Journal of Industrial Information Technology and Application *ISSN(Online): 2586-0852*
Vol.9 No. 4

Comparative Analysis of Deep Learning Architectures for Meat Freshness Classification

Fernando Quiroz, Jr.[1], Robert Roxas [2], and Edison Ralar [1,]*

[1] School of Tech. and Computer Studies. Biliran Province State University, Philippines
[2] College of Science. University of the Philippines Cebu, Philippines

Abstract. This research presents a comparative analysis of various deep learning architectures for the classification of meat freshness and address a critical issue in the Philippine meat market where consumers often struggle to assess meat quality accurately. The study evaluates six (6) prominent convolutional neural network (CNN) architectures: LeNet, AlexNet, VGGNet, GoogleNet, ResNet, and DenseNet, using a manually built dataset, categorized into fresh and non-fresh meat images. Each model was trained and evaluated based on accuracy, loss, and training efficiency, with a focus on practical applicability in local markets. Results indicate that LeNet achieved the highest validation accuracy (76.07%) and training efficiency, requiring only 8 epochs and 849 seconds for training. In contrast, more complex models like VGGNet and DenseNet exhibited significantly longer training times and lower accuracy, raising concerns about their practicality for real-time freshness detection.

Keywords; computer vision, deep learning, meat freshness, neural network

Cite this paper as: Fernando Quiroz, Jr., Robert Roxas, and Edison Ralar (2025) “Comparative Analysis of Deep Learning Architectures for Meat Freshness Classificaiton”, Journal of Industrial Information Technology and Application, Vol. 9. No. 4, pp. 1209 - 1219

* Corresponding author: evan.quiroz@bipsu.edu.ph
Received: Aug. 22. 2025 Accepted: Oct. 12. 2025 Published: Dec. 31. 2025

1. Introduction

The meat market in the Philippines is a critical component of the country's economy, providing essential protein sources for millions of Filipinos. According to the Philippine Statistics Authority (PSA), the livestock and poultry sector produced 230.11 and 798.11 metric tons, respectively in the third quarter of 2024. In addition, pork is one of the most consumed meat products with 403,000 metric tons consumed per capita. Biliran Island, known for its agricultural activities, has a growing meat industry that supports local livelihoods and contributes to food security. In its 2016 report, PSA recorded 6.735 kilograms per capita consumption in the province. Despite the abundance of meat resources, ensuring the freshness and quality of meat products presents a significant challenge for consumers in Biliran Island. Many consumers lack the knowledge and tools to accurately assess meat freshness, often relying on visual cues that can be misleading. Additionally, limited access to advanced preservation technologies and inadequate cold chain logistics worsens the problem and may lead to a higher likelihood of purchasing spoiled or subpar meat. This not only compromises consumer health but also diminishes trust in local markets and affects purchasing decisions and overall satisfaction with meat products. This research aims to address these challenges by exploring the application of transfer learning techniques in developing a practical solution for meat freshness detection and contribute to improved consumer safety and market sustainability.

2. Review of Literature

Advancements of technology have led to development of various methods for assessing meat freshness, ranging from traditional sensory evaluation to advanced technological approaches. A review on different approaches in meat freshness detection highlighted the use of electrochemical, electronic nose and gas sensors, and optical biosensors [1], which are costly to purchase. Other studies have explored the use of advanced sensors, such electrochemistry sensors [2], fluorescence screening [3] and imaging technology [4].

In recent years, studies have used computer vision techniques for a low cost detection of meat freshness [5], [6], [7]. A study employed colorimetric sensor to tracks CO_2 levels increase with bacterial population and analyze the grey scale from the RGB space color [8]. The use of genetic algorithm for feature extraction and artificial neural network for prediction of chicken meat freshness were also experimented and gained promising results [9].

Despite these current solutions, many of these methods require specialized equipment and expertise, which are not readily available in many local markets. Additionally, while some studies have explored the integration of computer vision techniques for meat freshness detection, there remains a significant gap in the literature regarding their practical application since computer vision tasks require large amount of data and high computing resources.

At present, there are six (6) widely-used deep learning architectures for computer vision primarily based on convolutional neural network (CNN). LeNet was one of the first convolutional neural networks designed for handwritten digit recognition consisting of seven layers with convolutional and subsampling layers [10]. Following this, AlexNet won the ImageNet competition and popularized deep learning in computer vision with its eight-layer architecture, use of ReLU activation functions, and dropout for regularization [11]. VGGNet is known for its simplicity and depth, employing small 3x3 convolutional filters stacked to create deeper networks which has influenced many subsequent models by emphasizing the importance of depth in improving performance [12]. In 2014, GoogLeNet introduced the inception module, allowing for multi-scale feature extraction within the same layer, which improved the model's ability to learn complex patterns while reducing the number of parameters [13]. ResNet further advanced the field by introducing residual connections and by enabling the training of very deep networks without the vanishing gradient problem, thus setting new benchmarks in image classification [14].DenseNet built upon the concept of residual connections by connecting each layer to every other layer, promoting feature reuse and improving gradient flow [15]. This research aims to apply the different deep learning architectures for the classification of meat freshness through comparative analysis of their performances.

3. Methodology

*A. **Dataset***

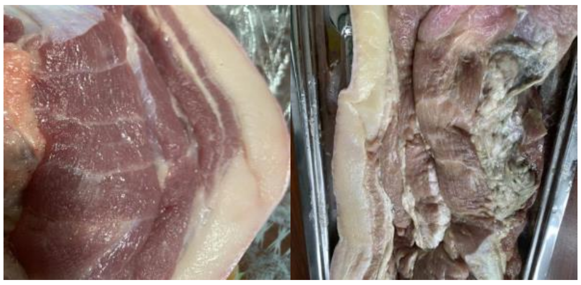

Figure 1. Sample meat images from the initial dataset. Left image: fresh; Right image: non-fresh

The dataset was manually gathered from a local market. The images were categorized into subdirectories (fresh and non-fresh), with each subdirectory representing a different class for binary classification, as shown in Figure 1.

A data augmentation technique was used and configured to perform several transformations on the images, including normalization of pixel values by rescaling them to the range [0, 1]. Additionally, 20% of the dataset was allocated for validation to ensure that the model's performance could be evaluated on unseen data. The augmentation techniques included random rotations (up to 20 degrees), width and height shifts (up to 20% of the total width and height), shear transformations, zooming, and horizontal flipping. The training and validation sets were then loaded and automatically labels the images based on their respective subdirectory names and resizes them to a target size of 150x150 pixels.

B. ***Model Training and Evaluation***

Each of the deep learning architecture was implemented using *Tensorflow* and *Keras*, and was compiled using *Adam* optimizer, which is known for its efficiency in training deep learning models [16]. The loss function used was binary cross-entropy, suitable for binary classification problems, and the model's performance was monitored using accuracy as a metric.

To prevent overfitting, early stopping callbacks were configured to monitor the validation loss, with a patience of 10 epochs, meaning that training would stop if the validation loss did not improve for 10 consecutive epochs. This approach helps to retain the best model weights and avoid overfitting to the training data. Additionally, model checkpoint was used to save the best-performing model based on validation loss. *TensorBoard* was also setup to visualize training progress, which aids in monitoring metrics such as loss and accuracy over time.

The model was trained with the training and validation data provided with a maximum of 100 epochs. After training, each model was evaluated on the validation dataset, using the validation loss and accuracy. Computed as well the *F1* score to assess the models' classification performance and constructed a confusion matrix to visualize the model's performance across the different classes.

The LeNet architecture was configured starting with an input layer designed to accept *RGB* images of size 150x150 pixels. It features three convolutional layers, comprising 32, 64, and 128 filters, respectively, each utilizing a 3x3 kernel and *ReLU* activation function to facilitate feature extraction. Each convolutional layer is followed by a max pooling layer with a 2x2 pooling size, which serves to down sample the feature maps, thereby reducing spatial dimensions while preserving critical information. After the convolutional and pooling operations, the output is flattened into a one-dimensional vector, which is then processed by a fully connected layer containing 128 units and employing *ReLU* activation. The architecture concludes with a final dense layer featuring a single unit and a sigmoid activation function, which outputs probabilities for binary classification.

AlexNet is designed to accept input images of size 150x150 pixels with three color channels (RGB). The architecture begins with the first convolutional layer, which employs 96 filters of size 11x11 and a stride of 4 to capture large features from the input images while significantly reducing the spatial dimensions. This is followed by a max pooling layer with a pool size of 3x3 and a stride of 2, which further down-samples the feature maps. The second convolutional layer consists of 256 filters of size 5x5, utilizing 'same' padding to maintain the spatial dimensions, followed by another max pooling layer. Subsequent layers include two convolutional layers with 384 filters of size 3x3, both using same padding to allow the model to learn more complex features. The fifth convolutional layer, which has 256 filters, is again followed by a max pooling layer. After the convolutional and pooling operations, the output is flattened into a one-dimensional vector, which is then processed through three fully connected layers. The first two fully connected layers each contain 4096 units and employ ReLU activation, with dropout layers added to mitigate overfitting by randomly setting a fraction of the input units to zero during training. The final output layer consists of a single unit with a sigmoid activation function.

VGGNet accepts input images of size 150x150 pixels with three color channels (RGB) and utilized a sigmoid activation function in the output layer. The architecture is organized into five distinct blocks, each comprising convolutional layers followed by max pooling layers. In **Block 1,** the model begins with two convolutional layers, each with 64 filters of size 3x3 and ReLU activation, both using same padding to preserve spatial dimensions. This is followed by a max pooling layer with a pool size of 2x2 and a stride of 2, which reduces the spatial dimensions of the feature maps. **Block 2** follows a similar structure, with two convolutional layers of 128 filters, again using 3x3 kernels and ReLU activation, followed by another max pooling layer. Block 3 consists of three convolutional layers, each with 256 filters, allowing the model to learn more complex features. This is followed by a max pooling layer to down sample the output. Block 4 and Block 5 mirror this structure, with three convolutional layers of 512 filters in each block. Each convolutional layer employs ReLU activation and same padding. After the convolutional and pooling operations, the output is flattened into a one-dimensional vector, which is then processed through three fully connected layers. The first two fully connected layers each contain 4096 units and utilize ReLU activation, allowing for complex decision-making based on the learned features. The final output layer consists of a single unit with a sigmoid activation function, suitable for binary classification tasks.

GoogleNet, also known as Inception V1, s configured for binary classification tasks, accepting input images of size 150x150 pixels with three color channels (RGB) and used a sigmoid activation function in the output layer, as shown in Figure 5.. The architecture begins with an initial convolutional layer that employs 64 filters with a 7x7 kernel size and a stride of 2, followed by a max pooling layer to reduce spatial dimensions. This is succeeded by another convolutional layer with 192 filters and a subsequent max pooling layer. The core of the GoogleNet architecture consists of multiple inception modules, which allow the model to capture multi-scale features by applying different convolutional filter sizes in parallel. Each inception module includes a combination of 1x1, 3x3, and 5x5 convolutions, as well as a max pooling operation, all of which are concatenated along the channel dimension to form a rich feature representation. The model incorporates several inception modules, each with varying filter configurations to learn complex patterns and features from the input images. After processing through the inception modules, the output is passed through a global average pooling layer, which reduces the spatial dimensions to a single value per feature map, effectively summarizing the learned features. Finally, the output is fed into a fully connected layer with a single unit and a sigmoid activation function, suitable for binary classification tasks.

ResNet is designed to accepting input images of size 150x150 pixels with three color channels (RGB). The architecture begins with an initial convolutional layer that employs 64 filters with a 7x7 kernel size and a stride of 2, followed by batch normalization and ReLU activation to enhance training stability and introduce non-linearity. This is followed by a max pooling layer to reduce the spatial dimensions of the feature maps. The core of the ResNet architecture consists of multiple residual blocks, each designed to learn residual mappings rather than the original unreferenced mappings. Each residual block includes two convolutional layers, each followed by batch normalization and ReLU activation. The input to the block (the shortcut connection) is added to the output of the second convolutional layer to allow gradients flow more easily during backpropagation and mitigating the vanishing gradient problem. The model features a series of residual blocks organized into four layers, with 3, 4, 6, and 3 blocks in each layer, respectively. After each set of residual blocks, a max pooling layer is applied to further down sample the feature maps. This structure enables the network to learn increasingly complex features while maintaining a manageable number of parameters. After processing through the residual blocks, the output is passed through a global average pooling layer, which reduces the spatial dimensions to a single value per feature map, effectively summarizing the learned features. Finally, the output is fed into a fully connected layer with a single unit and a sigmoid activation function, suitable for binary classification tasks.

DenseNet is designed to accept input images of size 150x150 pixels with three color channels (RGB) and utilizing a sigmoid activation function in the output layer. The architecture begins with an initial convolutional layer that employs 64 filters with a 7x7 kernel size and a stride of 2, followed by batch normalization and ReLU activation to enhance training stability. This is followed by a max pooling layer to reduce the spatial dimensions of the feature maps. The core of the DenseNet architecture consists of multiple dense blocks, where each block contains a specified number of layers. Within each dense block, a series of operations are performed: a 1x1 convolution (bottleneck layer) is applied to reduce dimensionality, followed by a 3x3 convolution to extract features. The output of each convolutional layer is concatenated with the input of the block, allowing for the accumulation of features from all preceding layers. After each dense block, a transition layer is applied (except after the last block), which consists of a 1x1 convolution to reduce the number of feature maps, followed by average pooling to down sample the feature maps. This transition helps to control the model's complexity and reduce the spatial dimensions while maintaining important features. The architecture allows for a high degree of feature reuse, as each layer receives input from all preceding layers, which mitigates the vanishing gradient problem and enhances gradient flow during training. After processing through the dense blocks and transition layers, the output is passed through a global average pooling layer, which reduces the spatial dimensions to a single value per feature map, effectively summarizing the learned features. Finally, the output is fed into a fully connected layer with a single unit and a sigmoid activation function.

4. Results and Discussion

As shown in Table 1, among the models evaluated, LeNet emerged as the most effective model in terms of training accuracy (91.76%) and validation accuracy (76.07%), which indicates its capability to generalize well to unseen data. However, its validation loss of 51.62% suggests that while it performs well, there is still room for improvement, particularly in reducing overfitting. The standard deviation of 20.40 in training accuracy indicates a moderate level of variability, suggesting that while LeNet is generally reliable, its performance may fluctuate depending on the specific training samples used. The variance of 416.02 further emphasizes this variability, highlighting the need for careful selection of training data to ensure consistent performances.

Table 1. Accuracy and loss results.

Deep Learning Model	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
LeNet	91.76%	76.07%	19.94%	51.62%
AlexNet	51.34%	50.55%	69.33%	69.31%
VGGNet	47.32%	50.74%	69.33%	69.31
GoogleNet	84.29%	67.83%	39.13%	61.18%
ResNet	95.53%	72.98%	11.77%	61.78%
DenseNet	95.57%	74.08%	11.49%	66.14%
SD (σ)	**20.40**	**10.71**	**24.74**	**6.10**
Variance (σ^2)	**416.02**	**114.66**	**612.20**	**37.25**

Moreover, LeNet stands out as the most efficient model, requiring only 8 optimal epochs and a total training time of 849 seconds, with an average training time of 47.16 seconds per epoch, as shown in Table 2. This efficiency makes LeNet particularly appealing for applications where quick training and inference are essential, such as real-time monitoring of meat freshness. The low training time combined with relatively high accuracy suggests that LeNet is well-suited for this specific task, allowing for rapid deployment in practical scenarios.

Table 2. Accuracy and loss results.

Deep Learning Model	Optimal Epoch	Total Training Time (s)	Average Training Time (s)
LeNet	8	849	47.16
AlexNet	19	1546	53.31
VGGNet	3	8552	657.85
GoogleNet	7	1201	70.65
ResNet	10	2175	108.75
DenseNet	12	55773	2535.14
SD (σ)	**4.95**	**19891.96**	**901.06**
Variance (σ^2)	**24.47**	**395690000**	**811909.89**

In contrast, AlexNet, while requiring more epochs (19) and a total training time of 1,546 seconds, still maintains a reasonable average training time of 53.31 seconds per epoch. However, its performance metrics were significantly lower than those of LeNet, indicating that the additional training time did not translate into improved accuracy for this specific application. This raises questions about the efficiency of using more complex architectures when simpler models can achieve comparable or superior results. VGGNet, despite converging in just 3 epochs, exhibited an extraordinarily high total training time of 8,552 seconds and an average training time of 657.85 seconds per epoch. This inefficiency highlights a significant drawback of VGGNet, as the extensive computational resources required may limit its practicality in real-world applications, especially when faster alternatives like LeNet are available. The long training time, coupled with its lower accuracy, suggests that VGGNet may not be the best choice for meat freshness classification.

GoogleNet required 7 epochs and a total training time of 1,201 seconds, with an average training time of 70.65 seconds per epoch. While its training time is more manageable than that of VGGNet, it still reflects a trade-off between complexity and performance. The results indicate that GoogleNet may offer some advantages in feature extraction but at the cost of increased training time. ResNet and DenseNet, while achieving high training accuracies, required significantly more training time, with ResNet taking 2,175 seconds and DenseNet an astonishing 55,773 seconds. The average training times of 108.75 seconds and 2,535.14 seconds per epoch, respectively, highlight the computational intensity of these architectures. The extensive training times may pose challenges for practical implementation, particularly in environments where rapid decision-making is crucial. The standard deviation (SD) and variance of the training times further emphasize the variability in the efficiency of these models. The high SD of 19,891.96 seconds and variance of 395,690,000 indicate that while some models are efficient, others, particularly DenseNet, exhibit extreme variability in training time, which could complicate resource planning and deployment strategies.

5. Conclusion and Recommendation

This comparative analysis of deep learning architectures for meat freshness classification shows the effectiveness of simpler models, particularly LeNet, in achieving high accuracy with significantly reduced training times. The findings suggest that while more complex architectures like ResNet and DenseNet may offer superior performance in certain contexts, their extensive computational requirements limit their practicality for real-time applications. Therefore, it is recommended to consider adopting LeNet or similar lightweight models for meat freshness detection systems. Further research should explore the integration of these models with mobile applications or low-cost imaging devices to empower consumers with accessible tools for assessing meat quality, ultimately enhancing food safety and consumer confidence in local meat products.

References

[1] K. H. Erna, K. Rovina, and S. Mantihal, “Current detection techniques for monitoring the freshness of meat-based products: A review,” *J Packag Technol Res*, vol. 5, no. 3, pp. 127–141, (2021).

[2] J. Johnson, D. Atkin, K. Lee, M. Sell, and S. Chandra, “Determining meat freshness using electrochemistry: Are we ready for the fast and furious?,” *Meat Sci*, vol. 150, pp. 40–46, (2019).

[3] H. Ye *et al.*, “Real-time fluorescence screening platform for meat freshness,” *Anal Chem*, vol. 94, no. 44, pp. 15423–15432, (2022).

[4] J. Zhang *et al.*, “Olfactory imaging technology and detection platform for detecting pork meat freshness based on IoT,” *Comput Electron Agric*, vol. 215, p. 108384, (2023).

[5] A. Taheri-Garavand, S. Fatahi, M. Omid, and Y. Makino, “Meat quality evaluation based on computer vision technique: A review,” *Meat Sci*, vol. 156, pp. 183–195, (2019).

[6] M. Modzelewska-Kapituła and S. Jun, “The application of computer vision systems in meat science and industry–A review,” *Meat Sci*, vol. 192, p. 108904, (2022).

[7] K. Lugatiman, C. Fabiana, J. Echavia, and J. J. Adtoon, “Tuna meat freshness classification through computer vision,” in *2019 IEEE 11th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM)*, (2019), pp. 1–6.

[8] I. M. P. de Vargas-Sansalvador, M. M. Erenas, A. Mart\'\inez-Olmos, F. Mirza-Montoro, D. Diamond, and L. F. Capitan-Vallvey, “Smartphone based meat freshness detection,” *Talanta*, vol. 216, p. 120985, (2020).

[9] A. Taheri-Garavand, S. Fatahi, F. Shahbazi, and M. de la Guardia, “A nondestructive intelligent approach to real-time evaluation of chicken meat freshness based on computer vision technique,” *J Food Process Eng*, vol. 42, no. 4, p. e13039, (2019).

[10] M. R. Islam and A. Matin, “Detection of COVID 19 from CT image by the novel LeNet-5 CNN architecture,” in *2020 23rd International Conference on Computer and Information Technology (ICCIT)*, (2020), pp. 1–5.

[11] A. Abd Almisreb, N. Jamil, and N. M. Din, “Utilizing AlexNet deep transfer learning for ear recognition,” in *2018 fourth international conference on information retrieval and knowledge management (CAMP)*, (2018), pp. 1–5.

[12] Z. Yang, “Classification of picture art style based on VGGNET,” in *Journal of Physics: Conference Series*, (2021), p. 12043.

[13] L. Yang *et al.*, “GoogLeNet based on residual network and attention mechanism identification of rice leaf diseases,” *Comput Electron Agric*, vol. 204, p. 107543, (2023).

[14] S. Targ, D. Almeida, and K. Lyman, "Resnet in resnet: Generalizing residual architectures," *arXiv preprint arXiv:1603.08029*, (2016).

[15] Y. Zhu and S. Newsam, "Densenet for dense flow," in *2017 IEEE international conference on image processing (ICIP)*, (2017), pp. 790–794.

[16] M. G. M. Abdolrasol *et al.*, "Artificial neural networks based optimization techniques: A review," *Electronics (Basel)*, vol. 10, no. 21, p. 2689, (2021).