DOI: https://dx.doi.org/10.4314/gjpas.v30i4.15

GLOBAL JOURNAL OF PURE AND APPLIED SCIENCES VOL. 30, 2024: 577-584
COPYRIGHT© BACHUDO SCIENCE CO. LTD PRINTED IN NIGERIA ISSN 1118 – 0579, e-ISSN: 2992 - 4464
www.globaljournalseries.com.ng, Email: globaljournalseries@gmail.com

THE COMPUTATIONAL EFFECT AND HYPERPARAMETERS TUNING OF DEEP CONVOLUTIONAL LAYER DEPTH OF HIGH-RANKING TUBERCULOSIS DETECTION MODELS

AUGUSTINE O. OTOBI, JOSEPH O. ESIN, IDONGESIT E. ETENG, B. I. ELE, S. I. ELE, D. U. ASHISHIE AND CLEOPAS, ANIETIE OKPAN

Email: otobiaugustine@unical.edu.ng, jesin57@unical.edu.ng, idongeteng@gmail.com, elebassey@unical.edu.ng ,myyrs2015up@gmail.com, ashishiedenis@gmail.com, okpancleo@gmail.com
ORCID:0000-0002-4711-8239

(Received 11 September 2024; Revision Accepted 23 October 2024)

ABSTRACT

In the past decade, artificial intelligence (AI) solutions have seen widespread application in medical fields, which include automated detection of breast cancer, brain tumors, physiological monitoring and detection of lung diseases such as pneumonia from chest X-Rays (CXRs). Machine learning, a subset of AI, empowers computers to learn autonomously, without direct human programming, by extracting patterns (feature extraction) from data (images). Deep learning, a specialized branch of machine learning, employs multiple convolutional layers to extract complex features from raw input data. This article examines the impact of varying the convolutional layers in deep learning models on their efficiency, focusing on algorithm complexity and parameter counts. We discuss theoretical foundations, and relevant factors affecting efficiency, and analyze algorithm complexity of high-ranking models developed for detecting tuberculosis from chest x-ray (CXR) using convolutional neural networks. The number of convolutional layers significantly influences model efficiency, affecting both performance and computational complexity. We could conclude practically that optimal layer depth balances model efficiency, accuracy and resource utilization. This assertion was reached by developing a model with fewer convolutional layer depth using the ResNet18 architecture. The parameters count of the ResNet18 model developed was compared with other model developed to detect tuberculosis from chest X-Ray images. The result of this comparison proved that fewer layer depth with the right hyperparameter tuning can produce a better and more efficient deep learning solutions to societal problems. This article also gives insights for future research and practical applications which includes the exploration of adaptive architectures that can dynamically adjust their depth based on the complexity of the task and available resources. Further research can also scientifically probe other methods of reducing the computational overhead of deep neural networks while maintaining priority for high computational performance, model scalability and efficiency.

KEYWORDS: Artificial Intelligence, Deep learning, convolutional neural networks, model efficiency, algorithm complexity, layer depth, parameters counts.

INTRODUCTION

Background: Deep learning has revolutionized various fields, with Convolutional Neural Networks (CNNs) being pivotal in tasks such as image recognition and natural language processing (LeCun et al., 2015). These networks mimic the human brain's neural structures, allowing machines to learn from vast amounts of data. CNNs have proven effective for visual tasks due to their ability to learn spatial hierarchies of features automatically and adaptively.
Problem Statement: One of the critical design

Augustine O. Otobi, University of Calabar, Department of Computer Science
Joseph O. Esin, University of Calabar, Department of Computer Science
Idongesit E. Eteng, University of Calabar, Department of Computer Science
B. I. Ele, University of Calabar, Department of Computer Science
S. I. Ele, University of Calabar, Department of Computer Science
D. U. Ashishie, University of Calabar, Department of Computer Science
Cleopas, Anietie Okpan, University of Calabar, Department of Computer Science

choices in CNNs is determining the number of convolutional layers. While deeper networks have the potential to learn more complex features, they also pose challenges such as increased computational complexity and risk of overfitting (He et al., 2016).

Objective: This research aims to explore the causal relationship between the number of convolutional layers in CNNs and model efficiency, with a particular focus on algorithm complexity.

Significance: Understanding this relationship helps in designing more efficient deep learning models, and balancing performance with computational and resource constraints. It provides valuable insights for both academic research and practical applications in various fields such as computer vision, natural language processing, and more.

Theoretical Background

Overview of Deep Learning and CNNs: Convolutional Neural Networks (CNNs) are designed to process data with a grid-like topology, such as images (Krizhevsky, Sutskever, & Hinton, 2012). They consist of multiple layers, including convolutional layers that apply filters to input data to extract features. The hierarchical structure of CNNs allows them to learn low-level features such as edges in early layers and high-level features such as objects in deeper layers (Zeiler & Fergus, 2014).

Importance of Layer Depth: The depth of a CNN, determined by the number of convolutional layers, affects its ability to learn complex features. Deeper networks can capture more intricate patterns but also pose challenges in terms of training and computational efficiency. He et al. (2016) introduced residual connections in ResNet to address the vanishing gradient problem in deep networks, demonstrating that deeper architectures could achieve better performance when appropriately managed.

Model Efficiency Metrics: Model efficiency in deep learning refers to the balance between the model's performance and the computational resources required. Metrics include accuracy, inference time, and resource usage such as memory and processing power. Efficient models are those that achieve high accuracy with minimal computational overhead (Han et al., 2016).

Factors Affecting Efficiency: Several factors influence model efficiency, including:

- **Layer Depth**: More layers can capture more features but also increase computational complexity (Simonyan et al. 2015).
- **Layer Width**: Wider layers can learn more features at each level but require more computational resources.
- **Filter Size**: Larger filters can capture broader patterns but increase the computational cost (Long et al. 2015).
- **Pooling and Stride**: Techniques like pooling reduce dimensionality and computational load but can also lead to information loss (Springenberg et al., 2014).

Methodology: Algorithm complexity in CNNs can be measured using metrics like computational cost (FLOPs - floating point operations), memory usage, and training/inference time. The complexity grows with the number of layers and the size of the filters used in each layer (Tan & Le, 2019). Increasing the number of layers typically enhances the model's capacity to learn but also increases the computational complexity, leading to higher resource consumption and longer training times (Szegedy et al., 2015). Deeper networks require more floating-point operations, which translates to greater energy consumption and longer inference times.

TABLE 1: Concise Complexity Comparison Table of Related Works

SN	Model	Time Complexity	Space Complexity	Default Layer depth
1	**ResNet-18**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**18** convolutional layers
2	**ResNet-50**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**50** convolutional layers
3	**VGG-16**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**13** convolutional layers
4	**TBNet**	Proprietary; typically optimized for time and space efficiency	Proprietary; typically optimized for time and space efficiency	Specific implementations of TBNet can vary, so the exact number of convolutional layers is not standardized.
5	**ChexNet (DenseNet-121)**	Based on DenseNet; time complexity similar to DenseNet	Based on DenseNet; time complexity similar to DenseNet	**57** convolutional layers, 121 layers total, with
6	**NASNetMobile**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**27** convolutional layers
7	**EfficientNet (B0)**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**16** convolutional layers
8	**Xception**	$O(N \cdot K^2 \cdot H \cdot W \cdot (C_{in} + C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**36** convolutional layers
9	**Inception V3**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**48** convolutional layers
10	**DenseNet**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**121** layers total, with **57** convolutional layers
11	**MobileNetV2**	$O(N \cdot K^2 \cdot H \cdot W \cdot (C_{in} + C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**53** convolutional layers
12	**AlexNet**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**5** convolutional layers
13	**ShuffleNet V2 (1.0x)**	$O(N \cdot K^2 \cdot H \cdot W \cdot (C_{in} + \frac{C_{out}}{g})$	$O(N \cdot H \cdot W \cdot C_{out})$	**86** convolutional layers
14	**SqueezeNet**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**26** convolutional layers
15	**ResNeXt**	$O(N \cdot K^2 \cdot H \cdot W \cdot (C_{in} + \frac{C_{out}}{g})$	$O(N \cdot H \cdot W \cdot C_{out})$	**50** convolutional layers
16	**Wide ResNet-50-2**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**50** convolutional layers
17	**SE-ResNet (50)**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**50** convolutional layers
18	**RegNet (Y-200MF)**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**12** convolutional layers
19	**MNasNet**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**54** convolutional layers
20	**GhostNet**	$O(N \cdot K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$	$O(N \cdot H \cdot W \cdot C_{out})$	**54** convolutional layers

K: Kernel (or Filter) size
N: Number of input channels :
C_{out}: Number of output channels
C_{in}: Number of input channels
H: Height of the input feature map
W: Width of the input feature map

Estimation of Parameters Count from Algorithm Complexity

In the context of neural networks, parameters are the internal variables that the model learns from the training data. These parameters are crucial for the network to make predictions and include the following:

1. **Weights**: These are the main parameters in neural networks. Each connection between neurons has an associated weight that determines the strength and direction of the connection. During training, the model adjusts these weights to minimize the error in its predictions.
2. **Biases**: Bias terms are added to the weighted sum of inputs to a neuron. They allow the activation function to be shifted to the left or right, which helps the model fit the training data better.

There are different types of parameters and this can be viewed from the perspective of Convolutional layers, fully connected layers, and Batch normalization layers. **Convolutional Layers:** Convolutional weights and it biases are primary in the evaluation of the complexity of a neural network. **Weights**: These are the filters (kernels) used in the convolution operation. Each filter has weights that are learned during training. **Biases**: Each filter can have an associated bias term. **Fully Connected (Dense) Layers Weights**: Each neuron in a fully connected layer is connected to every neuron in the previous layer, with a corresponding weight for each connection. **Batch Normalization Layers**: These parameters are learned during training to normalize the input to each layer and maintain the expressiveness of the network.

Parameters counts is given significant preference in algorithm complexity analysis for the following reasons:

1. **Learning**: The process of training a neural network involves adjusting these parameters to minimize the loss function, which measures how well the network's predictions match the actual data.
2. **Model Complexity**: The number of parameters affects the model's capacity to learn and generalize from the data. Too few parameters can lead to underfitting, while too many parameters can lead to overfitting.
3. **Computational Resources**: The number of parameters determines the amount of memory and computation required for training and inference. Models with a large number of parameters require more powerful hardware and longer training times.

Parameters Count for ResNet-18

ResNet-18 is designed with fewer layers compared to most of the other deeper convolutional neural network (CNN) architectures. It uses residual blocks that help in maintaining a **low time complexity** while being able to train deeper models effectively (He et. Al., 2015).

- Each convolutional layer follows the typical convolution complexity $O(N\ .\ K^2 \cdot C_{in} \cdot H \cdot W \cdot C_{out})$
- The residual connections (element-wise addition) are efficient and have a lower time complexity due to it fewer convolutional layers

ResNet-18 is more memory efficient than most other deeper convolutional neural network, making it suitable for resource-constrained environments (He et. al., 2016). Feature maps and convolutional weights dominate the **space complexity**, which can be expressed as:

$$O(N\ .\ H \cdot W \cdot C_{out})$$

Layer-wise Breakdown

1. **Initial Convolutional Layer**:
   - 7x7 convolution, 64 filters, stride 2
   - Complexity: $O\ (7\times7\times3\times64\times\frac{224\ X\ 224}{2\ X2}) = O\ (7\times7\times3\times64\times112\times112)$
2. **Max Pooling Layer**:
   - 3x3 max pooling, stride 2
   - Complexity: Not computationally expensive compared to convolutions
3. **Residual Blocks (Convolutional Layers)**:
   - Each block has two 3x3 convolutional layers
   - There are 4 stages, each stage has 2 residual blocks:
     - Stage 1: 64 filters, input size 56x56
     - Stage 2: 128 filters, input size 28x28
     - Stage 3: 256 filters, input size 14x14
     - Stage 4: 512 filters, input size 7x7

Stage-wise Complexity:

Stage 1 (4 layers, 64 filters, 56x56): 2×(3×3×64×64×56×56)×2

Stage 2 (4 layers, 128 filters, 28x28): 2×(3×3×64×128×28×28)×2

Stage 3 (4 layers, 256 filters, 14x14): 2×(3×3×128×256×14×14)×2

Stage 4 (4 layers, 512 filters, 7x7): 2×(3×3×256×512×7×7)×2

Fully Connected Layer:

- Output: 1000 classes
- Complexity: O(512×1×1×1000) = O(512×1000)

Total Computational Complexity

Summing up the complexities of all layers, the overall computational complexity of ResNet-18 can be approximated as:

$O(7\times7\times3\times64\times112\times112)+\sum_{I=1}^{4} O\ (2\times3\times3\times C_{in}\times C_{out}\times H\times W\times2)+O(512\times1000)$

Where $\mathbf{C_{in}}$ and $\mathbf{C_{out}}$ are the input and output channel dimensions, and **H** and **W** are the spatial dimensions of the feature maps at each stage.

Parameters Count

1. **Initial Convolutional Layer**: 7×7×3×64=9,4087
2. **Residual Blocks**:
   - Stage 1: 2×(3×3×64×64)×2=147,456
   - Stage 2: 2×(3×3×64×128+3×3×128×128)×2=1,179,648
   - Stage 3: 2×(3×3×128×256+3×3×256×256)×2=4,718,592
   - Stage 4: 2×(3×3×256×512+3×3×512×512)×2=18,874,368
3. **Fully Connected Layer**: 512×1000+1000(bias)=513,000512 \times 1000 + 1000 (bias) = 513,000512×1000+1000(bias)=513,000

Total Parameters: Approximately 11.7 million parameters.

Summary

- **Computational Complexity**: Dominated by the convolutional layers, especially in deeper stages.
- **Parameters Count**: About 11.7 million parameters.
- **Main Contributors**: Convolutional layers in the residual blocks are the primary contributors to both computational complexity and parameter count.

Parameters Count for ResNet-50

ResNet-50 is a deeper version of ResNet-18 with 50 layers.

Architecture Breakdown

- **Initial Layers**:
  - 7x7 convolution, 64 filters, stride 2
  - 3x3 max pooling, stride 2
- **Residual Blocks**:
  - Bottleneck design: 1x1, 3x3, 1x1 convolutions
  - 4 stages with the following configurations:
    - Stage 1: 3 blocks, 256 output filters
    - Stage 2: 4 blocks, 512 output filters
    - Stage 3: 6 blocks, 1024 output filters
    - Stage 4: 3 blocks, 2048 output filters
- **Fully Connected Layer**:
  - Output: 1000 classes

Complexity

1. **Initial Convolutional Layer**:
   - 7×7×3×64×112×112=38,943
2. **Residual Blocks**:
   - **Stage 1** (3 blocks, each with 1x1, 3x3, 1x1): 3×(1×1×64×64×56×56+3×3×64×64×56×56+1×1×64×256×56×56)
   - **Stage 2** (4 blocks): 4×(1×1×256×128×28×28+3×3×128×128×28×28+1×1×128×512×28×28)
   - **Stage 3** (6 blocks): 6×(1×1×512×256×14×14+3×3×256×256×14×14+1×1×256×1024×14×14)
   - **Stage 4** (3 blocks): 3×(1×1×1024×512×7×7+3×3×512×512×7×7+1×1×512×2048×7×7)
3. **Fully Connected Layer**:
   - 2048×1000=2,048,0002048

Total Parameters: ~25.6 million.

Parameters Count for VGG-16

VGG-16 is a convolutional neural network with 16 layers.

Architecture Breakdown

- **Convolutional Layers**:
  - 13 convolutional layers in 5 blocks
  - Each block followed by a max pooling layer
- **Fully Connected Layers**:
  - 3 fully connected layers

Complexity

1. **Convolutional Layers**:
   - Block 1: 2×(3×3×3×64×224×224)=17,251,5842
   - Block 2: 2×(3×3×64×128×112×112)=38,707,200
   - Block 3: 3×(3×3×128×256×56×56)=132,710,400
   - Block 4: 3×(3×3×256×512×28×28)=265,420,800
   - Block 5: 3×(3×3×512×512×14×14)=132,710,400
2. **Fully Connected Layers**:
   - FC1: 512×7×7×4096=102,760,448
   - FC2: 4096×4096=16,777,216
   - FC3: 4096×1000=4,096,000

Total Parameters: ~138 million.

Parameters Count for EfficientNet-BO

EfficientNet-B0

EfficientNet-B0 is designed for both accuracy and efficiency.

Architecture Breakdown

- **Mobile Inverted Bottleneck Convolution (MBConv) blocks**
  - 7 stages with different configurations and expansion factors

Complexity

- Complexity depends on the depth and expansion factor for each block.
- Example stage calculation:
  - Stage 1: 3×3×32×16×2242×2242=3,154,176

Total Parameters: ~5.3 million.

Parameters Count for Xception

Xception uses depthwise separable convolutions.

Architecture Breakdown

- **Entry Flow**: Series of separable convolutional layers
- **Middle Flow**: 8 blocks of separable convolutional layers
- **Exit Flow**: Separable convolutional layers and fully connected layers

Complexity

Separable Convolutions:

- Complexity significantly reduced compared to standard convolutions
- Example stage calculation:
  - 3×3×64×1×64×56×56=12,582,912

Total Parameters: ~23 million.

Parameters Count for Inception V3

Inception V3 uses inception modules for efficient computation.

Architecture Breakdown

- **Inception Modules**:
  - Modules with different filter sizes to capture different features

Complexity

Inception Modules:

- Combination of convolutions with different kernel sizes
- Example module calculation:
  - 1×1,3×3,5×51 convolutions combined

Total Parameters: ~23.8 million.

Here is a concise summary of the parameter counts:

- **ResNet-18**: ~11.7 million parameters
- **ResNet-50**: ~25.6 million parameters
- **VGG-16**: ~138 million parameters
- **EfficientNet-B0**: ~5.3 million parameters
- **Xception**: ~23 million parameters
- **Inception V3**: ~23.8 million parameters

Comparing table 1 and table 2, it can be seen that there is an inverse relationship between parameter count and efficiency of a deep learning model due to the depth of their convolutional layers.

TABLE 2: Systematic Review of Published Research on Tuberculosis Detection Using Convolutional Neural Networks

SN	Author(s)	Title	Architecture	Accuracy	Precision	Recall	F1_Score
1	Ahsan et al. (2019)	Application of a Convolutional Neural Network using transfer learning for tuberculosis detection	VGG	81.25%	94.1%	87.3%	---
2	Wong et al. (2022)	TB-Net: A Tailored, Self-Attention Deep Convolutional Neural Network Design for Detection of Tuberculosis Cases from Chest X-ray Images	TB-Net	99.86%	99.71%	100.0%	---
			ChexNet	99.42%	100%	98.85%	---
			NASNetMobile	99.28%	91.84%	99.71%	---
			EfficientNetB0	99.98%	99.42%	98.56%	---
3	Showkatianet al. (2022)	Deep learning-based automatic detection of tuberculosis disease in chest X-ray images	ConvNet	0.87%	0.88%	0.87%	0.87%
			Exception	0.91%	0.91%	0.91%	0.91%
			Inception_V3	0.88%	0.88%	0.88%	0.88%
			ResNet50	0.90%	0.91%	0.91%	0.91%
			VGG16	0.90%	0.91%	0.91%	0.91%
4	Sharma et al. (2023)	Deep learning models for tuberculosis detection and infected region visualization in chest X-ray images	Xception	99.29%,	99.30%	99.29%	99.29%

To further validate our assertion of fewer convolutional layers for efficiency of deep learning model, **aside the algorithm complexity and parameters counts**, this research also applied and experimental setup of a proposed Deep Convolution Neural Network which employed a pre-trained ResNet-18 architecture to develop a model for tuberculosis detection in chest X-Ray (CXR) images with the mindset of having an efficient model with small sized (hidden) convolutional layers and the right hyperparameters. The strength of this approach also includes improved latency achieved by ResNet blocks which uses **Identity shortcut** and **Projection shortcut** in the basic building blocks of ResNet-18, this block is repeated throughout the network to prevent vanishing gradient problem during backpropagaton and also improve computational and time complexity which in turn guarantee efficiency of the deep learning algorithm compared to the related work in table 2. The ResNet-18 deep learning model achieved an overall model accuracy of **99.5%**,**99.2%** precision, **99.7%** Recall and **99.5%** F1_Score having been trained with 900 images and validated 600 images of pre-processed chest x-ray (CXR). The following hyperparameter tuning was adopted in the training and validation of the ResNet-18 model:

DISCUSSION ON CONVOLUTIONAL NEURAL NETWORK VARYING LAYER DEPTH AND PARAMETERS COUNT

Synthesis of Key Points: The number of convolutional layers in a CNN significantly impacts model efficiency. While deeper networks can achieve higher accuracy, they require more computational resources and time. The balance between depth and efficiency is crucial for designing practical models.

Implications for Practice: Practitioners need to balance the benefits of deeper networks with the practical constraints of available computational resources. Techniques such as pruning, quantization, and efficient architecture design can help manage complexity (Han et al., 2016). Additionally, employing architectural innovations like residual connections and inception modules can allow for deeper networks without a proportional increase in complexity (Szegedy et al., 2015).

Future Directions: Future research should focus on developing methods to optimize the number of layers and overall architecture for specific tasks, considering both performance and resource efficiency.

Adaptive architectures that can dynamically adjust their depth based on the complexity of the input data and computational constraints are a promising area of study (Tan & Le, 2019). For future research, Artificial Intelligence Experts may explore the possibility of developing adaptive Architectures that can dynamically adjust their depth based on the complexity of the input data and computational constraints. As deep learning continues to evolve, understanding the trade-offs between model depth and efficiency will be crucial for developing more effective and practical AI solutions. The insights gained from this study can guide the design of future architectures that balance performance with computational efficiency.

Summary and Conclusion: This article highlights the importance of the number of convolutional layers in determining the efficiency of deep learning models. A careful balance between depth and computational complexity is essential for optimal performance.

Recommendations: Future studies should explore adaptive architectures that can dynamically adjust their depth based on the complexity of the task and available resources. Further research into methods for reducing the computational overhead of deep networks while maintaining high performance is essential.

REFERENCES

Ahsan, M., Gomes, R., Denton, A., 2019. "Application of a Convolutional Neural Network using transfer learning for tuberculosis detection" 2019 IEEE:978-1-7281-0927-5/19

Han, S., Mao, H., and Dally, W. J., 2016. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. International Conference on Learning Representations (ICLR).

He, K., Zhang, X., Ren, S., and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 1026-1034.

He, K., Zhang, X., Ren, S., and Sun, J., 2016. Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012. ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems (NIPS), 1097-1105.

LeCun, Y., Bengio, Y., and Hinton, G., 2015. Deep learning. Nature, 521(7553), 436-444.

Long, J., Shelhamer, E., and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 3431-3440.

Sharma, V., Gupta, S. K., Shukla, K. K., 2023. "Deep learning models for tuberculosis detection and infected region visualization in chest X-ray images" Intelligent Medicine, DOI:10.1016/j.imed.2023.06.001, LicenseCC BY-NC-ND 4.0

Showkatian, E., Salehi, M., Ghaffari, H., Reiazi, R., Sadighi, N., 2022. "Deep learning-based automatic detection of tuberculosis disease in chest X-ray images" Polish Journal of Radiology DOI: 10.5114/pjr.2022.113435

Simonyan, K., and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations (ICLR).

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M., 2014. Striving for simplicity: The all convolutional net. arXiv preprint arXiv:1412.6806.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., and Rabinovich, A., 2015. Going deeper with convolutions. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9.

Tan, M., and Le, Q., 2019. EfficientNet: Rethinking model scaling for convolutional neural networks. Proceedings of the International Conference on Machine Learning (ICML), 6105-6114.

Wong, A., Ren, H. L. J., Rahmat-Khah, H., Sabri, A., Alaref, A., Liu, H., 2022. "TB-Net: A Tailored, Self-Attention Deep Convolutional Neural Network Design for Detection of Tuberculosis Cases from Chest X-ray Images" Computerized Medical Imaging and Graphics DOI: 10.3389/frai.2022.827299

Zeiler, M. D., and Fergus, R., 2014. Visualizing and understanding convolutional networks. European Conference on Computer Vision (ECCV), 818-833.