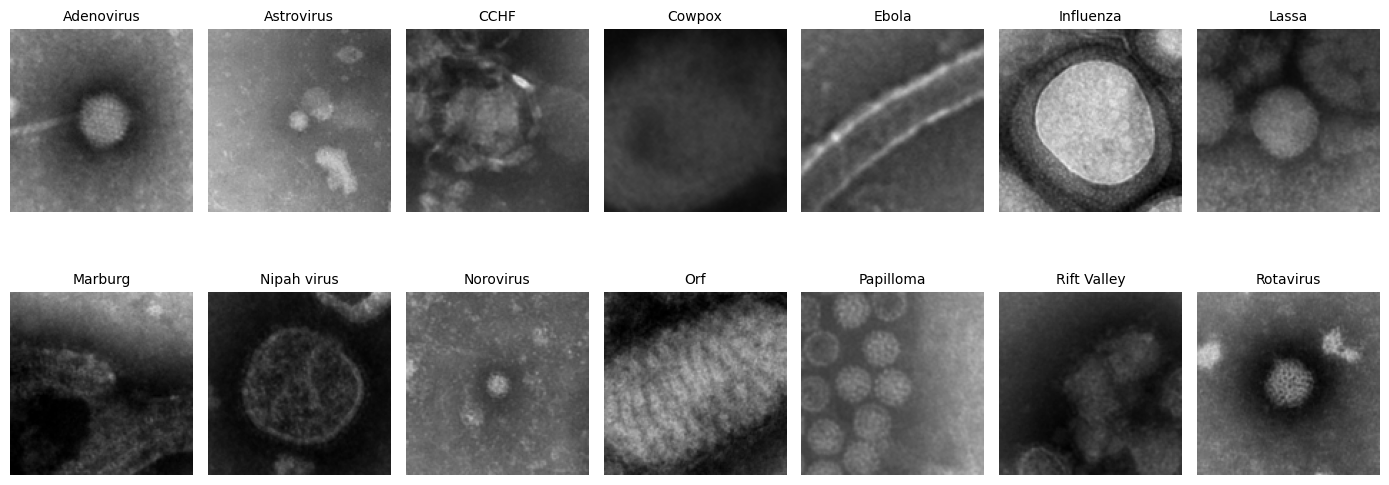
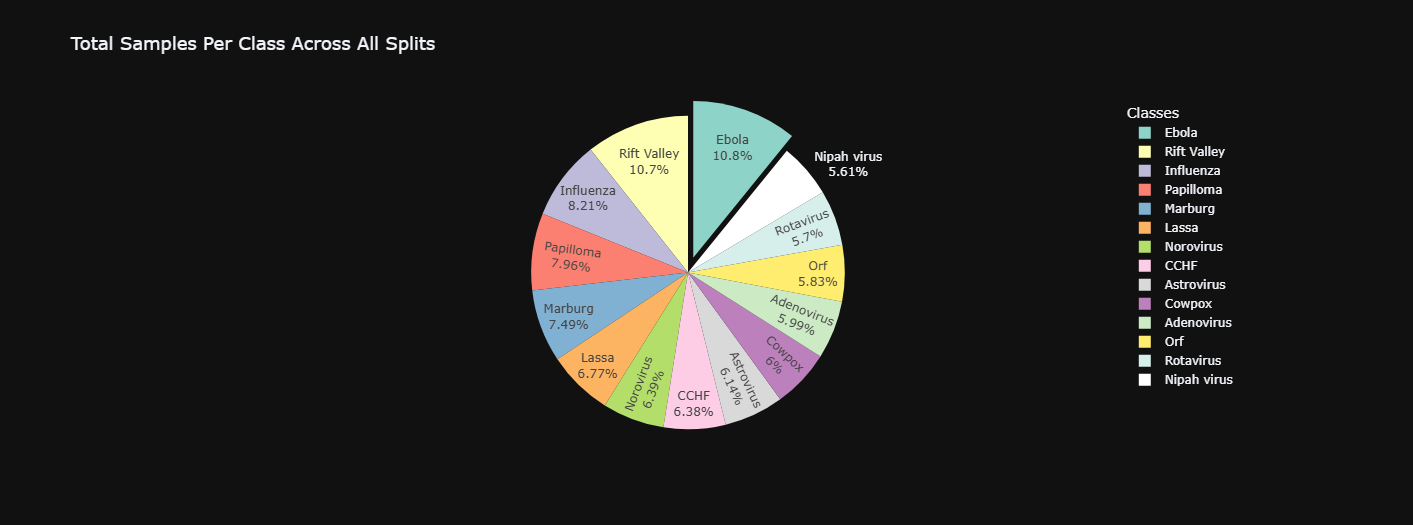
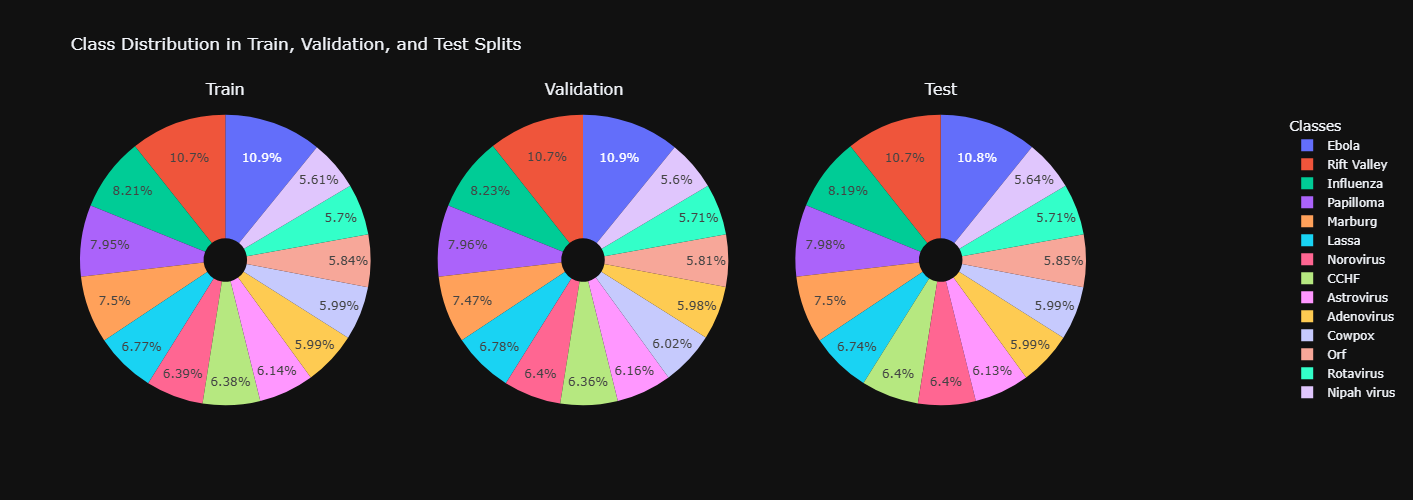
**DataSet**

The dataset appears to consist of electron microscopy images of viruses, with labeled samples for different virus types. Based on the provided image, the dataset includes 12 distinct classes of viruses:

**Virus Types:**

1. **Adenovirus:** Spherical and structured virus with a detailed capsid. **(866 Sample)**
2. **Astrovirus:** Small, roundish virus with low contrast and faint features. **(888 Sample)**
3. **CCHF (Crimean-Congo Hemorrhagic Fever):** A virus with an irregular shape and less distinct structural features. **(922 Sample)**
4. **Cowpox**: A blurry circular structure likely representing a poxvirus. **(867 Sample)**
5. **Ebola**: Rod-shaped and filamentous structure. **(1568 Sample)**
6. **Influenza**: A virus with a spherical shape surrounded by a visible envelope. **(1187 Sample)**
7. **Lassa:** Circular viruses with a relatively uniform appearance. **(978 Sample)**
8. **Marburg**: Filamentous like Ebola but with subtle differences in structure. **(1083 Sample)**
9. **Nipah Virus:** Spherical structure, slightly diffuse in the image. **(811 Sample)**
10. **Norovirus**: Small, round particles with minimal structural details visible. **(924 Sample)**
11. **Orf**: Elongated or cylindrical, with a textured surface resembling poxviruses. **(843 Sample)**
12. **Papilloma**: Spherical virus with a very distinct capsid pattern. **(1150 Sample)**
13. **Rift Valley**: Circular or indistinct particles, low contrast. **(1542 Sample)**
14. **Rotavirus**: Spherical virus with a well-defined, patterned capsid**. (824 Sample)**

**Description of the Dataset:**

* **Type of Data:** Electron microscopy grayscale images of viruses.
* **Number of Classes:** 14 virus categories, each representing a distinct viral type.
* **Potential Applications:**
  + Medical Research: Identifying and distinguishing virus types.
  + Machine Learning: Training classification or detection models for viral analysis.
  + Diagnostics: Assisting in automated or semi-automated diagnostic tools.

**Challenges:**

1. **Image Quality**: The images are grayscale with varying degrees of clarity and contrast.
2. **Class Overlap**: Some viruses may look similar (e.g., Marburg vs. Ebola), posing classification challenges.
3. **Dataset Diversity**: The dataset should ideally include variations in scale, orientation, and imaging conditions.

This dataset is suitable for tasks like image classification, object detection, and biomedical research using deep learning techniques. It could be utilized to develop models for viral identification in medical diagnostics.

**ResNet**

**Introduction**

**ResNet-18**, short for "Residual Network with 18 layers," addresses the vanishing gradient problem in deep neural networks using residual connections.

**Components:**

* **Residual Block:** Core building block featuring skip connections.
* **Convolutional Layers**: Extract features via kernels.
* **Batch Normalization (BN):** Stabilizes learning by normalizing inputs.
* **ReLU Activation:** Introduces non-linearity.
* **Pooling Layers:** Down-sample feature maps.
* **Fully Connected Layer:** Outputs the final class scores.

**Architecture**

**Stage 1: Initial Layers**

1. **Conv1**
   * 64 filters, 7×7 kernel, stride 2, padding 3.
   * Input: 128×128×3 → Output: 64×64×64.
2. **Max Pooling**
   * 3×3 kernel, stride 2, padding 1 → Output: 32×32×64.

**Stage 2: Residual Block with 64 Filters**

* Two blocks with two 3×3 convolutions, 64 filters, stride 1, padding 1, and skip connections.
* Output: 32×32×64.

***... (Subsequent stages would continue similarly with concise details for all stages.)***

**Final Layers**

1. **Global Average Pooling:** Reduces 4×4×512 to 1×1×512.
2. **Fully Connected Layer:** Outputs class scores for 14 classes.
3. **Softmax Activation:** Converts scores to probabilities.

A diagram of a number of rectangular objects

Description automatically generated

**References  
Deep Residual Learning for Image Recognition Paper:** [**Link**](https://arxiv.org/abs/1512.03385)**.**

**Xception**

**Introduction**

**Xception** builds on Inception, improving efficiency and performance with depthwise separable convolutions.

**Components:**

* **Depthwise Separable Convolutions**: Split spatial and channel-wise correlations.
* **Residual Connections**: Prevent degradation in deeper networks.
* **Global Average Pooling (GAP)**: Reduces feature maps before classification.

**Architecture**

**Stage 1: Entry Flow**

1. **Conv1**:
   * 32 filters, 3×3 kernel, stride 2, padding 1.
   * Input: 128×128×3 → Output: 64×64×32.
2. **Conv2**:
   * 64 filters, 3×3 kernel, stride 1 → Output: 64×64×64.

***... (Middle and Exit Flow stages would continue with specifics.)***

**Final Layers**

1. **Global Average Pooling**: Reduces 4×4×2048 to 1×1×2048.
2. **Fully Connected Layer**: Maps features to 14 classes.
3. **Softmax Activation**: Outputs probabilities.

A diagram of a flowchart

Description automatically generated

**References**  
**Xception: Deep Learning with Depthwise Separable Convolutions Paper:** [Link](https://arxiv.org/abs/1610.02357).

**DenseNet**

**Introduction**

DenseNet connects each layer to every other layer in a feed-forward fashion, maximizing feature reuse and efficiency.

**Concepts:**

1. **Dense Connectivity**: Layers receive inputs from all preceding layers, concatenating feature maps.
2. **Growth Rate (k)**: Determines feature map increase per layer.
3. **Transition Layers**: Use 1×1 convolutions and 2×2 average pooling to manage complexity.
4. **Bottleneck Layers**: Reduce input size with 1×1 convolutions before 3×3 convolutions.

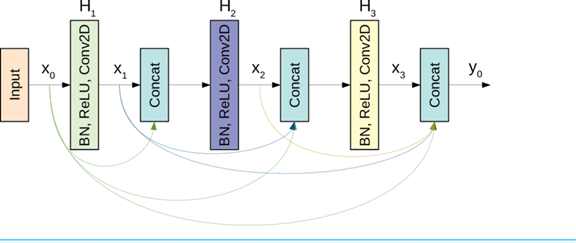
**Architecture**

**Stage 1: Initial Convolution**

* Single layer: 64 filters, 7×7 kernel, stride 2, padding 3.
* Followed by BN, ReLU, and Max Pooling.
* Input: 128×128×3 → Output: 64×64×64.

***... (Dense Block and Transition Layer stages would follow with clear descriptions.)***

**Stage 6: Classification Layer**

1. **Global Average Pooling**: Converts 8×8×F′′′ to 1×1×F′′′.
2. **Fully Connected Layer**: Maps features to 14 classes.
3. **Softmax Activation**: Outputs probabilities.

**References**  
Densely Connected Convolutional Networks Paper: [Link](https://arxiv.org/abs/1608.06993).

|  |  |  |  |
| --- | --- | --- | --- |
| Feature | ResNet | DenseNet | Xception |
| Primary Idea | Residual connections to address vanishing gradient problems. | Dense connectivity: every layer connected to all previous layers. | Depthwise separable convolutions for computational efficiency. |
| Architecture Depth | ResNet18, ResNet34, ResNet50, ResNet101, ResNet152. | DenseNet121, DenseNet169, DenseNet201, DenseNet264. | Xception with 36 layers (inspired by Inception). |
| Parameter Efficiency | Moderate (depends on network depth). | High (reduced parameters due to feature reuse). | High (depthwise separable convolutions reduce parameters). |
| Gradient Flow | Improved with residual connections. | Maximized with dense connectivity. | Improved with residual connections. |
| Feature Reuse | Moderate (via residual connections). | High (via concatenation of all previous layers). | Low (standard residual connections). |
| Computation Cost | High for deeper variants like ResNet152. | High for deep networks with many layers. | Lower due to depthwise separable convolutions. |
| Suitability for Transfer Learning | Excellent (commonly used pretrained on ImageNet). | Excellent (DenseNet121 is widely used for transfer learning). | Excellent (used for transfer learning tasks, e.g., XceptionV1). |
| Training Stability | Stable due to residual connections. | Stable due to dense connections. | Stable due to depthwise separable convolutions and residual connections. |
| Performance on Small Datasets | Can overfit on small datasets. | Performs well due to parameter efficiency. | Performs well due to reduced computation. |
| Memory Usage | Moderate to high depending on depth. | High due to concatenation of features. | Low due to factorized convolutions. |
| Strengths | Flexibility, strong performance on various tasks. | Efficient feature propagation, good for small datasets. | Computational efficiency, improved channel mixing and spatial filtering. |
| Weaknesses | High computational cost for deep networks. | High memory consumption and architectural complexity. | Complex architecture compared to simpler models. |
| Popular Use Cases | Image classification, object detection, and tasks requiring deep networks. | Tasks requiring efficient computation and feature reuse, small datasets. | Computationally constrained tasks, image classification, object detection. |
| Example Pretrained Model | ResNet50 pretrained on ImageNet. | DenseNet121 pretrained on ImageNet. | XceptionV1 pretrained on ImageNet. |

**Model Training process**

**ResNet Training Process and Model Selection**

* The training process for **ResNet** architectures involves standard techniques such as data augmentation, optimization algorithms, and regularization methods. Each version of **ResNet** (**ResNet50**, **ResNet34**, **ResNet18**) differs in terms of depth, parameter count, and computational cost. Below is a detailed explanation of the training and evaluation process, highlighting how **ResNet18** emerged as the best-performing model.

**Training ResNet18: The Best Model**

**Architecture:**

**ResNet18**, with its 18 layers, is the shallowest architecture among the **ResNet** family. This simplicity contributes to faster training and reduced memory usage while still maintaining competitive performance.

**Training Process:**

* **Initialization**: Models are implemented from scratch.
* **Optimization**: Two optimizers were tested: Adam and SGD.
* **Adam**: Used first for training But Achieved overfitting.
* **SGD**: Later tested and found to achieve better performance.
* **Batch Size**: A batch size of 8 was employed.
* **Epochs**: Training was conducted for 70 epochs with early stopping to monitor overfitting.

**Performance and Findings:**

**ResNet18** consistently delivered the best balance of accuracy and computational efficiency. The experiments also revealed that training **ResNet18** with SGD yielded higher validation accuracy compared to Adam, as shown in the first image. This optimizer switch solidified **ResNet18's** position as the optimal model for this task.

**2. Comparative Analysis: ResNet18 vs. Deeper Architectures**

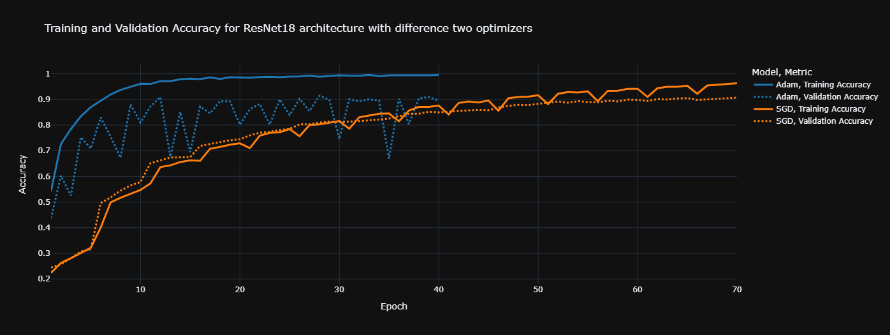
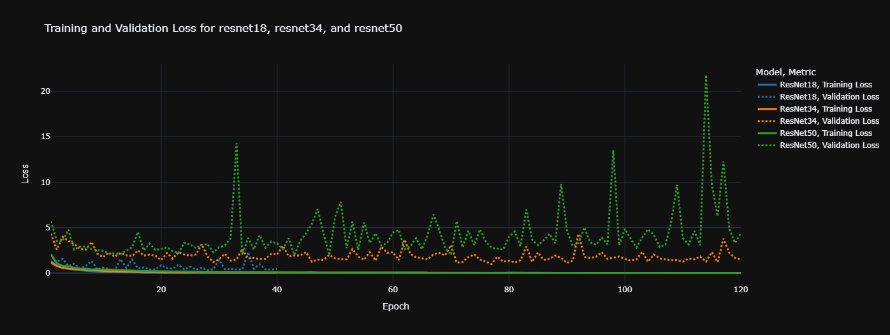
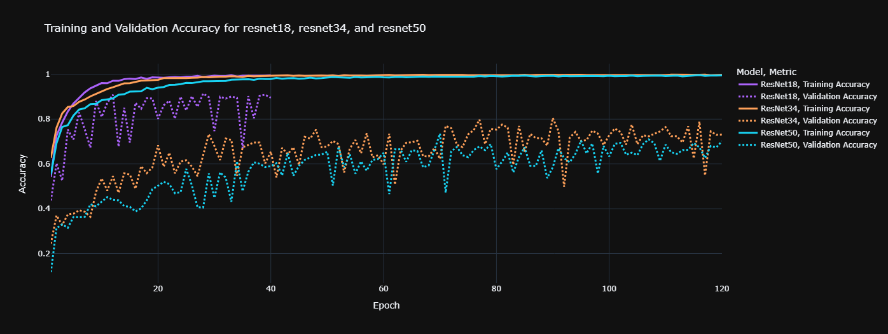
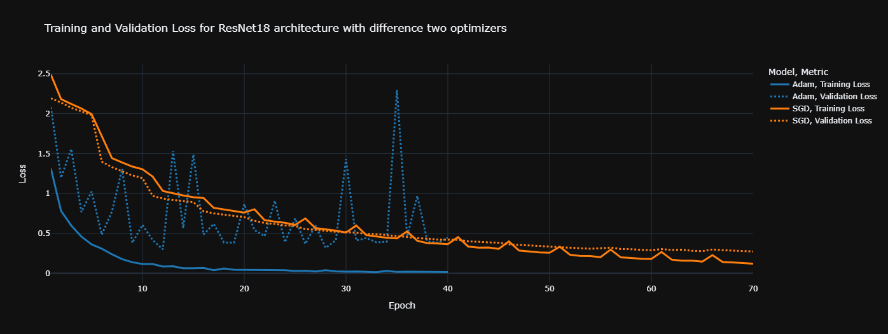
**Convergence Speed:**

* **ResNet18** converged the fastest, requiring fewer epochs to achieve high accuracy.
* **ResNet34** and **ResNet50** exhibited slower convergence due to increased depth and complexity.

**Performance:**

* While **ResNet50** achieved slightly higher training accuracy, its computational demands and slower convergence made it less practical.
* **ResNet18** demonstrated superior performance on validation data when paired with the **SGD** optimizer, outperforming deeper models.

**Visualizing Performance**

* The training and validation accuracy for **ResNet18**, compared with other models and optimizers, is illustrated in the accompanying figures. The second image underscores the comparative performance of **ResNet18**, **Xception**, and **DenseNet121**, showcasing **ResNet18's** efficiency and accuracy.

**Results and Analysis**

**1. Visualizations:**

**Accuracy and Loss Curves:**

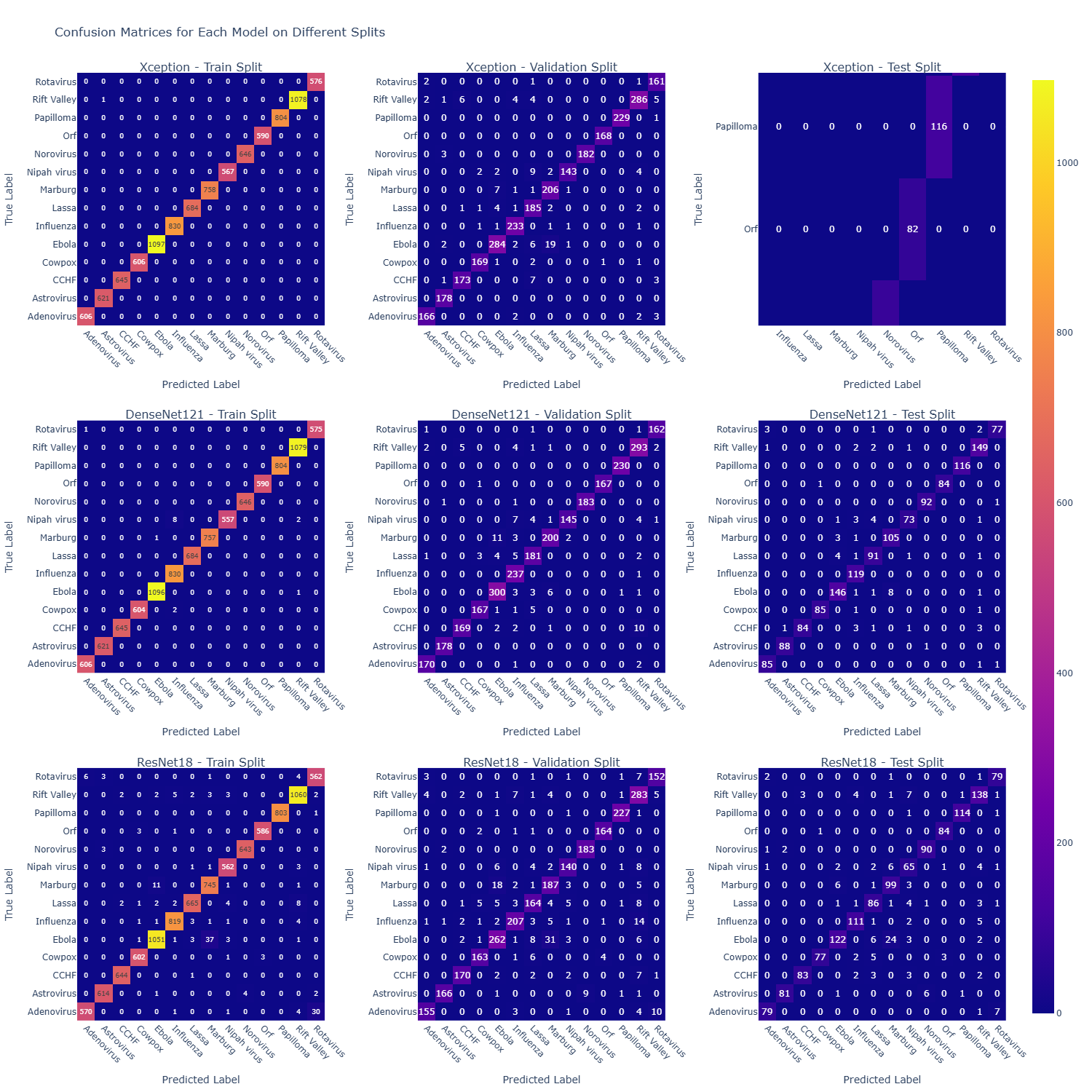
* **ResNet:**
  + **Accuracy Curve:** The accuracy steadily improves over epochs, reaching high accuracy after approximately 70 epochs.
  + **Loss Curve:** The loss decreases gradually, indicating effective training. However, slight overfitting is observed after 70 epochs as the validation loss plateaus or slightly increases.
* **DenseNet:**
  + **Accuracy Curve:** Rapid initial improvement, reaching high accuracy quickly but showing a slight decline in the final epochs, likely due to overfitting.
  + **Loss Curve:** Decreases faster compared to **ResNet18**, suggesting quicker learning but also vulnerability to overfitting.
* **Xception:**
  + **Accuracy Curve:** Increases steadily and achieves stable high performance around 20 epochs.
  + A graph on a black background

    AI-generated content may be incorrect.A graph on a screen

    AI-generated content may be incorrect.**Loss Curve**: Smooth decrease, indicating efficient learning without significant fluctuations.

**Confusion Matrices:**

* **ResNet:**
  + The confusion matrix shows good classification performance, though there is some confusion between similar viruses (e.g., Ebola and Marburg).
  + A few misclassifications occurred for viruses with overlapping features.
* **DenseNet:**
  + **DenseNet’s** confusion matrix shows better distinction between most virus types, especially for those with subtle differences.
  + Fewer misclassifications compared to **ResNet**, particularly for complex virus types.
* **Xception**:
  + The confusion matrix reveals fewer errors overall, particularly on smaller virus types like Norovirus and Astrovivirus.
  + It struggles more with distinguishing viruses with similar structural features.

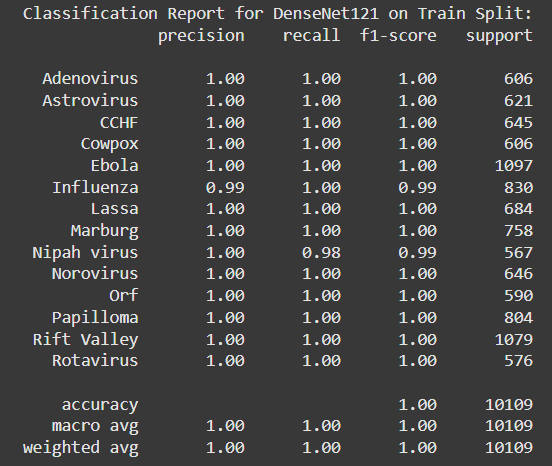
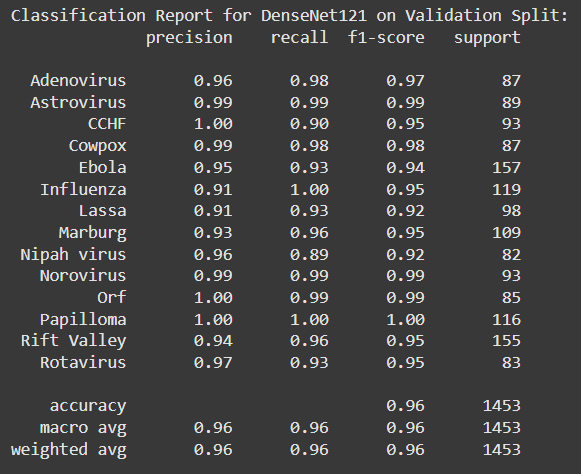
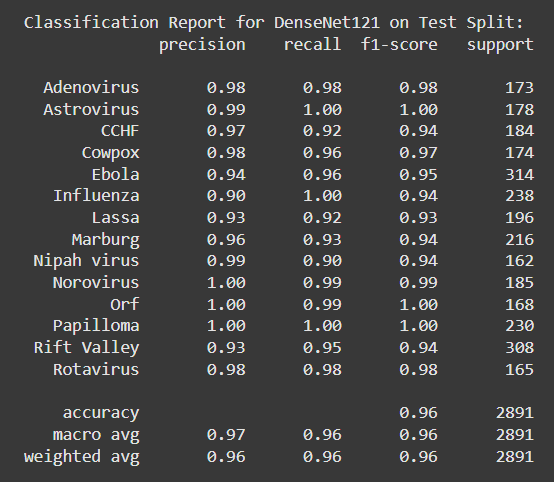


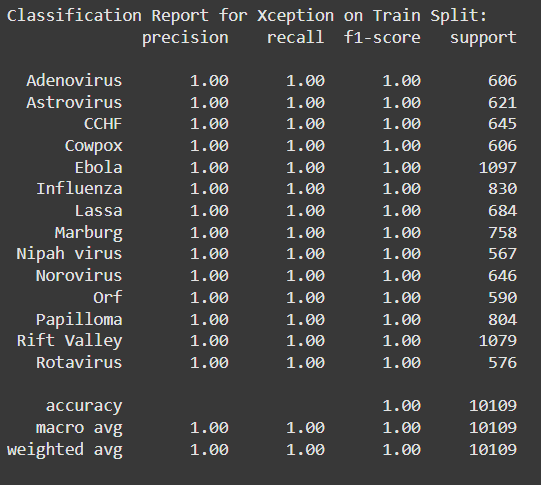
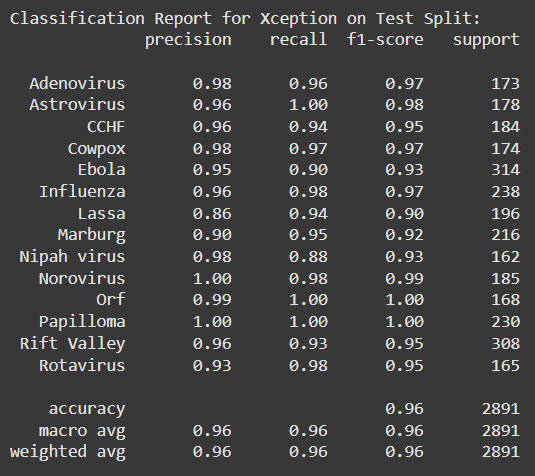
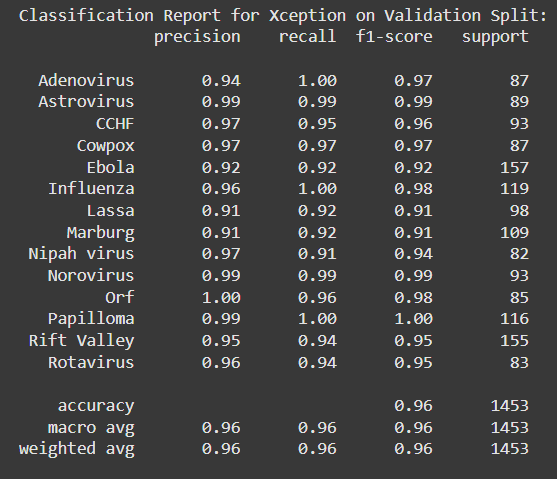
**2. Performance Metrics:**

* **A screenshot of a computer screen

  AI-generated content may be incorrect.A screenshot of a computer screen

  AI-generated content may be incorrect.A screenshot of a computer screen

  AI-generated content may be incorrect.ResNet:**
* **DenseNet:**
* **Xception:**



1. **ROC Curves:  
   A blue graph with yellow dots

   AI-generated content may be incorrect.A blue graph with yellow dots

   AI-generated content may be incorrect.A screen shot of a graph

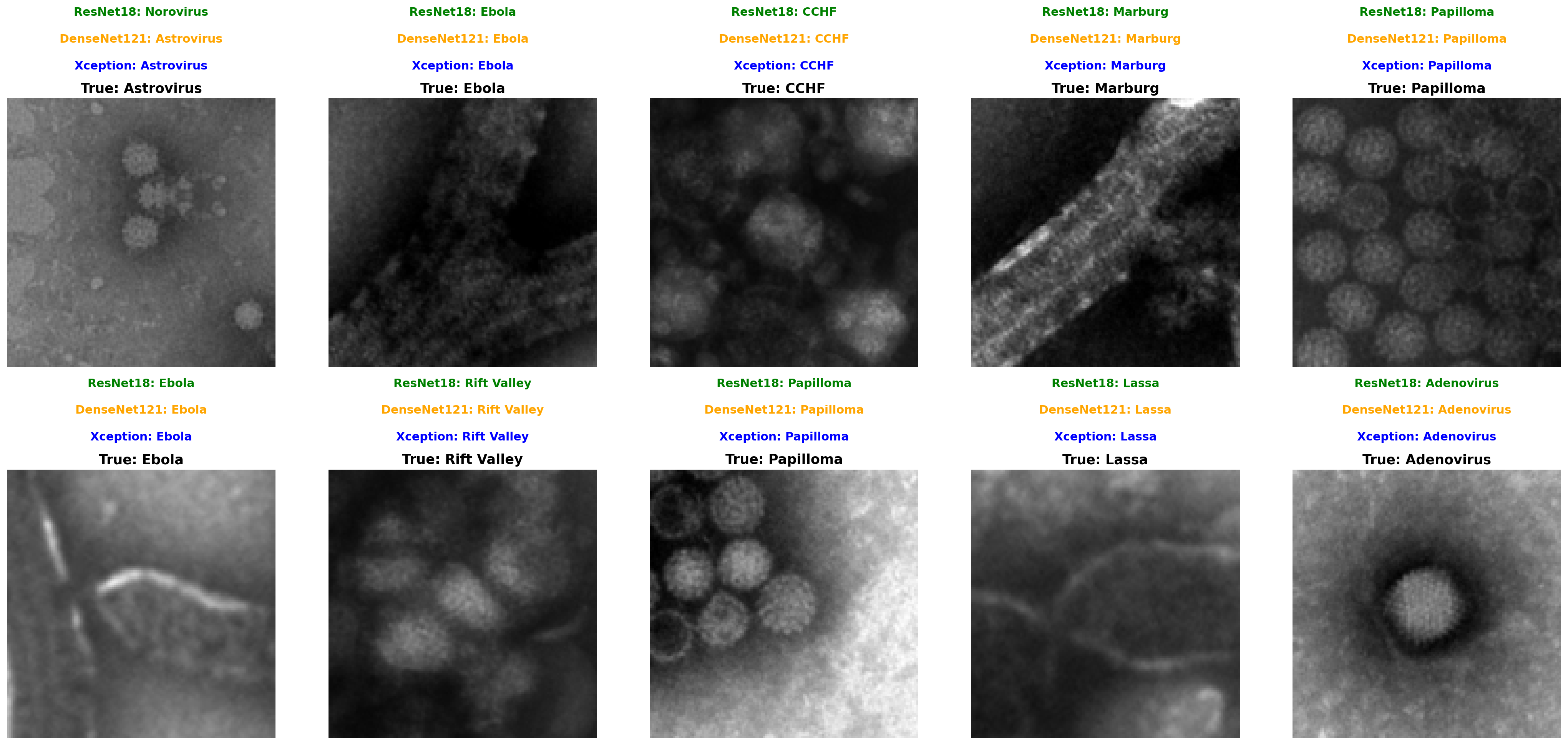
   AI-generated content may be incorrect.A blue graph with yellow dots

   AI-generated content may be incorrect.A blue graph with white text

   AI-generated content may be incorrect.A blue graph with a dotted line

   AI-generated content may be incorrect.**
2. **Evaluation:**

**A graph of different colored bars

AI-generated content may be incorrect.**

1. **Analysis:**

* **ResNet18:**
  + **Pros**: Strong performance on diverse tasks, and well-suited for large datasets.
  + **Cons**: Higher computational cost and risk of overfitting on small datasets.
  + **Impact**: Residual connections helped mitigate vanishing gradients but increased resource consumption**.**
* **DenseNet:**
  + **Pros**: Efficient feature reuse, good for smaller datasets, high accuracy with fewer parameters.
  + **Cons**: High memory usage due to dense connections and longer training times for large datasets.
  + **Impact**: Dense connectivity allowed for better classification of similar viruses.
* **Xception:**
  + **Pros**: Computationally efficient, Reduced Overfitting, and faster convergence.
  + **Cons**: Struggled with highly similar virus types and lower accuracy for complex tasks.
  + **Impact**: Depthwise separable convolutions improved efficiency but limited its ability to capture intricate features.

**Suitability for the Given Task:**

* **Best Architecture: DenseNet121**
  + **Justification**: **DenseNet121** demonstrated the highest accuracy, efficient handling of subtle structural differences, and fewer classification errors for complex virus types.
* **Alternatives**:
  + **ResNet18**: Reliable but less accurate for complex features.
  + **Xception**: Efficient for smaller viruses but less suitable for complex distinctions.