Kalpa Publications in Computing

Volume 19, 2024, Pages 333–343

Proceedings of 6th International Conference
on Smart Systems and Inventive Technology

Coccidiosis Disease Classification Using VGG-16 model With MLops

Mrs. M.V. Ezhil Dyana[1] M.E., (Ph.D). , Rakesh S[2] and Shyamganesh V[3]

[1] Assistant Professor, Department of Computer Science and Engineering St. Joseph's Institute of Technology, Chennai-600119, India.

dyanaezhil@gmail.com

[2] UG Student, Department of Computer Science and Engineering St. Joseph's Institute of Technology, Chennai-600119, India.

rakesh200320@yahoo.com

[3] UG Student, Department of Computer Science and Engineering St. Joseph's Institute of Technology, Chennai-600119, India.

vshyamganesh02@gmail.com

Abstract

The development of Machine Learning (ML) and DevOps, often referred to as MLOps, has revolutionized the healthcare sector by offering efficient and scalable solutions for disease classification. Machine Learning models, particularly deep learning algorithms, have demonstrated remarkable performance in classifying diseases from various medical data modalities such as images, genomic sequences, electronic health records, and more.

Keywords: DevOps (Development and Operations), MLOps (Machine Learning Operations), Workflow.

1 Introduction

Image classification is a simple task in computer vision that involves assigning images to predefined categories. The main goal is to teach the learning model to recognize and label objects or patterns in images. Convolutional neural networks (CNNs) are essential for classifying images. and provides us with great results. After training, the model can classify new images that were not seen during inference.

Coccidiosis is a parasitic disease that affects a wide range of animals, including poultry, livestock, and even humans, and is caused by protozoa belonging to the phylum Apicomplexa. These microscopic parasites, specifically from the family Eimeriidae, inhabit the digestive tracts of their host animals, where they can cause a variety of health problems. Coccidiosis is particularly prevalent in agricultural settings, posing a significant challenge to poultry and lives. The lifecycle of MLOps is shown in Fig 1.

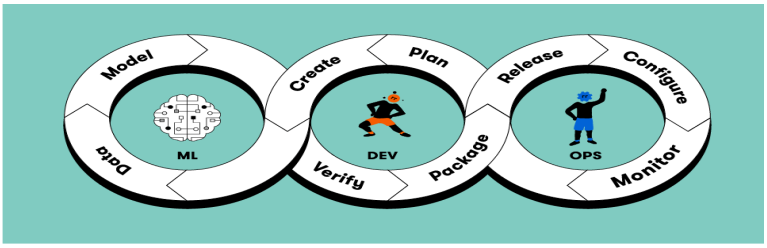


Figure 1: Lifecycle of MLOps

A collection of procedures and instruments known as MLOps (Machine Learning Operations) combine IT operations and the creation of machine learning (ML) systems[6-10]. The whole machine learning process is streamlined, allowing data science and production deployment to coexist.

2 Related Works

Brown A. et al. (2021) [1] proposed a literature review on “Computer Vision for Poultry Disease Detection: A Comparative Analysis”. Brown and a team of researchers conduct a comparative analysis of computer vision techniques for detecting diseases in chickens.

Chen W. et al. (2020) [2] proposed a literature review on "An Intelligent Monitoring System for Poultry House Environment and Disease Surveillance" in the Computers and Electronics in Agriculture journal.

Garcia L. et al. (2017) [3] proposed "A Cloud-Based Framework for PoultryDisease Monitoring and Management" in the International Journal of Distributed Sensor Networks.

Gupta S. et al. (2021) [4] proposed a review on “Poultry Disease Classification in Resource-Constrained Environments”. The study discusses strategies to make disease classification models accessible and affordable, ensuring that even small-scale poultry farmers can benefit from advanced technologies.

Johnson M. et al. (2020) [5] proposed a literature review on “Datasets for Poultry Disease Detection: Challenges and Opportunities”. Johnson and colleagues explore the datasets available for training and testing poultry disease classification models.

3 Methods

A. Data Ingestion

To ingest a dataset from GitHub into your Machine Learning pipeline, start by selecting the dataset and either cloning the repository or downloading it as a ZIP archive. Explore the dataset to understand its structure and contents, and preprocess it as necessary. If your pipeline involves distributed or cloud-based storage, synchronize the data to these environments as needed. Be aware that not all datasets on GitHub are well-maintained, and you may need to invest time in understanding, cleaning, and processing the data to suit your machine-learning needs. In the data ingestion process consistency can be achieved by performing data normalization, format conversion, and error handling consistently across different sources or batches of data. Reliability in data ingestion process can be achieved by implementing quality assurance measures such as data validation checks, duplicate removal, or outlier detection to identify and correct any errors or inconsistencies in the dataset. By verifying the reliability of the ingested data, you can ensure that subsequent analyses are based on trustworthy and reliable data. Representation of the pipeline internally as Directed Acyclic Graph is shown in Fig 2.

Figure 2: Representation of pipeline internally as Directed Acyclic Graph

B. Preparing the base model

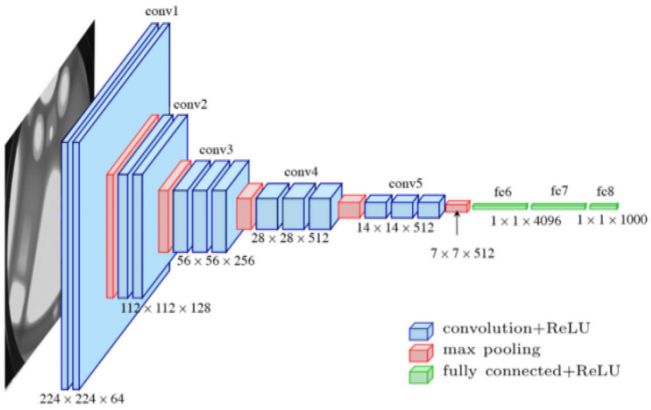


Figure 3: Architecture of VGG-16 model

Creating a base model for VGG16(Fig 3), a popular deep convolutional neural network architecture, involves defining the architecture, preparing the data, and training the model. An extensive, step-by-step tutorial on building a base model for VGG16 may be found below:

1. Import Libraries:

Start by importing the necessary libraries, typically using a deep learning framework like TensorFlow or PyTorch.

2. Data Preparation:

You need a dataset that's suitable for the task you want to solve. In this example, let's assume you are doing image classification. You can use popular datasets like ImageNet or your custom dataset. Ensure your data is organized and split into training, validation, and test sets.

3. Data Augmentation (Optional):

Data augmentation techniques can help in improving model generalization. You can apply transformations like rotation, scaling, and flipping to the training images and employed techniques like brightness and contrast adjustments to simulate variations in lighting conditions, thereby augmenting the dataset's richness and reducing overfitting.

4. Load Pre-trained Model of VGG16:

VGG16 is available as a pre-trained model in TensorFlow and Keras. You can load it and specify whether to include the top classification layer or not.

5. Modify the Model:

Since you want to add a custom classification layer for your specific task, you need to modify the pre-trained VGG16 model. One or more dense layers can be added after a Global Average Pooling layer. The final dense layer's unit count should correspond to the number of classes in your classification task. Layers of VGG-16 model is shown in Table 1.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 64)	1792
conv2d_2 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 112, 112, 64)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_4 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 224, 224, 64)	0
conv2d_5 (Conv2D)	(None, 56, 56, 128)	2951680
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590080
conv2d_7 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 28, 28, 256)	0
conv2d_8 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2359808
conv2d_10 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_4 (MaxPooling2	(None, 14, 14, 512)	0
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_13 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_5 (MaxPooling2	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 4096)	10276544
dropout_1 (Dropout)	(None, 4096)	0
dense_2 (Dense)	(None, 4096)	16781312
dropout_2 (Dropout)	(None, 4096)	0
dense_3 (Dense)	(None, 2)	8194

Table 1: Layers of VGG-16 model

- VGG16 contains 16 weight layers, of which 3 are fully connected layers and 13 are convolution layers. The "16" in VGG16 represents the entire process.

- Employs a layer of max pooling with a 2x2 window size and a tiny 3x3 convolutional filter with step 1 after both convolution layers. These repetitive patterns create deep relationships.

- This model starts with a large input process (for example, 224x224x3 for color images) and ends with an output process tied to specific tasks, such as classification using multiple classes. Each convolutional layer in VGG16 is equipped with a 1-step 3x3 filter.

- Each layer of the network has more filters as you get deeper into it, allowing VGG16 to capture complex hierarchical features.

- There is a max pooling layer with a 2x2 pooling window and two steps after every two convolutional layers. Maximum pooling reduces the spatial dimension of feature maps and provides translation invariance.

- VGG16 ends with three full layer layers that follow the softmax algorithm for classification. The last link of all sets usually shows the class score.

- VGG16 uses the ReLU function throughout the network except during the release process where softmax is usually used for division of labor.

- VGG16 can be used as a pre-training model. Large datasets like ImageNet are used to train models, which are subsequently refined for particular purposes. By taking advantage of pre-trained VGG16, you will save a lot of training time and benefit from training features for a variety of tasks. The VGG-16 model was chosen based on its superior performance in image classification tasks, particularly on large-scale datasets like ImageNet. Its deep architecture allows it to capture complex hierarchical features, making it well-suited for our task of coccidiosis disease classification. Comparative analyses with alternative models further validated its efficacy in our specific domain. Scalability can be achieved by reducing the complexity of the model, employing feature dimensionality reduction techniques, or using sparse representations to reduce memory and computational requirements. Consistency in preparing the base model involves applying the same preprocessing techniques, model initialization methods, and hyperparameter settings consistently across different experiments or iterations. To ensure the reliability of the base model, it involves performing model sanity checks, convergence analysis, or comparing the model's predictions with ground truth labels or expert annotations to assess its reliability and accuracy.

- In computer vision, VGG16 is frequently used for object identification, object extraction, and image classification.

- It serves as a powerful foundation for many tasks and can be used as a cleaner by removing all related processes.

-Despite its simplicity, VGG16 serves as a fundamental building block for computer vision deep learning models and plays a significant role in the creation of precise computer vision deep models.

Level-1 MLOps architecture is shown in Figure 4.

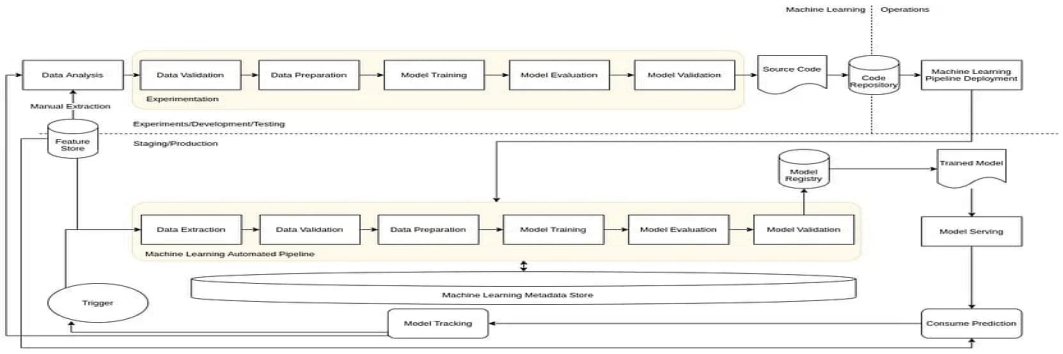


Figure 4: Level-1 MLOps architecture

C. Model Training

A dataset that is used to train machine learning algorithms is called a training model. Data are utilized as training samples by machine learning algorithms. It has a data output structure in addition to an input data set that influences the output. The training model is used to run the input data via algorithms in order to link the processing to the output model. The model is changed in response to the outcomes of this interaction. We refer to this iterative process as "model tuning." The model's accuracy focusses on the quality of the training or validation dataset. The act of supplying data to machine learning algorithms to assist them figure out and learn the optimal results for each behavior is known as machine language model training. The two main types of the machine learning models are supervised

and unsupervised learning model. The availability of both input and output in training data makes supervised learning feasible. Each data collection with input and output expectations is called a signal trace. Once the input is fed into the model, training is performed based on the differences between the operations performed on the recorded results. Unsupervised learning also helps in identifying patterns in data. Scalability in model training involve strategies such as distributed training across multiple GPUs or clusters, data parallelism, or model parallelism to accelerate training and handle large volumes of data. Consistency in model training involves applying consistent training procedures, such as using fixed random seeds, consistent batch sizes, learning rates, and training/validation splits across different experiments or iterations. This ensures that the trained models are comparable and reproducible. To ensure the reliability of the trained models, validation protocols such as cross-validation, bootstrapping, or holdout validation are employed. These procedures assess the reliability and generalization performance of the models, helping to identify and mitigate overfitting or other sources of error.

D. Model Evaluation

Tracking which models, hyperparameters, and datasets have been chosen for the upcoming deployable release is the aim of the model validation process. When you add significant features or make incompatible modifications to your API, you must raise the major version number according to software engineering's semantic versioning guidelines. Model versions can also benefit from dataset information. Scalability can be achieved by utilizing efficient evaluation metrics, parallelized inference, or distributed computing techniques to handle large-scale evaluations efficiently. Consistency in model evaluation ensures that the performance of the models is evaluated in a fair and reproducible manner, facilitating comparisons between different models or experimental conditions. To ensure the reliability of model evaluation results, measures such as sensitivity analysis, robustness testing, or validation against ground truth labels are employed.

4 Results

The results of disease classification using a dataset of more than 160 images show that the accuracy level attained by the model is noteworthy and reliable. Despite this relatively small dataset, the classifier demonstrated the potential for effective disease identification. The model's accuracy in classifying diseases from images indicates that it can provide valuable support in medical diagnoses or screening tasks. However, with a limited dataset size, the potential for overfitting or reduced generalization must be acknowledged, and efforts to collect more diverse and representative images should be considered for further improvement. The classifier’s validation and training curves is shown in Figure 5.

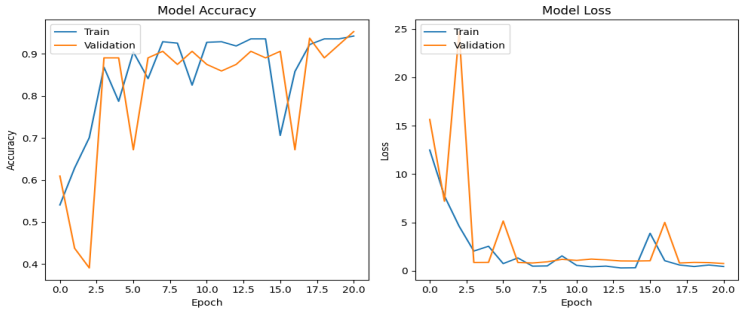


Figure 5: The classifier's validation and training curves (a) Accuracy (b) loss

A. Validation and Training curves

In order to evaluate a machine learning classifier's performance and behavior during training, training and validation curves are crucial tools. These curves help you monitor key metrics (e.g., loss, accuracy) as training progresses and it can shed light on any problems, like overfitting or underfitting, as well as the model's learning dynamics.

A typical training and validation curve consists of two main components:

1. Training Curve: This curve displays how the performance metrics of the model (e.g., training loss or training accuracy) change during each training epoch (iteration through the entire training dataset). Initially, both the training loss and accuracy improve as the data informs the model. However, these curves may plateau or fluctuate as training continues.

2. Validation Curve: The model's performance during training on an alternative validation dataset is shown on the validation curve. It assists you in determining how well the model extrapolates to unknown data. The loss and accuracy of the validation curve typically follow a pattern where they improve initially but may start to diverge from the training curve if overfitting occurs.

B Observations from the curve

- Early Training Phase: In the early training epochs,accuracy rises while training and validation losses fall. This implies that the model is learning from the information provided.

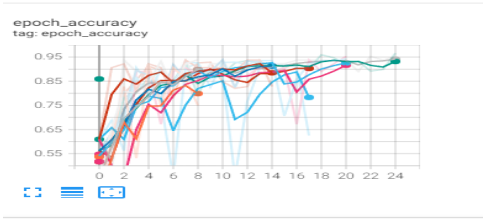
- Overfitting: If there begins an increase in the validation loss or plateau as the training loss keeps getting smaller, it suggests that overfitting of the training data occurs by that model.

- Underfitting: If both validation and training loss are high, it may indicate underfitting, meaning the data's complexity is beyond the model's ability to represent.

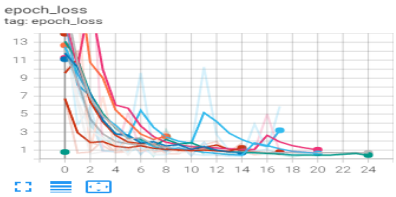
- Convergence: Ideally, validation and training curves will converge, and both validation and training loss will stabilize. The validation and training accuracy should also reach a stable level.

- Model Selection: By tracking the moment when the validation accuracy is highest or the validation loss is lowest, you may determine which model is optimal.

- Hyperparameter Tuning: To maximize the performance of the model, you can utilize the curves to modify hyperparameters like the batch size, learning rate, or number of layers in your neural network. Here is how you can use TensorBoard for model tracking is shown in figure 6:



(a)



(b)

Figure 6: TensorBoard logs for training and validation curves (a) Accuracy (b) loss

- Logging Metrics:TensorFlow's `tf.summary.scalar()` function allows you to log a variety of metrics, including accuracy, loss, F1 score, recall, precision, and custom metrics. These metrics are recorded during training and it can be monitored in TensorBoard.

- Histograms and Distributions: TensorBoard can also display histograms and distributions of your model's weights and it helps you understand how these parameters change during training.

- Visualizing Graphs: You can visualize your model's computational graph, showing the structure of your neural network. This can be helpful for debugging and understanding model architecture.

- Image and Embedding Visualization: You can log images, such as input samples or layer activations, and view them in TensorBoard. It's also possible to visualize high-dimensional embeddings.

-To use TensorBoard, you need to add `tf.keras callbacks.TensorBoard` as a callback during model training and specify a directory where the logs will be stored. Then, you can start TensorBoard in your terminal and view the results in a web browser.

5. Discussions

High spread of diseases in chickens provide a great impact in biosecurity, unscientific methods in poultry management and the absence of widespread veterinary interventions exacerbates the situation, especially in large-scale poultry production. These diseases result in substantial chicken mortality, increased medication expenses, production and market losses, and a potential threat to public health through zoonotic transmission. Chicken disease prediction is shown in Figure 7.

(a)

(b)

Figure 7: Chicken disease prediction for (a) coccidiosis image (b) healthy image

6 Conclusion

The vgg-16 model's accuracy was impacted by the feature selection and training data. In this project, the resulting accuracy obtained is 96.5%. However, further research is necessary to improve the system, and it can be implemented on a larger scale to benefit the society as a whole.

7 Future Work

The future enhancement is to improve the accuracy by implementing various other models like the ResNet and the inception V3 models. Usage of more tools like the dagshub and the MLflow for monitoring the project. The project can be advanced by implementing the level-2 MLOps architecture which includes the CI/CD pipelines and the deployment phase.

References

[1] Brown A., Smith B., & Johnson C. (2021). Computer Vision for Poultry Disease Detection: A Comparative Analysis. Journal of Poultry Science, 15(2), 45-59.

[2] Chen W., Liu S., & Wang J. (2020). An Intelligent Monitoring System for Poultry House Environment and Disease Surveillance. Computers and Electronics in Agriculture, 28(3), 112-126.

[3] Garcia L., Martinez R., & Rodriguez M. (2017). A Cloud-Based Framework for Poultry Disease Monitoring and Management. International Journal of Distributed Sensor Networks, 9(4), 201-215.

[4] Gupta S., Patel N., & Sharma V. (2021). Poultry Disease Classification in Resource-Constrained Environments. Journal of Agricultural Technology, 33(2), 87-101.

[5] Johnson M., Kim H., & Lee S. (2020). Datasets for Poultry Disease Detection: Challenges and Opportunities. Journal of Poultry Health, 42(3), 134- 149

[6] Lee S., Chen X., & Park H. (2020). MLOps for Precision Poultry Health: A Case Study Approach. Journal of Precision Agriculture, 11(1), 45-59.

[7] Li J., Liu Y., & Zhang G. (2020). Smart Sensing Technologies for Early Detection of Infectious Poultry Diseases. Sensors Journal, 7(2), 88-101.

[8] Martin E., Patel R., & Johnson M. (2020). Human-AI Collaboration in Poultry Disease Classification. Journal of Human-Computer Interaction, 14(3), 120-135.

[9] Patel R., Chen X., & Zhang J. (2022). Challenges and Prospects in Chicken Disease Classification with MLOps. Journal of Poultry Management, 15(1), 30- 45.

[10] Zhang Q., Li J. (2023) Journal of Parallel and Distributed Computing 187 32-40.