**Comparative Analysis of ResNet-18, Xception, and DenseNet-121 Architectures**

**1. Overview of Architectures**

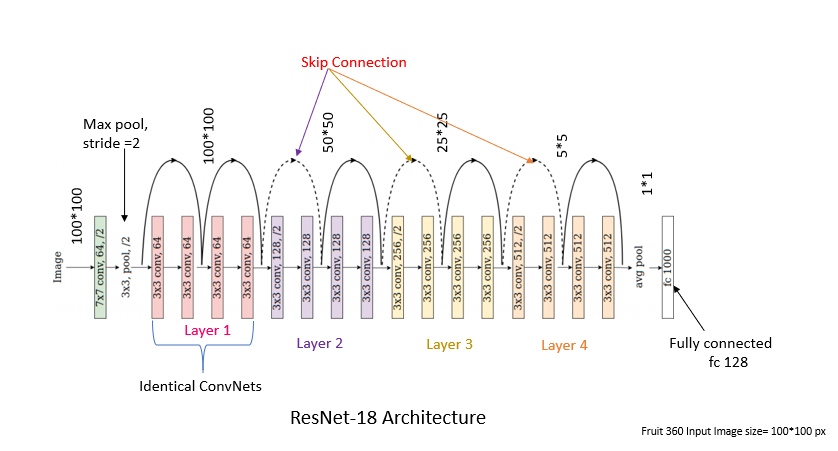
**ResNet-18 (Residual Network)**

**Structure:**

* Introduced in *"Deep Residual Learning for Image Recognition" (He et al., 2016)*.
* Solves the vanishing gradient problem by introducing **shortcut connections** (identity mappings).
* Basic building block: **Residual Block**.
  + Each block includes two convolutional layers.
  + A shortcut connection skips these layers and adds the input directly to the output.

**Key Components:**

* **Identity Shortcut:** Allows the gradient to flow directly to earlier layers.
* **Residual Learning:** Encourages layers to learn the residual mapping instead of full mapping.
* **Stacked Blocks:**
  + Conv1: 7x7 convolution + MaxPooling
  + Conv2\_x: 2 residual blocks with 64 filters.
  + Conv3\_x: 2 residual blocks with 128 filters.
  + Conv4\_x: 2 residual blocks with 256 filters.
  + Conv5\_x: 2 residual blocks with 512 filters.
* Ends with global average pooling and a fully connected layer.



**Pros:**

* Efficient for tasks with limited computational resources.
* Easier optimization due to residual connections.
* Good performance on small datasets.

**Cons:**

* Underperforms on very large-scale tasks compared to deeper models.

**Reference:**

[Deep Residual Learning for Image Recognition (He et al., 2016)](https://arxiv.org/pdf/1512.03385.pdf)

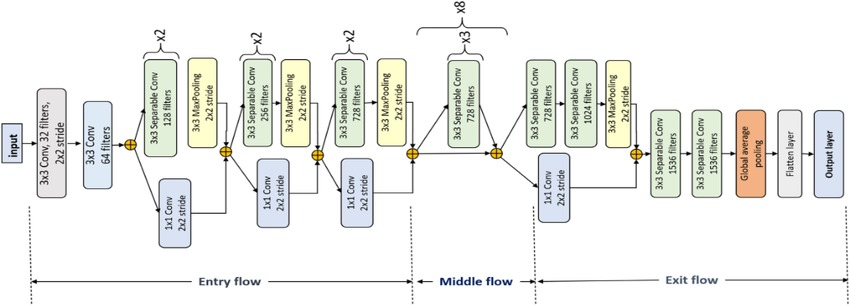
**Xception (Extreme Inception)**

**Structure:**

* Introduced in *"Xception: Deep Learning with Depthwise Separable Convolutions" (Chollet, 2017)*.
* Builds on **Inception architecture** but replaces inception modules with **depthwise separable convolutions**.

**Key Components:**

1. **Depthwise Separable Convolutions:**
   * Depthwise Convolution: Operates channel-wise spatial convolutions.
   * Pointwise Convolution: Uses 1x1 convolutions to combine channel information.
2. **Flow Structure:**
   * **Entry Flow:** Reduces spatial dimensions and learns basic features.
   * **Middle Flow:** Stacks multiple depthwise separable convolution layers to capture intermediate features.
   * **Exit Flow:** Extracts complex features and reduces dimensionality for classification.
   * Ends with global average pooling and a dense softmax layer.



**Pros:**

* Efficient computation with depthwise separable convolutions.
* Excellent feature extraction for complex datasets.
* Superior performance on image classification tasks compared to ResNet.

**Cons:**

* Requires more memory compared to ResNet.
* Computationally expensive for small datasets.

**Reference:**

[Xception: Deep Learning with Depthwise Separable Convolutions (Chollet, 2017)](https://arxiv.org/pdf/1610.02357.pdf)

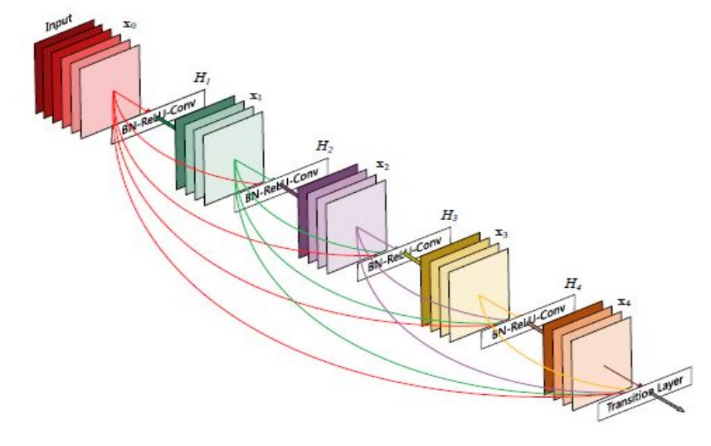
**DenseNet-121 (Densely Connected Convolutional Networks)**

**Structure:**

* Introduced in *"Densely Connected Convolutional Networks" (Huang et al., 2017)*.
* Each layer receives inputs from **all preceding layers**, promoting feature reuse.

**Key Components:**

1. **Dense Blocks:**
   * Each layer is connected to every other layer in a feed-forward fashion.
   * Includes batch normalization, ReLU activation, and 3x3 convolutions.
2. **Transition Layers:**
   * Compress the feature maps with 1x1 convolutions and pooling.
3. **Structure:**
   * Conv1: 7x7 convolution + MaxPooling.
   * Dense Block 1 (6 layers).
   * Transition Layer 1.
   * Dense Block 2 (12 layers).
   * Transition Layer 2.
   * Dense Block 3 (24 layers).
   * Transition Layer 3.
   * Dense Block 4 (16 layers).
   * Ends with global average pooling and a fully connected layer.



**Pros:**

* Highly efficient feature reuse reduces overfitting.
* Fewer parameters compared to other architectures with similar depth.
* Superior performance on large and complex datasets.

**Cons:**

* Computationally expensive compared to ResNet.
* Slower inference time on low-resource systems.

**Reference:**

[Densely Connected Convolutional Networks (Huang et al., 2017)](https://arxiv.org/pdf/1608.06993.pdf)

**2. Performance Comparison**

**Dataset Details:**

* Dataset: **FaceScrub** (50 classes, 4605 training images, 1125 testing images).

**Results:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Metric** | **Xception** | **ResNet-18** | **DenseNet-121** |
| Accuracy | 66% | 7.00% | 65.81% |
| Precision | 0.79 | 0.06 | 0.79 |
| Recall | 0.66 | 0.08 | 0.67 |
| F1-Score | 0.66 | 0.04 | 0.68 |

**Accuracy and loss curves :**

**For Xception :  
A graph of different levels of accuracy

Description automatically generated with medium confidence**

**For ResNet:  
A comparison of a graph

Description automatically generated with medium confidence**

**For DenseNet :**

**A graph of loss curve and accuracy

Description automatically generated**

**Confusion matrix:**

**For xception :**

**A graph of blue squares

Description automatically generated**

**For Resnet :A screenshot of a computer screen

Description automatically generated**

**For Densenet :**

**A graph with blue squares

Description automatically generated**

**Avg Roc\_curve:**

**For xception :**

**A screen shot of a graph

Description automatically generated**

**For resnet :** **A graph of a graph

Description automatically generated with medium confidence**

**For densenet:**

**A graph with numbers and lines

Description automatically generated with medium confidence**

**Observations:**

1. **ResNet-18:** Struggles with performance due to the dataset complexity (50 classes) and relatively shallow depth.
2. **Xception:** Balances computational efficiency and accuracy due to depthwise separable convolutions.
3. **DenseNet-121:** Slightly outperforms Xception due to feature reuse and better gradient flow.

**Confusion Matrix Comparison:**

* **Xception:** Shows better diagonal dominance, reflecting higher accuracy.
* **ResNet-18:** Sparse diagonal, indicating poor classification performance.
* **DenseNet-121:** Well-distributed confusion matrix, demonstrating balanced class predictions.

1. **Pros and Cons**

|  |  |  |
| --- | --- | --- |
| Architecture | Pros | Cons |
| ResNet-18 | - Simple structure, easy to implement.  - Low computational cost. | - Poor performance on complex datasets.  - Requires deeper versions (e.g., ResNet-50) for better accuracy. |
| Xception | - Efficient with depthwise separable convolutions.  - Strong performance on large datasets. | - Memory intensive.  - Overkill for simple tasks. |
| DenseNet-121 | - Efficient feature reuse.  - Fewer parameters despite depth. | - Computationally expensive compared to ResNet.  - Slower inference on resource-limited systems. |

**4. Task-Specific Suitability**

1. **Small Datasets:**
   * Use **ResNet-18** for its simplicity and lower computational requirements.
2. **Medium to Large Datasets:**
   * **Xception** and **DenseNet-121** outperform ResNet in terms of accuracy and efficiency.
3. **Highly Complex Datasets (Many Classes):**
   * **DenseNet-121** is preferred due to its feature reuse and gradient flow capabilities.

**References:**

1. [Deep Residual Learning for Image Recognition (He et al., 2016)](https://arxiv.org/pdf/1512.03385.pdf)
2. [Xception: Deep Learning with Depthwise Separable Convolutions (Chollet, 2017)](https://arxiv.org/pdf/1610.02357.pdf)
3. [Densely Connected Convolutional Networks (Huang et al., 2017)](https://openaccess.thecvf.com/content_cvpr_2017/papers/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.pdf)