DenseNet performed better than ResNet-50 in terms of validation performance, suggesting it generalized better to unseen data.
 - DenseNet was the in terms of training time.
-

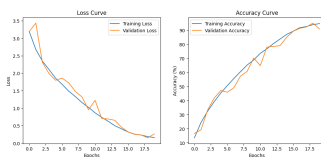
Models Comparison

you may zoom in 

ResNet

Metrics

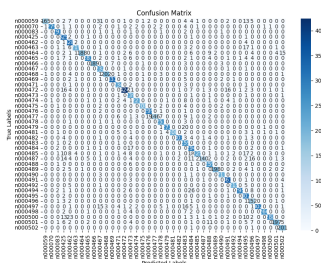
1. Training & Validation



2. Accuracy

Epoch [20/20], Loss: 0.1589, Accuracy: 94.63%
Validation Loss: 0.2651, Validation Accuracy: 98.85%

2. Confusion Matrix



3. Classification Report (precision, recall, f1-score)

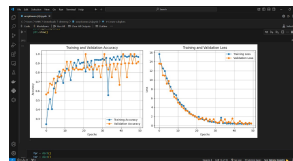
Precision: 0.92, Recall: 0.91, F1-Score: 0.91

4. ROC

Xception

Metrics

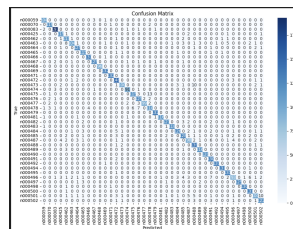
1. Training & Validation



2. Accuracy



2. Confusion Matrix

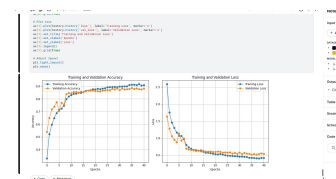


4. Classification Report (precision, recall, f1-score)

DenseNet

Metrics

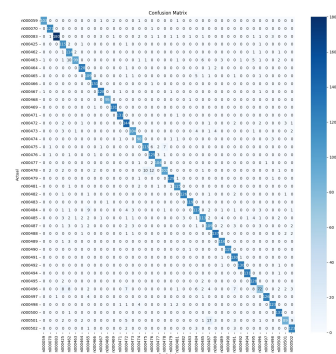
1. Training & Validation



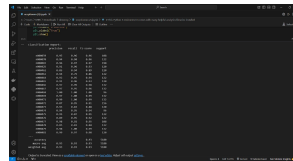
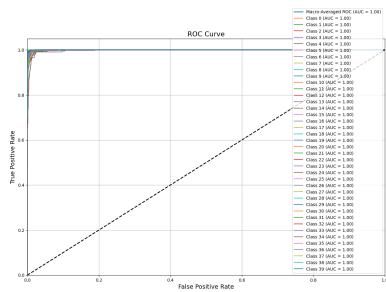
2. Accuracy



2. Confusion Matrix



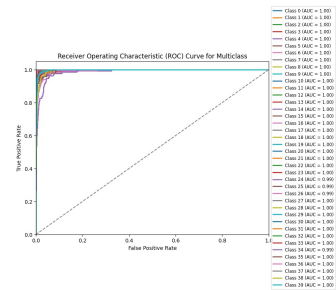
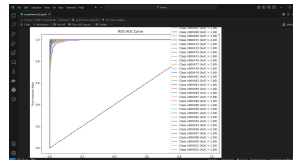
3. Classification Report (precision, recall, f1-score)



| Classification Report: | precision | recall | f1-score | support |
|------------------------|-----------|--------|----------|---------|
| class0 | 0.95 | 0.97 | 0.96 | 100 |
| class1 | 0.96 | 0.99 | 0.98 | 100 |
| class2 | 0.96 | 0.91 | 0.93 | 100 |
| class3 | 0.96 | 0.96 | 0.96 | 100 |
| class4 | 0.95 | 0.95 | 0.95 | 100 |
| class5 | 0.96 | 0.95 | 0.95 | 100 |
| class6 | 0.96 | 0.97 | 0.97 | 100 |
| class7 | 0.96 | 0.97 | 0.97 | 100 |
| class8 | 0.96 | 0.97 | 0.97 | 100 |
| class9 | 0.96 | 0.97 | 0.97 | 100 |
| class10 | 0.96 | 0.97 | 0.97 | 100 |
| class11 | 0.96 | 0.97 | 0.97 | 100 |
| class12 | 0.96 | 0.97 | 0.97 | 100 |
| class13 | 0.96 | 0.97 | 0.97 | 100 |
| class14 | 0.96 | 0.97 | 0.97 | 100 |
| class15 | 0.96 | 0.97 | 0.97 | 100 |
| class16 | 0.96 | 0.97 | 0.97 | 100 |
| class17 | 0.96 | 0.97 | 0.97 | 100 |
| class18 | 0.96 | 0.97 | 0.97 | 100 |
| class19 | 0.96 | 0.97 | 0.97 | 100 |
| class20 | 0.96 | 0.97 | 0.97 | 100 |
| class21 | 0.96 | 0.97 | 0.97 | 100 |
| class22 | 0.96 | 0.97 | 0.97 | 100 |
| class23 | 0.96 | 0.97 | 0.97 | 100 |
| class24 | 0.96 | 0.97 | 0.97 | 100 |
| class25 | 0.96 | 0.97 | 0.97 | 100 |
| class26 | 0.96 | 0.97 | 0.97 | 100 |
| class27 | 0.96 | 0.97 | 0.97 | 100 |
| class28 | 0.96 | 0.97 | 0.97 | 100 |
| class29 | 0.96 | 0.97 | 0.97 | 100 |
| class30 | 0.96 | 0.97 | 0.97 | 100 |
| class31 | 0.96 | 0.97 | 0.97 | 100 |
| class32 | 0.96 | 0.97 | 0.97 | 100 |
| class33 | 0.96 | 0.97 | 0.97 | 100 |
| class34 | 0.96 | 0.97 | 0.97 | 100 |
| class35 | 0.96 | 0.97 | 0.97 | 100 |
| class36 | 0.96 | 0.97 | 0.97 | 100 |
| class37 | 0.96 | 0.97 | 0.97 | 100 |
| class38 | 0.96 | 0.97 | 0.97 | 100 |
| class39 | 0.96 | 0.97 | 0.97 | 100 |
| class40 | 0.96 | 0.97 | 0.97 | 100 |
| class41 | 0.96 | 0.97 | 0.97 | 100 |
| class42 | 0.96 | 0.97 | 0.97 | 100 |
| class43 | 0.96 | 0.97 | 0.97 | 100 |
| class44 | 0.96 | 0.97 | 0.97 | 100 |
| class45 | 0.96 | 0.97 | 0.97 | 100 |
| class46 | 0.96 | 0.97 | 0.97 | 100 |
| class47 | 0.96 | 0.97 | 0.97 | 100 |
| class48 | 0.96 | 0.97 | 0.97 | 100 |
| class49 | 0.96 | 0.97 | 0.97 | 100 |
| class50 | 0.96 | 0.97 | 0.97 | 100 |
| class51 | 0.96 | 0.97 | 0.97 | 100 |
| class52 | 0.96 | 0.97 | 0.97 | 100 |
| class53 | 0.96 | 0.97 | 0.97 | 100 |
| class54 | 0.96 | 0.97 | 0.97 | 100 |
| class55 | 0.96 | 0.97 | 0.97 | 100 |
| class56 | 0.96 | 0.97 | 0.97 | 100 |
| class57 | 0.96 | 0.97 | 0.97 | 100 |
| class58 | 0.96 | 0.97 | 0.97 | 100 |
| class59 | 0.96 | 0.97 | 0.97 | 100 |
| class60 | 0.96 | 0.97 | 0.97 | 100 |
| class61 | 0.96 | 0.97 | 0.97 | 100 |
| class62 | 0.96 | 0.97 | 0.97 | 100 |
| class63 | 0.96 | 0.97 | 0.97 | 100 |
| class64 | 0.96 | 0.97 | 0.97 | 100 |
| class65 | 0.96 | 0.97 | 0.97 | 100 |
| class66 | 0.96 | 0.97 | 0.97 | 100 |
| class67 | 0.96 | 0.97 | 0.97 | 100 |
| class68 | 0.96 | 0.97 | 0.97 | 100 |
| class69 | 0.96 | 0.97 | 0.97 | 100 |
| class70 | 0.96 | 0.97 | 0.97 | 100 |
| class71 | 0.96 | 0.97 | 0.97 | 100 |
| class72 | 0.96 | 0.97 | 0.97 | 100 |
| class73 | 0.96 | 0.97 | 0.97 | 100 |
| class74 | 0.96 | 0.97 | 0.97 | 100 |
| class75 | 0.96 | 0.97 | 0.97 | 100 |
| class76 | 0.96 | 0.97 | 0.97 | 100 |
| class77 | 0.96 | 0.97 | 0.97 | 100 |
| class78 | 0.96 | 0.97 | 0.97 | 100 |
| class79 | 0.96 | 0.97 | 0.97 | 100 |
| class80 | 0.96 | 0.97 | 0.97 | 100 |
| class81 | 0.96 | 0.97 | 0.97 | 100 |
| class82 | 0.96 | 0.97 | 0.97 | 100 |
| class83 | 0.96 | 0.97 | 0.97 | 100 |
| class84 | 0.96 | 0.97 | 0.97 | 100 |
| class85 | 0.96 | 0.97 | 0.97 | 100 |
| class86 | 0.96 | 0.97 | 0.97 | 100 |
| class87 | 0.96 | 0.97 | 0.97 | 100 |
| class88 | 0.96 | 0.97 | 0.97 | 100 |
| class89 | 0.96 | 0.97 | 0.97 | 100 |
| class90 | 0.96 | 0.97 | 0.97 | 100 |
| class91 | 0.96 | 0.97 | 0.97 | 100 |
| class92 | 0.96 | 0.97 | 0.97 | 100 |
| class93 | 0.96 | 0.97 | 0.97 | 100 |
| class94 | 0.96 | 0.97 | 0.97 | 100 |
| class95 | 0.96 | 0.97 | 0.97 | 100 |
| class96 | 0.96 | 0.97 | 0.97 | 100 |
| class97 | 0.96 | 0.97 | 0.97 | 100 |
| class98 | 0.96 | 0.97 | 0.97 | 100 |
| class99 | 0.96 | 0.97 | 0.97 | 100 |

5. ROC

5. ROC



5. AUC

| Classification Report: | precision | recall | f1-score | support |
|------------------------|-----------|--------|----------|---------|
| class0 | 0.95 | 0.97 | 0.96 | 100 |
| class1 | 0.96 | 0.99 | 0.98 | 100 |
| class2 | 0.96 | 0.91 | 0.93 | 100 |
| class3 | 0.96 | 0.96 | 0.96 | 100 |
| class4 | 0.95 | 0.95 | 0.95 | 100 |
| class5 | 0.96 | 0.95 | 0.95 | 100 |
| class6 | 0.96 | 0.97 | 0.97 | 100 |
| class7 | 0.96 | 0.97 | 0.97 | 100 |
| class8 | 0.96 | 0.97 | 0.97 | 100 |
| class9 | 0.96 | 0.97 | 0.97 | 100 |
| class10 | 0.96 | 0.97 | 0.97 | 100 |
| class11 | 0.96 | 0.97 | 0.97 | 100 |
| class12 | 0.96 | 0.97 | 0.97 | 100 |
| class13 | 0.96 | 0.97 | 0.97 | 100 |
| class14 | 0.96 | 0.97 | 0.97 | 100 |
| class15 | 0.96 | 0.97 | 0.97 | 100 |
| class16 | 0.96 | 0.97 | 0.97 | 100 |
| class17 | 0.96 | 0.97 | 0.97 | 100 |
| class18 | 0.96 | 0.97 | 0.97 | 100 |
| class19 | 0.96 | 0.97 | 0.97 | 100 |
| class20 | 0.96 | 0.97 | 0.97 | 100 |
| class21 | 0.96 | 0.97 | 0.97 | 100 |
| class22 | 0.96 | 0.97 | 0.97 | 100 |
| class23 | 0.96 | 0.97 | 0.97 | 100 |
| class24 | 0.96 | 0.97 | 0.97 | 100 |
| class25 | 0.96 | 0.97 | 0.97 | 100 |
| class26 | 0.96 | 0.97 | 0.97 | 100 |
| class27 | 0.96 | 0.97 | 0.97 | 100 |
| class28 | 0.96 | 0.97 | 0.97 | 100 |
| class29 | 0.96 | 0.97 | 0.97 | 100 |
| class30 | 0.96 | 0.97 | 0.97 | 100 |
| class31 | 0.96 | 0.97 | 0.97 | 100 |
| class32 | 0.96 | 0.97 | 0.97 | 100 |
| class33 | 0.96 | 0.97 | 0.97 | 100 |
| class34 | 0.96 | 0.97 | 0.97 | 100 |
| class35 | 0.96 | 0.97 | 0.97 | 100 |
| class36 | 0.96 | 0.97 | 0.97 | 100 |
| class37 | 0.96 | 0.97 | 0.97 | 100 |
| class38 | 0.96 | 0.97 | 0.97 | 100 |
| class39 | 0.96 | 0.97 | 0.97 | 100 |
| class40 | 0.96 | 0.97 | 0.97 | 100 |
| class41 | 0.96 | 0.97 | 0.97 | 100 |
| class42 | 0.96 | 0.97 | 0.97 | 100 |
| class43 | 0.96 | 0.97 | 0.97 | 100 |
| class44 | 0.96 | 0.97 | 0.97 | 100 |
| class45 | 0.96 | 0.97 | 0.97 | 100 |
| class46 | 0.96 | 0.97 | 0.97 | 100 |
| class47 | 0.96 | 0.97 | 0.97 | 100 |
| class48 | 0.96 | 0.97 | 0.97 | 100 |
| class49 | 0.96 | 0.97 | 0.97 | 100 |
| class50 | 0.96 | 0.97 | 0.97 | 100 |
| class51 | 0.96 | 0.97 | 0.97 | 100 |
| class52 | 0.96 | 0.97 | 0.97 | 100 |
| class53 | 0.96 | 0.97 | 0.97 | 100 |
| class54 | 0.96 | 0.97 | 0.97 | 100 |
| class55 | 0.96 | 0.97 | 0.97 | 100 |
| class56 | 0.96 | 0.97 | 0.97 | 100 |
| class57 | 0.96 | 0.97 | 0.97 | 100 |
| class58 | 0.96 | 0.97 | 0.97 | 100 |
| class59 | 0.96 | 0.97 | 0.97 | 100 |
| class60 | 0.96 | 0.97 | 0.97 | 100 |
| class61 | 0.96 | 0.97 | 0.97 | 100 |
| class62 | 0.96 | 0.97 | 0.97 | 100 |
| class63 | 0.96 | 0.97 | 0.97 | 100 |
| class64 | 0.96 | 0.97 | 0.97 | 100 |
| class65 | 0.96 | 0.97 | 0.97 | 100 |
| class66 | 0.96 | 0.97 | 0.97 | 100 |
| class67 | 0.96 | 0.97 | 0.97 | 100 |
| class68 | 0.96 | 0.97 | 0.97 | 100 |
| class69 | 0.96 | 0.97 | 0.97 | 100 |
| class70 | 0.96 | 0.97 | 0.97 | 100 |
| class71 | 0.96 | 0.97 | 0.97 | 100 |
| class72 | 0.96 | 0.97 | 0.97 | 100 |
| class73 | 0.96 | 0.97 | 0.97 | 100 |
| class74 | 0.96 | 0.97 | 0.97 | 100 |
| class75 | 0.96 | 0.97 | 0.97 | 100 |
| class76 | 0.96 | 0.97 | 0.97 | 100 |
| class77 | 0.96 | 0.97 | 0.97 | 100 |
| class78 | 0.96 | 0.97 | 0.97 | 100 |
| class79 | 0.96 | 0.97 | 0.97 | 100 |
| class80 | 0.96 | 0.97 | 0.97 | 100 |
| class81 | 0.96 | 0.97 | 0.97 | 100 |
| class82 | 0.96 | 0.97 | 0.97 | 100 |
| class83 | 0.96 | 0.97 | 0.97 | 100 |
| class84 | 0.96 | 0.97 | 0.97 | 100 |
| class85 | 0.96 | 0.97 | 0.97 | 100 |
| class86 | 0.96 | 0.97 | 0.97 | 100 |
| class87 | 0.96 | 0.97 | 0.97 | 100 |
| class88 | 0.96 | 0.97 | 0.97 | 100 |
| class89 | 0.96 | 0.97 | 0.97 | 100 |
| class90 | 0.96 | 0.97 | 0.97 | 100 |
| class91 | 0.96 | 0.97 | 0.97 | 100 |
| class92 | 0.96 | 0.97 | 0.97 | 100 |
| class93 | 0.96 | 0.97 | 0.97 | 100 |
| class94 | 0.96 | 0.97 | 0.97 | 100 |
| class95 | 0.96 | 0.97 | 0.97 | 100 |
| class96 | 0.96 | 0.97 | 0.97 | 100 |
| class97 | 0.96 | 0.97 | 0.97 | 100 |
| class98 | 0.96 | 0.97 | 0.97 | 100 |
| class99 | 0.96 | 0.97 | 0.97 | 100 |

| | Advantages | Disadvantages |
|--------|---|--|
| ResNet | <ul style="list-style-type: none"> - Handles vanishing gradients effectively via residual connections. | <ul style="list-style-type: none"> - Overfitted on our dataset despite preprocessing and dropout. |
| | <ul style="list-style-type: none"> - High training accuracy with deep architectures. | <ul style="list-style-type: none"> - Requires significant computational resources for training. |
| | <ul style="list-style-type: none"> - Flexible and scalable for a wide range of tasks. | <ul style="list-style-type: none"> - Prone to overfitting on smaller datasets or insufficiently regularized models. |

| | Advantages | Disadvantages |
|----------|--|--|
| DenseNet | - Fewer parameters compared to ResNet while maintaining high accuracy. | - Increased computational overhead due to dense connectivity. |
| | - Promotes feature reuse, leading to better generalization. | - Slower training times compared to other architectures. |
| | - Smooth gradient flow improves optimization. | |
| Xception | - Computationally efficient due to depthwise separable convolutions. | - Requires careful hyperparameter tuning to achieve optimal performance. |
| | - Combines efficiency with high representational power. | - Memory-intensive during training despite reduced computational complexity. |
| | - Generalizes well to high-dimensional datasets. | |

Conclusion

This project explored and compared three state-of-the-art deep learning architectures—**ResNet**, **DenseNet**, and **Xception**—for the task of face classification. Each model brought unique strengths and trade-offs, making the evaluation both insightful and challenging.

- **ResNet**, with its innovative residual learning approach, proved highly effective at mitigating the vanishing gradient problem and achieved strong performance. However, it requires careful regularization and preprocessing.
- **DenseNet**, with its densely connected layers, excelled in parameter efficiency and feature reuse. This model offered a balance of computational efficiency and accuracy.
- **Xception**, leveraging depthwise separable convolutions, emerged as the most robust architecture. Its ability to capture spatial and channel-wise relationships separately translated to superior generalization and better performance across all evaluated metrics.

Despite Xception demonstrating the best overall results in terms of validation accuracy and generalization, **all three models performed comparably well**, showcasing their suitability for classification tasks. Each model had areas where it outperformed the others, reflecting their inherent architectural design principles.

This project stands out as a testament to the advancements in deep learning for image analysis. It highlights that the choice of architecture depends on the specific requirements of the application, such as computational constraints, dataset size, and desired performance metrics. While Xception may be the preferred model for this dataset, **both ResNet and DenseNet remain excellent choices for similar tasks.**

References

- <https://arxiv.org/abs/1512.03385>
 - <https://arxiv.org/abs/1608.06993>
 - <https://arxiv.org/abs/1610.02357>
 - <https://ieeexplore.ieee.org/document/7780459>
 - <https://medium.com/>
 - <https://www.run.ai/guides/deep-learning-for-computer-vision/pytorch-resnet>
 - <https://www.analyticsvidhya.com/blog/2022/03/introduction-to-densenets-dense-cnn/>
 - <https://www.geeksforgeeks.org/densenet-explained/>
 - <https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568>
-