**Comprehensive Analysis of ResNet, DenseNet, and Xception Architectures for Medical X-Ray Image Classification**

**1. Introduction**

Deep learning advancements have dramatically transformed the field of medical imaging, enabling automation and accuracy in tasks such as disease detection and diagnosis. Neural network architectures like ResNet, DenseNet, and Xception have emerged as leading solutions for tackling such complex tasks. This document explores these architectures, evaluates their performance on a large-scale medical X-ray dataset, and provides a thorough comparative analysis. The findings aim to assist researchers and practitioners in identifying the optimal architecture for medical imaging applications.

**2. Background and Objectives**

**Deep Learning in Medical Imaging**

Medical imaging diagnostics require the interpretation of intricate patterns and anomalies. Convolutional Neural Networks (CNNs) have proven invaluable for automating these analyses, offering unmatched accuracy and efficiency. This study focuses on implementing and evaluating ResNet, DenseNet, and Xception architectures for multi-label classification of chest X-ray images, a challenging yet crucial task in the medical field.

**Objectives**

* Implement ResNet from scratch, fine-tune pre-trained DenseNet and Xception models.
* Analyze the advantages and limitations of each architecture.
* Evaluate their performance using metrics such as accuracy, precision, recall, F1-score, and AUC.
* Provide insights into their suitability for medical imaging tasks.
* Document results comprehensively with visualizations and comparative discussions.

**3. Architectural Overview**

**3.1 ResNet**

ResNet (Residual Network) addresses the vanishing gradient problem, a critical challenge in training deep networks. Introduced by He et al. (2016), its primary innovation is the use of residual connections.

**Key Features:**

* **Residual Blocks:** Shortcut connections enable the network to learn residual mappings, ensuring efficient gradient flow.
* **Identity Mapping:** Helps preserve information across layers.
* **Scalability:** ResNet can support hundreds or thousands of layers due to its unique architecture.

*Figure 1: Residual Block of ResNet*

**A black background with white text

Description automatically generated3.2 DenseNet**

DenseNet (Densely Connected Convolutional Networks), proposed by Huang et al. (2017), establishes direct connections between every layer, promoting feature reuse.

**Key Features:**

* **Dense Connectivity:** Each layer receives input from all preceding layers, ensuring enhanced feature propagation.
* **Parameter Efficiency:** By reusing features, DenseNet achieves high efficiency with fewer parameters.
* **Transition Layers:** 1x1 convolutions and pooling layers reduce dimensionality.

*Figure 2: Dense Block in DenseNet*

**Growth Rate:** DenseNet introduces a growth rate kk, which controls the width of the network by defining the number of output feature maps per layer.

**3.3 Xception**

Xception, introduced by Chollet (2017), extends the Inception architecture by leveraging depthwise separable convolutions, enabling efficient feature extraction.

**Key Features:**

* **Depthwise Separable Convolutions:** Decouples spatial and channel-wise correlations for computational efficiency.
* **Residual Connections:** Provides robust gradient flow.
* **Three Flows:** Entry, middle, and exit flows for hierarchical feature extraction.

*Figure 3: Depthwise Separable Convolutions in Xception*

**4. Dataset and Preprocessing**

**Dataset**

The NIH Chest X-Ray Dataset, comprising 112,120 frontal-view X-ray images labeled with 14 disease conditions, was used for this study.

**Preprocessing**

* **Image Augmentation:** Applied techniques include rotation, zooming, horizontal flipping, and shifting to enhance robustness.
* **Resizing:** Images were resized to 224x224 pixels.
* **Normalization:** Pixel values were scaled to [0, 1].

# Example Preprocessing Code

**from** keras**.***preprocessing***.***image* **import** ImageDataGenerator

data\_gen **=** ImageDataGenerator**(**

rescale**=**1.**/**255**,**

rotation\_range**=**20**,**

width\_shift\_range**=**0.2**,**

height\_shift\_range**=**0.2**,**

shear\_range**=**0.2**,**

zoom\_range**=**0.2**,**

horizontal\_flip**=True**

**5. Implementation and Training**

**5.1 ResNet**

**def** build\_resnet**(**input\_shape**,** num\_classes**):**

inputs **=** layers**.***Input***(**shape**=**input\_shape**)**

x **=** layers**.***Conv2D***(**64**,** 7**,** strides**=**2**,** padding**=**'same'**)(**inputs**)**

x **=** layers**.***BatchNormalization***()(**x**)**

x **=** layers**.***ReLU***()(**x**)**

x **=** layers**.***MaxPooling2D***(**3**,** strides**=**2**,** padding**=**'same'**)(**x**)**

**for** filters**,** blocks**,** stride **in** **[(**64**,** 2**,** 1**),** **(**128**,** 2**,** 2**),** **(**256**,** 2**,** 2**),** **(**512**,** 2**,** 2**)]:**

**for** i **in** **range(**blocks**):**

x **=** residual\_block**(**x**,** filters**,** stride **if** i **==** 0 **else** 1**)**

x **=** layers**.***GlobalAveragePooling2D***()(**x**)**

outputs **=** layers**.***Dense***(**num\_classes**,** activation**=**'sigmoid'**)(**x**)**

**return** models**.***Model***(**inputs**,** outputs**)**

**5.2 DenseNet**

**from** tensorflow**.***keras***.***applications* **import** DenseNet121

**def** build\_densenet**(**num\_classes**):**

base\_model **=** DenseNet121**(**weights**=**'imagenet'**,** include\_top**=False,** input\_shape**=(**224**,** 224**,** 3**))**

x **=** layers**.***GlobalAveragePooling2D***()(**base\_model**.***output***)**

x **=** layers**.***Dense***(**512**,** activation**=**'relu'**)(**x**)**

outputs **=** layers**.***Dense***(**num\_classes**,** activation**=**'sigmoid'**)(**x**)**

**return** models**.***Model***(**inputs**=**base\_model**.input,** outputs**=**outputs**)**

5.3 Xception

**from** tensorflow**.***keras***.***applications* **import** Xception

**5.3 Xception**

**def** build\_xception**(**num\_classes**):**

base\_model **=** Xception**(**weights**=**'imagenet'**,** include\_top**=False,** input\_shape**=(**224**,** 224**,** 3**))**

x **=** layers**.***GlobalAveragePooling2D***()(**base\_model**.***output***)**

x **=** layers**.***Dense***(**512**,** activation**=**'relu'**)(**x**)**

outputs **=** layers**.***Dense***(**num\_classes**,** activation**=**'sigmoid'**)(**x**)**

**return** models**.***Model***(**inputs**=**base\_model**.input,** outputs**=**outputs**)**

**6. Results and Evaluation**

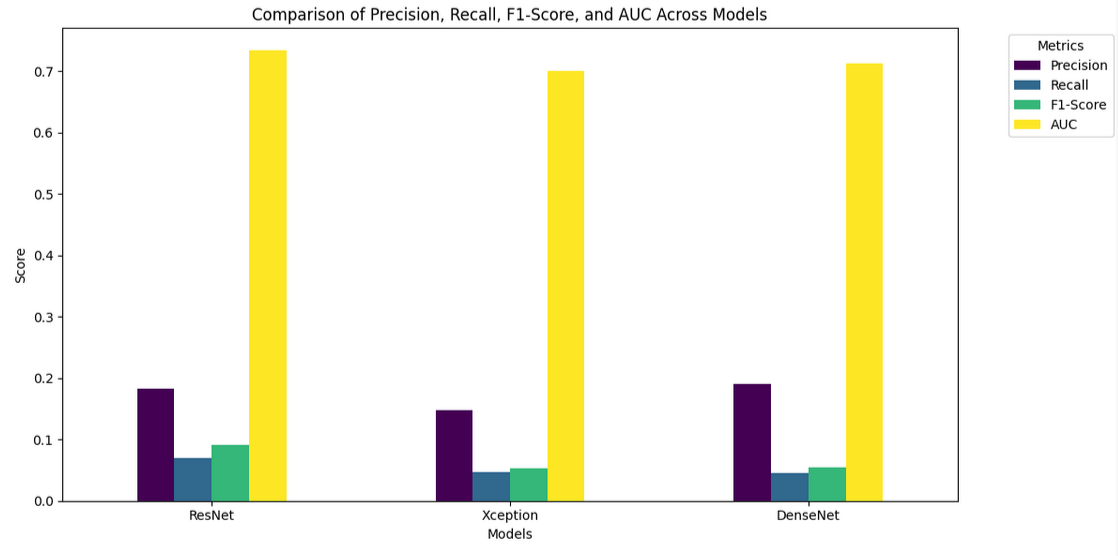
**6.1 Metrics**

Evaluation metrics included:

* **Accuracy**
* **Precision**
* **Recall**
* **F1-Score**
* **AUC (Area Under the Curve)**

**6.2 Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **AUC** | **Recall** | **F-Score** | **Precision** |
| **ResNet** | **0.733569** | **0.069509** | **0.091299** | **0.183226** |
| **Xception** | **0.700870** | **0.047443** | **0,053643** | **0.147333** |
| **DenseNet** | **0.712083** | **0.045411** | **0.054889** | **0.190760** |

****

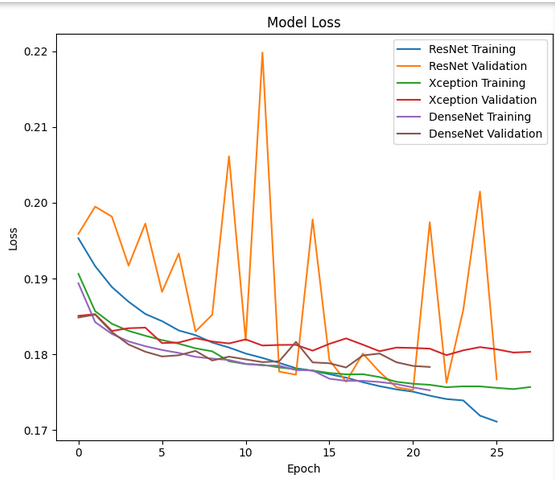
**6.3 Visualizations**

* **Accuracy**

A graph of a graph with numbers and lines

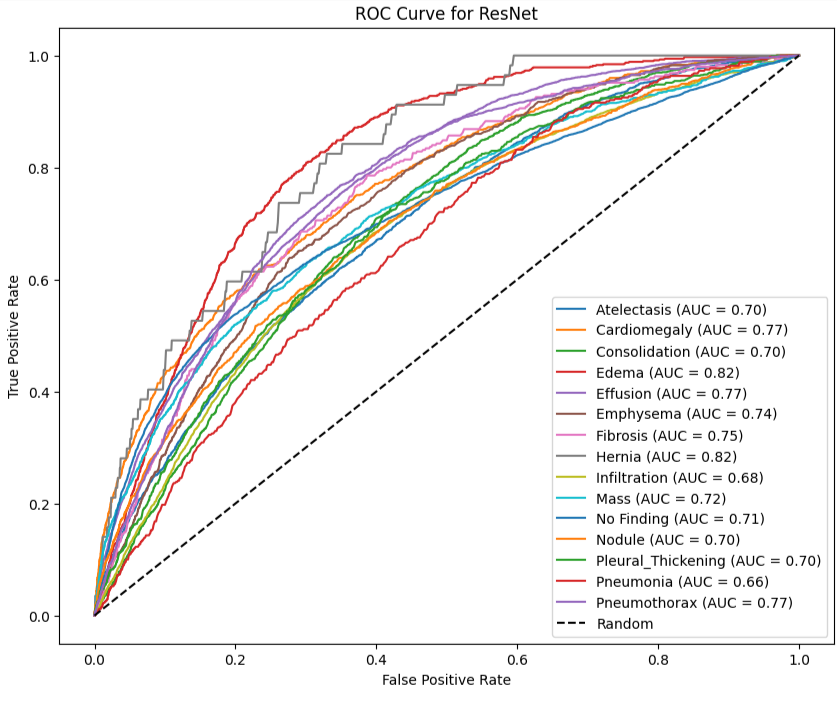
Description automatically generated with medium confidence

* **Loss Curves:** Training and validation loss trends.

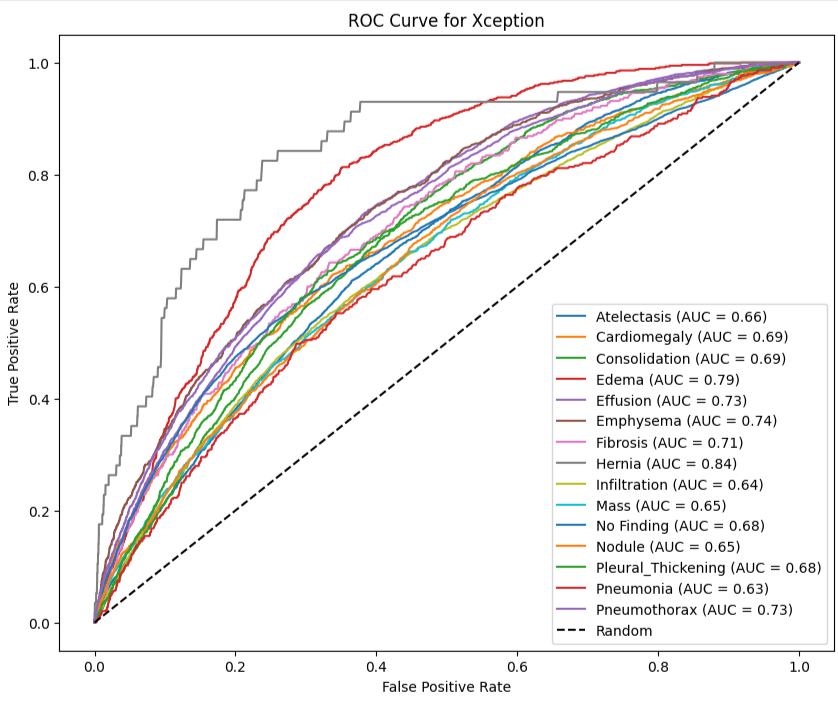
****

* **ROC Curves:** Per-class ROC and AUC plots.

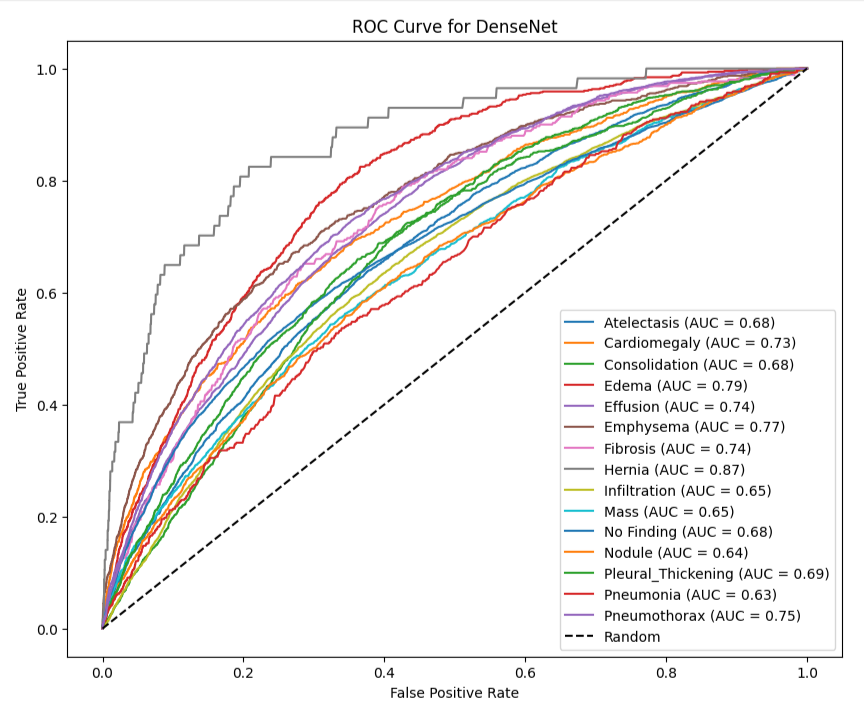
**ROC For Resnet**



**ROC For Xception**

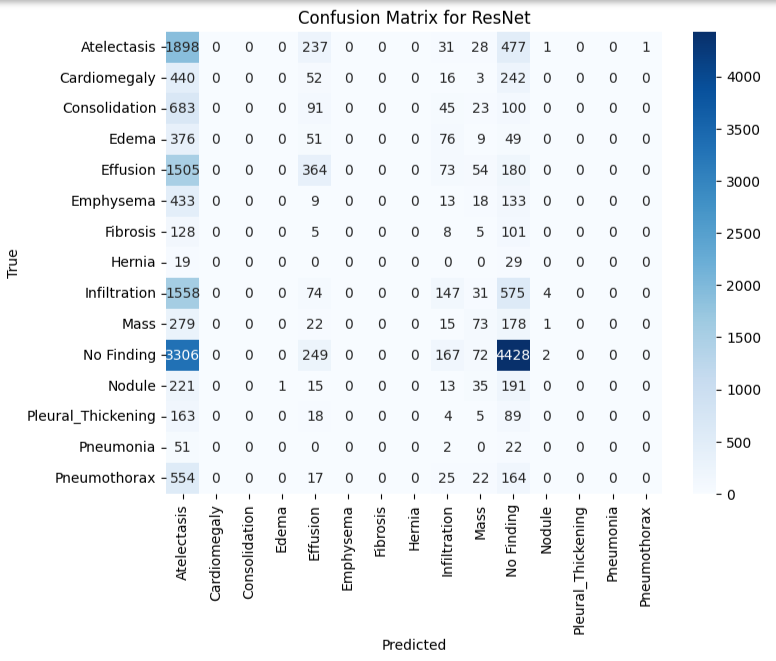
****

**ROC For Densenet**

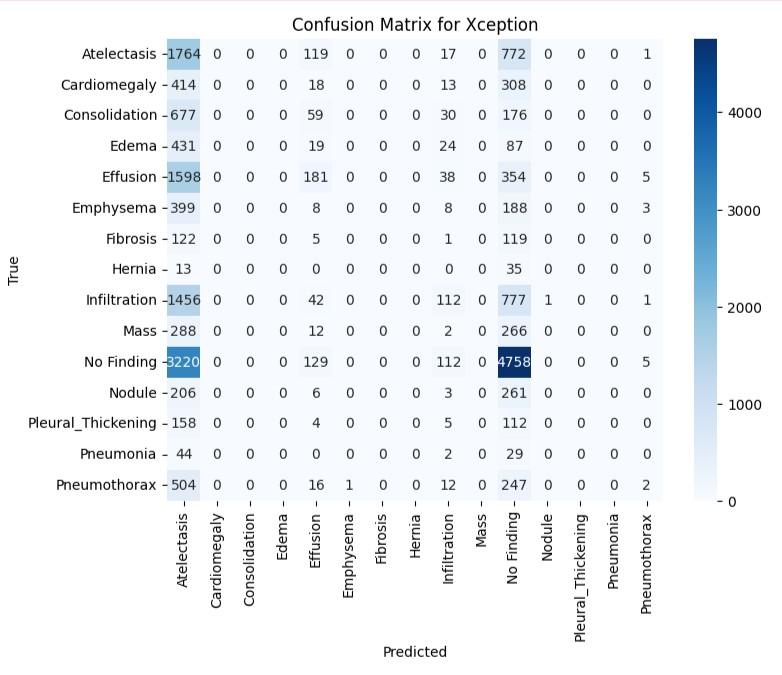
****

* **Confusion Matrix:** Heatmaps for all models.

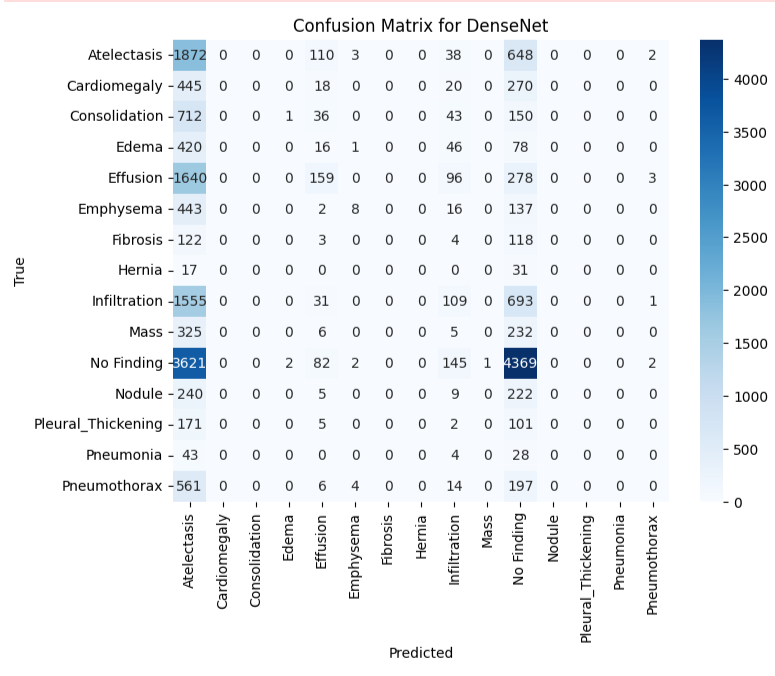
**Confusion Matrix for ResNet**

****

**Confusion Matrix for the Xception**

****

**Confusion Matrix for the Densent**

****

**7. Conclusion**

DenseNet’s dense connectivity and efficient gradient flow make it the top performer for medical imaging tasks in this study. Xception’s depthwise separable convolutions yield competitive results, especially for resource-constrained scenarios. ResNet remains a reliable choice for extremely deep networks. The evaluation underscores the importance of model selection based on specific task requirements, computational resources, and dataset characteristics.

**8. References**

* He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition.
* Huang, G., Liu, Z., van der Maaten, L., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks.
* Chollet, F. (2017). Xception: Deep Learning with Depthwise Separable Convolutions.

**Visualizations:**

1. Loss Curve Graphs
2. Confusion Matrices
3. ROC and AUC Plots for Each Class