Pneumonia Classification using different CNNs

Omar Tamer, Abdelhalim Ashraf, Eid Osama
Abdelrahman Ramadan, Mohyeldin Maher, Ali Adel

December 2024

Contents

1 Introduction

Pneumonia is a serious respiratory infection that primarily affects the lungs and can cause serious health complications if not diagnosed and treated promptly. Despite its high prevalence, especially among children, the elderly and immunosuppressed individuals, pneumonia can be challenging to diagnose accurately. Diagnosing pneumonia typically relies on chest X-rays and clinical expertise , however; Human error and variability among radiologists can result in misdiagnosis. In remote or underdeveloped areas, access to radiologists and specialists is limited. These factors call for reliable and automated classification methods to support healthcare professionals. Advances in machine learning (ML) and deep learning provide an opportunity to: Automatically analyze chest radiographs to classify pneumonia. Improve accuracy and reduce diagnosis time. Support radiologists by acting as a second opinion.

2 Data set

The data set is organized into 3 folders (train, test, val) and contains subfolders for each category of images (Pneumonia / Normal). There are 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia/Normal). in figure 1 The normal chest X-ray (left panel) depicts clear lungs without any areas of abnormal opacification in the image. Bacterial pneumonia (middle) typically exhibits a focal lobar consolidation, in this case in the right upper lobe (white arrows), whereas viral pneumonia (right) manifests with a more diffuse “interstitial” pattern in both lungs.

Figure 1: demo of the images in the dataset

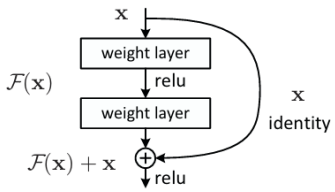


Figure 2: Residual learning: a building block.

3 CNN Architectures

3.1 ResNet

Deeper neural networks are more difficult to train. a residual learning framework was presented to ease the training of networks that are substantially deeper than those used previously[2]. The need for deeper networks arises from more difficult problems such as the image-net data set [4] and one question was fatal **.Is learning better networks as easy as stacking more layers?**. answering this question was difficult due to vanishing / exploding gradient problem. Increasing the depth of the network saturates the accuracy and then leads to degrading performance as shown in Figure 5.if a shallow network was constructed then adding more layers will learn linear mapping and the the rest of layers are copied from the shallow network. this indicates that deeper networks will not have training error higher then the shallow networks , but the analysis showed that this construction does not produce better solutions. the hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers. the residual block shown in figure 2 the formulation of the blocK is $y = F(xW_i) + x$ Here, x and y are the input and output vectors of the layers considered. The function $F(xW_i)$ represents the residual mapping to be learned.

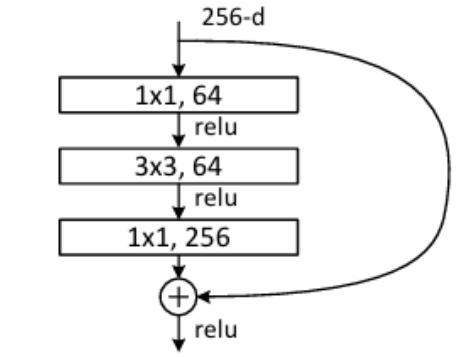


Figure 3: a"bottleneck"building block for ResNet-50

50-layer
7×7, 64, stride 2
3×3 max pool, stride 2
[1×1, 64 ; 3×3, 64 ; 1×1, 256] ×3
[1×1, 128 ; 3×3, 128 ; 1×1, 512] ×4
[1×1, 256 ; 3×3, 256 ; 1×1, 1024] ×6
[1×1, 512 ; 3×3, 512 ; 1×1, 2048] ×3
average pool, 1000-d fc, softmax

Figure 4: Architecture of resnet 50 . Building blocks are shown in brackets,with the numbers of blocks stacked

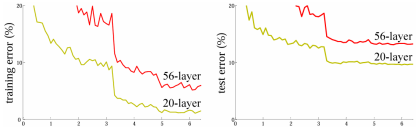


Figure 5: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

3.1.1 ResNet 50

in this project a specific resnet was used (50 layers). the residual block for this network is composed of 3 convolution-al layers followed by relu as shown in figure 3. the structure of the resnet 50 is shown in figure 4. the loss used was binary cross entropy and the optimizer was ADAM with learning rate = 0.001 and weight decay = 0.0001

3.2 DenseNet

as resent introduces skip connection to solve the vanish gradient problem for deeper neural network , dense net builds on this idea. all layers (with matching feature-map sizes) connects di rectly with each other [3]. To preserve the feed-forward nature, each layer obtains additional inputs from all preceding lay ers and passes on its own feature-maps to all subsequent layers as shown in figure 7. Dense net use concatenation not sum like resnet. Consequently, the l-th layer receives the feature-maps of all preceding layers, $x_0, x_1, \ldots, x_{l-1}$, as input:

$$x_l = H([x_0, x_1, \ldots, x_{l-1}])$$

where $x_0, x_1, \ldots, x_{l-1}$ refers to the concatenation of the feature-maps produced in layers $0, \ldots, l-1$. the $H_L(.)$ is a composite function with three consecutive operation: Batch normalization , RELU, convolution layer.if each function H_L produces k features then the L^{th} layer has $k_0 + k * (L-1)$ input feature map. We refer to the hyper parameter k as the growth rate of the network.

3.2.1 DenseNet-121

in this project , we use the the DenseNet with 121 layers . it composed of 4 Dense Blocks and 4 transition layers . each transition layer composed of 1*1 convolv layer followed by average pooling. the whole architecture shown in

Layers	Output Size	DenseNet-121
Convolution	112 × 112	7 × 7 conv, stride 2
Pooling	56 × 56	3 × 3 max pool, stride 2
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv
	28 × 28	2 × 2 average pool, stride 2
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv
	14 × 14	2 × 2 average pool, stride 2
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Transition Layer (3)	14 × 14	1 × 1 conv
	7 × 7	2 × 2 average pool, stride 2
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$
Classification Layer	1 × 1	7 × 7 global average pool
		1000D fully-connected, softmax

Figure 6: Dense Net architecture , The growth rate for all the networks is k=32

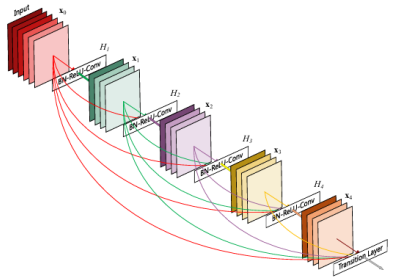


Figure 7: A 5-layer dense block with a growth rate of k = 4. Each layer takes all preceding feature-maps as input.

figure 6. the loss used is binary cross entropy with stochastic gradient descent with learning rate = 0.0001 and momentum = 0.9.

3.3 Xception

Xception is build on the idea of inception [5]. the inception hypothesis is the channel correlation relationships and spatial can be decoupled , this by convolving the channels by 1*1 kernels , then by 3*3 kernels followed by concatenation as shown in figure 8. we can reformulate the canonical module as 1*1 convolution kernel to get the channel correlations then divide the output channel into segments and performing 3*3 convolution in each segment to know spa-

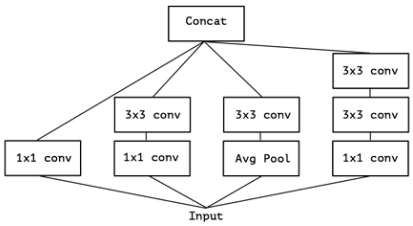


Figure 8: A canonical Inception module (Inception V3)

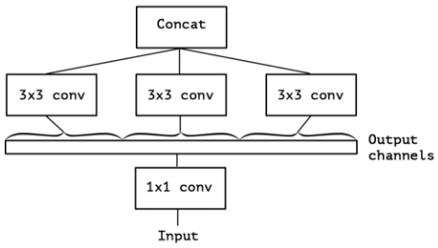


Figure 9: A strictly equivalent reformulation of the simplified inception module

tial relationships as shown in figure 9 An "extreme" version of an Inception module, based on this stronger hypothesis, would first use a 1x1 convolution to map cross-channel correlations, and would then separately map the spatial correlations of every output channel as shown in figure 10 .

this extreme version is almost identical to depthwise sepa rable convolution. A depthwise separable convolution consists in a depthwise convolution, i.e. a spatial convolution performed independently over each channel of an input, followed by a pointwise convolution, i.e. a 1x1 convolution, projecting the channels

Figure 4. An "extreme" version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

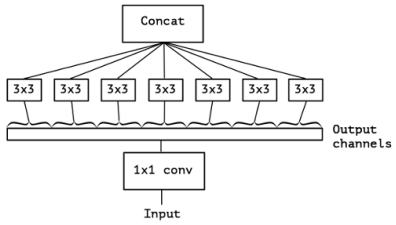


Figure 10: An "extreme" version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution

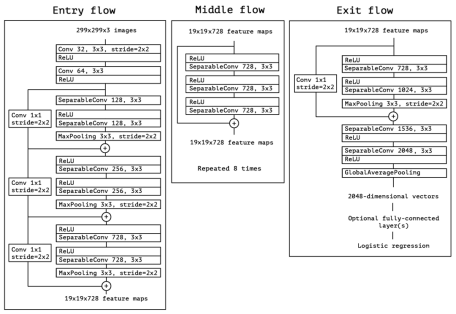


Figure 11: The Xception architecture: the data first goes through the entry flow, then through the middle flow which is repeated eight times, and finally through the exit flow. Note that all Convolution and SeparableConvolution layers are followed by batch normalization . All SeparableConvolution layers use a depth multiplier of 1 (no depth expansion).

output by the depthwise convolution onto a new channel space.so in xception has been suggested that it may be possible to improve upon the Inception family of archi tectures by replacing Inception modules with depthwise sep arable convolutions, i.e. by building models that would be stacks of depthwise separable convolutions.the Xception [1] architecture is a linear stack of depthwise separable convolution layers with residual con nections as shown in figure 11 .

4 pros and cons for each architecture

4.0.1 ResNet

pros

- Scalable Depth:

  Trained networks with up to 152 layers on ImageNet, achieving a 3.57 precnt top-5 error rate. Depth allows for more complex feature extraction without significant degradation of performance.

- Optimization Efficiency:

  Residual connections (skip connections) help mitigate vanishing gradients by allowing gradients to flow directly back through identity shortcuts.

- Transferability:

Residual networks generalize well across multiple tasks like object detection (COCO dataset), showing their adaptability

cons

- Redundant Parameters: Identity mapping and summation in skip connections can result in parameter redundancy.
- High Computational Cost for Larger Models: Although efficient in terms of training deep networks, inference can still be costly, particularly in deeper versions like ResNet-152.
- Feature Reuse Limitation: Summation in residual blocks leads to limited feature reuse compared to concatenation-based methods like DenseNet.

.

4.0.2 DenseNet

pros

- Efficient Parameter Usage:

  Dense connections lead to reduced parameters since features are reused rather than relearned, outperforming ResNet on CIFAR datasets with fewer parameters. Example: DenseNet achieves an error rate of 3.46 percent on CIFAR-10 compared to ResNet's 4.62 percent with similar or fewer parameters.

- Feature Reuse:

  Each layer accesses feature maps from all preceding layers, enabling strong feature propagation and reducing redundancy.

- Regularization Effect:

  Dense connections act as an implicit regularizer, reducing overfitting, especially on smaller datasets.

cons

- Memory Usage: Dense connections require keeping all feature maps in memory, making DenseNet more memory-intensive than ResNet.
- Scaling Limitations: As network depth increases, the feature concatenation can lead to unwieldy memory and computation requirements.

- Complexity of Implementation: The dense connectivity pattern can be harder to implement and debug compared to simpler architectures like ResNet.

4.0.3 Xception

pros

- Efficiency through Depthwise Separable Convolutions:

  Xception replaces standard convolutions with depthwise separable convolutions, reducing computational cost and number of parameters significantly. Achieved better results on ImageNet compared to Inception V3, despite having similar parameter counts.

- Improved Decoupling of Features:

  Depthwise convolutions handle spatial filtering, while pointwise convolutions mix channels, optimizing the learning process.

- Versatility:

  Outperformed Inception V3 on large-scale datasets, including the JFT dataset with 17,000 classes, showing scalability and generalization.

cons

- Requires Large Datasets: Depthwise separable convolutions benefit more from large datasets, underperforming on smaller datasets compared to ResNet or DenseNet.
- Careful Hyperparameter Tuning: Suboptimal configurations can lead to underperformance due to the decoupling of spatial and channel operations.
- Single Hypothesis Dependency: The architecture relies heavily on the assumption that spatial and channel correlations are fully decoupled, which may not hold for all tasks.

5 Data preprocessing

the data set suffers from imbalance and also the number of validation examples as too small so to resolve this problem , we put all the data examples in single file . then , we remove all the extra data points from the data set to balance it. after that , we devide it to train , test , val.

5.1 ResNet50

- Resize image to 224 x 224
- Convert grayscale to RGB (3 channels)
- Random Horizontal Flip the images with a 50 percent chance of flipping the image.
- convert image To Tensor.
- Normalize the images with mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]

5.2 DenseNet-121

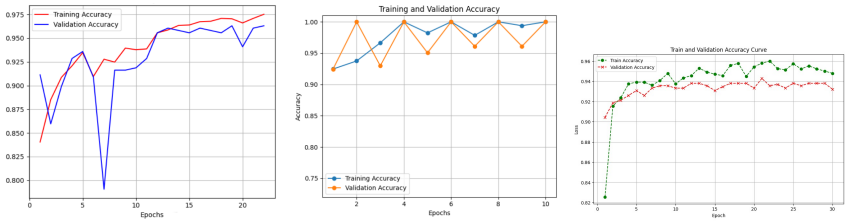
- convert to grey scale to 3 channels
- resize the image to 256
- crop the image to be 224*244
- convert the image to tensor
- Normalize the images with mean=[0.485, 0.456, 0.406] and std=[0.229, 0.224, 0.225]

5.3 Xception

image rescaling was applied to normalize pixel values by scaling them to the range [0, 1]. The data was then shuffled, and a split of 85 percent was allocated for training, while the remaining 15percent was reserved for testing and validation. To ensure consistency in input data, all images were reshaped using a data generator, converting them from various shapes and grayscale to a uniform shape of (299,299,3).

6 Results

6.1 Accuracy Plots



(a) ResNet Accuracy (b) Xception Accuracy (c) DenseNet Accuracy

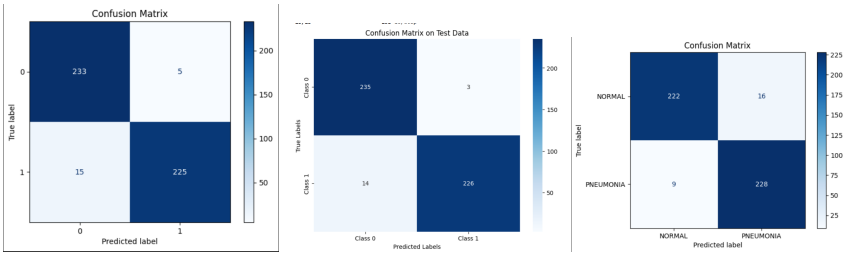
Figure 12: Accuracy plots for ResNet, Xception, and DenseNet.

6.2 Loss plots

(a) ResNet loss (b) Xception loss (c) DenseNet loss

Figure 13: loss plots for ResNet, Xception, and DenseNet.

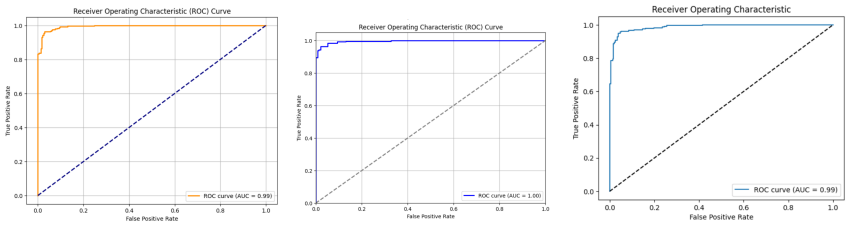
6.3 Confusion Matrix



(a) ResNet confusion matrix (b) Xception confusion matrix (c) DenseNet confusion matrix

Figure 14: confusion matricies for ResNet, Xception, and DenseNet.

6.4 ROC (AUC curve)



(a) ResNet ROC AUC (b) Xception ROC AUC (c) DenseNet ROC AUC

Figure 15: ROC curve with area under the curve for ResNet, Xception, and DenseNet.

7 metrics

Model	Class	Precision	Recall	F1-Score
ResNet	NORMAL	0.94	0.98	0.96
	PNEUMONIA	0.98	0.94	0.96
Xception	NORMAL	0.94	0.99	0.97
	PNEUMONIA	0.99	0.94	0.96
DenseNet	NORMAL	0.96	0.93	0.95
	PNEUMONIA	0.93	0.96	0.95

Table 1: Classification Metrics for Different Models

8 Analysis based on the result

for Accuracy ResNet achieves the best accuracy progression and convergence. DenseNet stabilizes quickly but underperforms compared to ResNet.Xception's accuracy instability limits its generalization. **for loss** ResNet achieves the lowest and most stable loss values. DenseNet minimizes loss efficiently but stagnates early. Xception struggles with loss optimization. **from the confusion matrix** ResNet produces a balanced and accurate prediction with very few misclassifications for both classes. Xception struggles to classify PNEUMONIA correctly, indicating an imbalance between classes. DenseNet performs well but misclassifies more NORMAL samples than ResNet.ResNet achieves balanced precision, recall, and F1-scores for both NORMAL and PNEUMONIA, with an F1-score of 0.96. Xception has high precision but poor recall for PNEUMONIA (F1 =

0.93), suggesting it misclassifies true positives more often. DenseNet is balanced but slightly underperforms compared to ResNet, achieving an F1-score of 0.95 for both classes.

9 conclusion

ResNet outperformed Xception and DenseNet because its residual architecture allowed it to train effectively from scratch, even with limited data. This robustness, combined with its balanced complexity and ability to avoid overfitting, made it the best choice for the small dataset. Fine-tuned Xception and DenseNet, while pretrained on larger datasets, struggled to adapt their complex architectures to the specific nuances of the smaller target dataset.

References

[1] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks, 2018.

[4] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015.

[5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.