Athens Journal of Technology and Engineering - Volume 13, Issue 1,
March 2026 – Pages 47-62

Optimizing Inception Architectures for Automated Quality Control in Binary Classification

By Ambra Korra[*], Indrit Enesi[±] & Anduel Kuqi[°]

In computer vision, within convolutional neural network architecture, Inception algorithms play a crucial role for image classification tasks. This study focuses on optimizing Inception architectures for binary classification, particularly for automating quality control processes to simplify control phases and improve accuracy. Conventional quality control methods often rely on manual inspection, which can be time-consuming and prone to human error. The optimization process involves careful consideration of transfer learning methods, the benefits of incorporating fine-tuning and other regularization techniques such as data augmentation, choosing the right values for dropout layers and learning rates will show the best results in terms of accuracy and efficiency. The examination of three different Inception algorithms, Inception v3, Inception Resnet v2, and Xception, reveals that Xception achieves higher validation accuracy 99.72% compared with Inception Resnet V2 and Inception V3 in larger datasets. In the other hand for smaller dataset, Inception V3 achieves higher validation accuracy 99.69% compared with Inception Resnet V2 and Xception. The decision-making regarding the use of these algorithms should be guided by the specific use cases, as each algorithm presents distinct strengths suitable for quality control applications.

Keywords: *CNN, inception algorithms, transfer learning, quality control, image classification*

Introduction

The processing and manufacturing industry has been focusing on enhancing product quality and customer safety through defect detection methods. Manual inspection, commonly used in traditional quality control processes, has limitations such as being time-consuming, prone to human error, and not very efficient. Consequently, there is a growing interest in leveraging advanced machine learning methods, particularly in the realm of computer vision, for automatic quality control (Jing et al. 2020).

To automate the quality control process, the use of Convolutional Neural Networks (CNNs) is deemed to be the most fitting approach due to their superior performance in various image classification tasks. Among the wide array of CNN architectures, Inception-based models have garnered attention for their effectiveness and the ability to capture multi-scale features within images. The original Inception architecture, as

[*]PhD Student, Electronic and Telecommunication Department, Polytechnic University of Tirana, Albania.
[±]Professor, Electronic and Telecommunication Department, Polytechnic University of Tirana, Albania.
[°]Assistant Lecturer, Electronic and Telecommunication Department, Polytechnic University of Tirana, Albania.

https://doi.org/10.30958/ajte.13-1-3 doi=10.30958/ajte.13-1-3

well as its subsequent variants such as Inception v3, Inception-ResNet, and Xception, have consistently elevated the standards of accuracy and efficiency (Szegedy et al. 2015).

This research is specifically geared toward optimizing Inception architectures for the automated quality control of binary classification problems, aimed at distinguishing defective from non-defective products. Leveraging the power of transfer learning, which facilitates the adaptation of pre-trained models to new tasks, our objective is to fine-tune Inception-based models to achieve high accuracy in defect detection while upholding computational efficiency (Praveen Gujjar et al. 2021). Dataset obtained from Kaggle are used for training and evaluating our models. The dataset was split into training and testing sets approximately (90%/10%) and underwent thorough preprocessing to ensure the robustness of the model.

Starting with a literature review, highlights the recent advancement and applications of Inception algorithms, then the methodology section details the use of Inception V3, Inception Resnet V2 and Xception models and transfer learning techniques. In the results section, it is shown the use of various optimization techniques to identify the best performing model, following with some discussions and conclusions about the insights of selecting optimal algorithm for quality control processes.

Literature Review

The use of Inception architectures in different fields has led to significant progress in image classification tasks. This review discusses recent studies that employ Inception models and related architectures to solve various problems.

A range of machine vision techniques have been researched and applied to automated quality control. For instance, the study by Zhou et al. (2021) achieved an accuracy of 98.50%, while Habibzadeh Motlagh et al. (2018) used pre-trained deep learning models, including ResNet and Inception, for automatic white blood cell classification. This study demonstrates that Inception models deliver high classification accuracy by leveraging their deep feature extraction capabilities. The successful use of Inception models in medical diagnostics suggests their potential for similar critical applications, such as automated quality control in manufacturing. Szegedy et al. (2017) introduced Inception-v4 and Inception-ResNet, highlighting the impact of residual connections on learning. The incorporation of residual connections in Inception architectures represents a significant enhancement, providing a robust framework for tasks requiring high precision. Additionally, Patel and Shah (2023) conducted a comparative study on the early detection of rice diseases using various CNN architectures, including Inception v3, VGG16, VGG19, and ResNet50. The study reinforces the importance of these algorithms in image classification tasks. This comparative analysis underscores the strengths of Inception v3 in handling complex image classification tasks. The paper of Chollet (2017) introduces a novel deep learning architecture that builds upon the Inception model by utilizing depthwise separable convolutions. This approach reduces the number of parameters and computational cost while maintaining model performance. Also, the study by Poojary & Pai (2019) shows a Training accuracy of approximately 99% for ResNet50.

The reviewed literature emphasizes the effectiveness of Inception architectures in various domains, particularly in image classification and feature extraction tasks. The adaptability and robustness of these models make them ideal candidates for automated quality control in manufacturing, where precise defect detection is essential. By harnessing the advanced feature extraction capabilities of Inception models, this research aims to optimize their application in binary classification tasks, contributing to more efficient and reliable quality control processes in industrial settings.

Methodology

Transfer Learning

The Inception algorithms consist of various convolutional neural network (CNN) architectures that have played a significant role in computer vision and deep learning tasks. These architectures are characterized by "Inception modules," which contain convolutional layers with different filter sizes to capture features at multiple scales. In this study, we focus on utilizing Inception V3, Inception-ResNet-V2, and Xception. We conduct our research within the Google Colab environment, which provides GPU capabilities, and utilize the Kaggle Dataset for training and validating our models.

Google Colab offers cloud-based computing power and free GPU resources, making it a preferred choice for deep learning researchers. To get started, users need to create a Google account, upload their dataset into subfolders as specified in the Kaggle dataset, and mount their Google Drive in the Colab environment to enable seamless access to files stored in Google Drive directly from the Colab notebook. Taking advantage of the GPU acceleration feature offered by Google Colab enhances computational capabilities. In this case, we will be using T4 GPU, with GPU RAM of 15.0 GB, System RAM of 51 GB, and Disk space of 201.2 GB. The next step involves installing the necessary libraries and loading pre-trained models. TensorFlow, an open-source machine learning framework, provides a comprehensive ecosystem of tools, libraries, and community resources for building and deploying machine learning models. Keras, an open-source high-level neural network API written in Python, simplifies the process of building, training, and deploying deep learning models by combining the high-level abstractions of Keras with the underlying computational power of TensorFlow.

We load the Pretrained InceptionV3, Inception-ResNet-V2, and Xception models. With a dataset of 7348 images, and a second dataset of 2078 images, transfer learning can be an efficient solution to enable Inception network training without overfitting and convergence problems. This work initializes the Inception and Inception ResNet and Xception convolutional and fully connected layer weights from pre-trained ImageNet models. The InceptionV3 model in tensorflow.keras.applications contains Convolutional blocks, max pooling, activation, batch normalization, and fully connected layers. The architecture of the InceptionV3 model provides a summary of layer names, types, and their connections within the neural network. The InceptionResNetV2 model, available in TensorFlow Keras applications, has a more complex architecture

compared to InceptionV3 and includes a combination of Inception modules and residual connections. The Xception model is an extreme form of an Inception module, almost identical to a depthwise separable convolution, performing spatial convolution independently over each channel of an input, followed by a pointwise convolution.

Based on computational costs for Inception Algorithms, Inception Resnet v2 requires higher computational resources compared with Inception v3 (Korra et al. 2023). In this study we will also compare Xception algorithms, and this can be very helpful in deciding which architecture is better suited for specific image classification tasks.

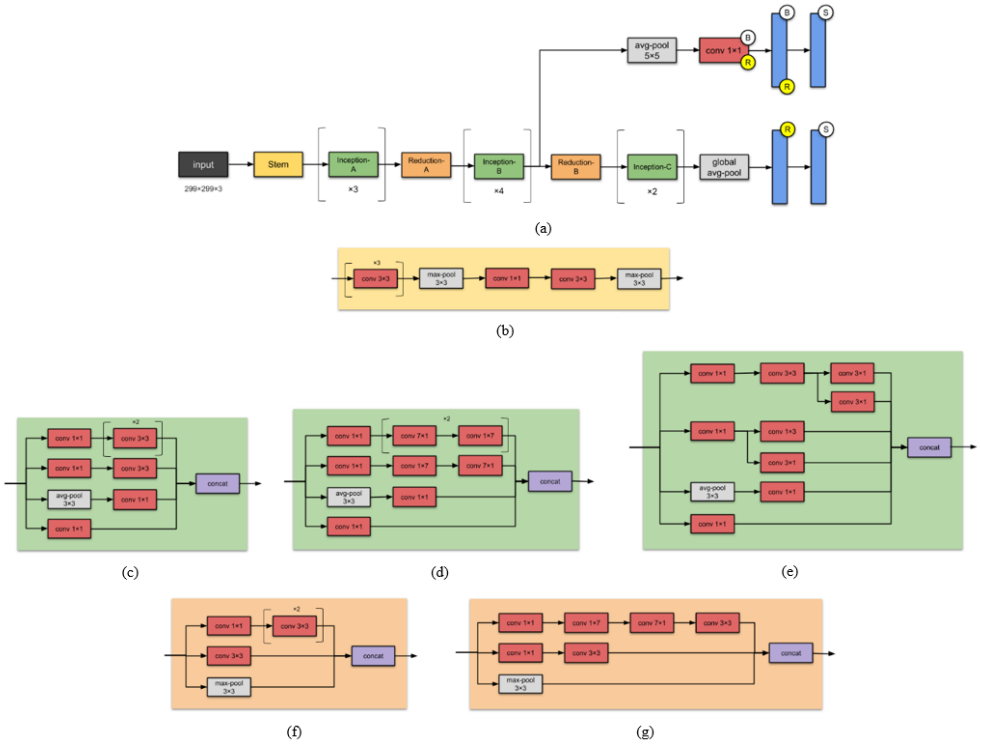
Inception Algorithms Architecture

Convolutional Neural Networks (CNNs) are crucial for image recognition tasks because they can learn spatial hierarchies of features from input images (Fan et al. 2023). They are especially useful for automated quality control systems in manufacturing, where they can identify defects in products. CNNs consist of layers of convolutions, pooling, and fully connected layers, which enable the network to learn complex features. Inception architectures, introduced by Szegedy et al. (2015), represent a significant advancement in CNN design. These architectures incorporate multiple convolutional filters of different sizes in parallel, which allows the network to capture various spatial hierarchies of features simultaneously. This is particularly beneficial for defect detection in manufacturing.

Inception V3 architecture, trained on the ImageNet dataset, uses mathematical operations pooling and convolutions by applying a filter to the input of any layer. Inception V3 is a build-up of the Inception module with multiple filter sizes and then stacks the outputs into one. This allows the detection of objects of different sizes effectively. This architecture also involves reduction modules, which are essentially the same as the inception module but are designed to decrease the dimensions of the input. As described in Figure 1, the input format is 299x299x3, and it goes into the Stem, which has some convolution and max-pooling layers. It is then applied 2x Inception A block, 1x Reduction A, 4x Inception B, 1x Reduction B, and 2x Inception C, global average pooling, Dense layers, and an auxiliary classifier (Korra et al. 2023).

The Inception module is strategically employed to reduce network dimensions. Using 1 × 1 and 3 × 3 convolutional layers, this module effectively minimizes the number of channels and parameters (Fan et al. 2023). Additionally, asymmetrical factorized convolutional neural networks, such as 1x3, 3x1, 1x7, and 7x1 convolution layers, contribute to this dimension reduction. To address convergence challenges in large deep learning layers, an auxiliary classifier is integrated, complemented by average pooling, convolutional 1×1 layers, fully connected layers, and softmax activation, providing a robust solution for the vanishing gradient problem in the final layers (Szegedy et al. 2016).

Figure 1. *Inception V3 Architecture(a), Schematic View of the Stem(b) and Inception A, B, C (c, d, e) and Reduction A, B blocks (f, g)*

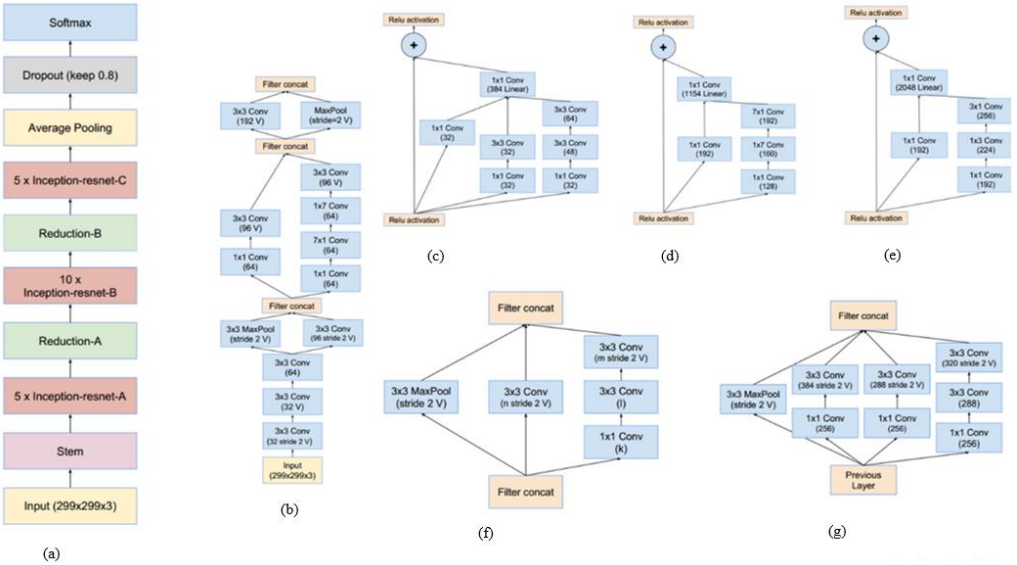


Source: ICITEE 2023.

In 2017, Szegedy et al. introduced the Inception-ResNet, a hybrid model that combines the Inception architecture with residual connections. This unique combination allows the network to benefit from the multi-scale feature extraction of Inception modules while also addressing the vanishing gradient problem through residual connections. The result is a significant improvement in training efficiency and model accuracy, especially in deep networks. In Figure 2, the complexity of the Inception ResNet v2 architecture is shown to exceed that of Inception v3. The layers of Inception ResNet, such as Inception ResNet A, B, and C, are depicted in Figures 2c–2e, with 5, 10, and 5 layers respectively. Additionally, Figures 2f and 2g illustrate the Reduction Layer, Reduction A, and Reduction B. Each Inception block is associated with a filter layer for dimension transformation and input mapping.

In a parallel context, the ResNet architecture addresses the degradation problem by incorporating shortcut connections that create direct paths from shallower sections to deeper networks. This strategy ensures smoother information flow throughout the network. Furthermore, the incorporation of adaptive learning rates, through the Adam optimizer with learning rate 0.001, plays a crucial role in enhancing the efficiency of deep learning concepts. Given the limited size of the available dataset relative to the numerous model parameters, regularization techniques such as dropout and batch normalization are considered essential. These regularization methods contribute to model generalization and stability, ensuring effective training for various scenarios (Poojary & Pai 2019).

Figure 2. *Inception Resnet V2 Architecture (a), Schematic View of the Stem(b), Inception Resnet A, B, C (c, d, e) and Reduction A, B Blocks (f, g)*



Source: AAAI conference on artificial intelligence 2017.

The Xception model, introduced by Chollet (2017), pushes the boundaries of the Inception architecture by replacing standard Inception modules with depthwise separable convolutions. This enhancement results in a more efficient network with fewer parameters and superior performance. Xception has demonstrated outstanding results in image classification tasks, especially in contexts that demand detailed feature extraction. The Xception architecture is a linear stack of depthwise separable convolution layers with residual connections. The network's feature extraction base consists of 36 convolutional layers organized into 14 modules, each with linear residual connections except for the first and last modules. The data first passes through the entry flow, then through the middle flow, which is repeated eight times, and finally through the exit flow.

Optimization Techniques

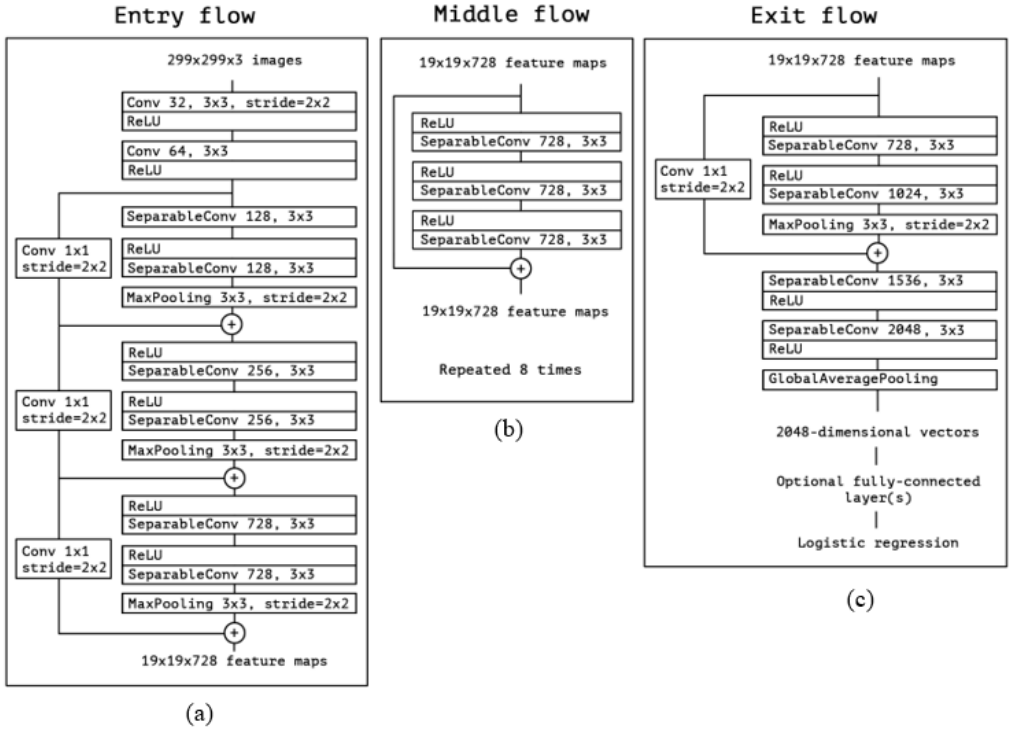
We have leveraged pre-trained models such as Inception V3, Inception ResNet V2, and Xception as the foundation for training our quality control detection model. This method involves utilizing the knowledge obtained from extensive datasets in general image recognition tasks and applying it to quality control detection.

By incorporating transfer learning, we can achieve high precision in our models, even with limited training data and computational resources. Furthermore, we employ optimization techniques to enhance performance, training efficiency, and model robustness. Below are specific techniques that can be implemented for these architectures:

- Learning Rate: Fine-tuning the learning rate is crucial. We will explore the application of values such as 0.001 and 0.0015.

- Dropout: Randomly deactivating units in the dense layers during training to prevent overfitting. Typical dropout rates range from 0.2 to 0.5.

Figure 3. *Xception Architecture Entry Flow(a), Middle Flow (b), Exit Flow (c)*



Source: IEEE Conference on Computer Vision and Pattern Recognition 2017.

Freezing Layers: Freezing the convolutional base of the pre-trained model and only training the dense layers on top.

Data Augmentation: Introducing random rotations up to 20 degrees to make the model orientation-invariant, random zooming in/out by 20% to aid the model in recognizing objects at different scales, and horizontally flipping images to augment the dataset.

In this study it is utilized a pre-trained model (InceptionV3, InceptionResNetV2, or Xception) and frozen its layers to serve as a feature extractor. Following this, I added a dense layer with 256 units, along with a dropout layer and a final dense layer for binary classification. The model was then compiled using the Adam optimizer with a specified learning rate, and its performance was evaluated. By applying these optimization techniques, the model is now better equipped to handle the binary classification task in an automated quality control system achieving an accuracy more than 99.50%. This approach can be adapted and further fine-tuned to meet the specific requirements and constraints of the application. In the context of quality control in manufacturing, the primary objective often involves determining whether a product is defective or not, posing a binary classification problem. Advanced CNN architectures such as Inception V3, Inception-ResNet, and Xception are particularly well-suited for this task due to their capacity to learn complex and detailed feature

representations. This capability is essential for detecting subtle defects that may not be discernible through traditional inspection methods.

Results

A collection of approximately 7348 images (Table 1) and 2076 images (Table 2) was initially obtained from an online machine-learning dataset repository called Kaggle, Casting product image data (Dabhi 2020) and Lemon Quality Dataset (Emir 2022). These images include 'casting product image data for quality inspection, divided into two subfolders train and test with, "ok_front" and "def_front", and Lemon quality respectively "bad_quality" and "good quality". A preprocessing step was executed by the image data generator for training and validation data using the TensorFlow/Keras library. This includes rescale=1./255 normalizes the image pixel values to the range [0, 1] random rotations up to 20 degrees, random zooming in/out by 20%, and horizontally flipping images. Create a generator for training data using the "flow_from_directory" method of the "train_datagen" instance. Resizes the images to the specified target size (299x299), this is often required to match the input size expected by certain deep learning models, such as Inception V3 and Inception-ResNetV2 and Xception in this case. Sets the batch size for training. The model will be updated based on batches of 32 images. Specifies that the task is binary classification. After that, it is needed to create a generator for validation data using a similar approach to the training generator.

Table 1. *Details of Image Dataset as per Classes*

Images	def_front	ok_front	Total test or train
Test	453	262	715
Train	3758	2875	6633
Total	4211	3137	7348

Table 2. *Details of Second Image Dataset as per Classes*

Images	bad_quality	good_quality	Total test or train
Test	95	112	207
Train	856	1013	1869
Total	951	1125	2076

The next step involves setting up a neural network model for binary classification using the InceptionV3 and Inception Resnet v2 and Xception architecture with transfer learning. First, an instance of the Inception model is created with pre-trained weights from ImageNet. The "include_top" parameter is set to "False" to exclude the fully connected layers at the top, and the input shape is specified as (299, 299, 3). All the layers in the pre-trained InceptionV3 model are frozen, which is common practice in transfer learning to keep the pre-trained weights fixed during the initial training.

A Sequential model is initialized, and layers are added to it:

The Flatten layer converts the 2D features from the Xception model into a 1D vector.
A dense layer with 256 units and ReLU activation is added for further learning.
A Dropout layer with a rate of 0.5 is added to prevent overfitting.
Finally, a dense layer with 1 unit and sigmoid activation is added for binary classification.

The model is then compiled with the Adam optimizer (learning rate set to 0.001), binary cross-entropy loss function (suitable for binary classification), and accuracy as the evaluation metric. This code creates a binary classification model based on the InceptionV3, Inception Resnet V2 and Xception architecture using transfer learning, with the pre-trained layers frozen and new layers added on top for the specific binary classification task. The model is compiled for training, with the training process iterating over the entire training dataset 10 times (epochs). We will gather information about the training process, such as the training and validation loss, and accuracy. Detailed information will be shown in the respective tables.

In this study, firstly it is important to choose the right optimization techniques, testing the parameters of dropouts and learning rate. After specifying the optimal parameters, a comparison between Inception V3, Inception Resnet V2 and Xception is defined.

For the first Dataset (7348 images) at Tables 3-5 it is shown that when we increase the dropouts and learning rate the overall accuracy is decreased. The best combination by comparing loss, accuracy, val_loss and val_accuracy in this case is using dropouts 0.2 and learning rate 0.001.

Table 3. *Performance of Xception Model Along Evaluation Metrics for the Epoch with Highest Accuracy*

Xception Model	Epoch	Loss	Accuracy	Val_loss	Val_accurancy
Dropout 0.2; Learning rate 0.001	7/10	0.038	0.9863	0.017	0.9972
Dropout 0.5; Learning rate 0.001	9/10	0.0945	0.9603	0.0213	0.9958
Dropout 0.2; Learning rate 0.0015	9/10	0.0524	0.9812	0.0202	0.9958

Table 4. *Performance of Inception Resnet V2 Model Along Evaluation Metrics for the Epoch with Highest Accuracy*

Inception Resnet V2 Model	Epoch	Loss	Accuracy	Val_loss	Val_accurancy
Dropout 0.2; Learning rate 0.001	4/10	0.0433	0.9873	0.019	0.9958
Dropout 0.5; Learning rate 0.001	9/10	0.1778	0.9303	0.0235	0.9958
Dropout 0.2; Learning rate 0.0015	9/10	0.061	0.9745	0.0197	0.9958

Table 5. *Performance of Inception V3 Model Along Evaluation Metrics for the Epoch with Highest Accuracy*

Inception V3 Model	Epoch	Loss	Accuracy	Val_loss	Val_accurancy
Dropout 0.2; Learning rate 0.001	6/10	0.057	0.9798	0.0197	0.9944
Dropout 0.5; Learning rate 0.001	10/10	0.1132	0.9533	0.0214	0.9944
Dropout 0.2; Learning rate 0.0015	9/10	0.051	0.9806	0.0225	0.9944

Secondly it is needed to compare these algorithms to check which of them is more efficient, while using the optimal dropout and learning rate values.

This information is shown in table 6 where we notice that Xception achieves higher accuracy, followed by Inception Resnet V2 and Inception V3.

Table 6. *Model Comparison Using Optimal Dropout Layers and Learning Rates*

Model	Epoch	Loss	Accuracy	Val_loss	Val_accurancy
Xception	7/10	0.038	0.9863	0.017	0.9972
Inception Resnet V2	4/10	0.0433	0.9873	0.019	0.9958
Inception V3	6/10	0.057	0.9798	0.0197	0.9944

More detailed information for each algorithm is shown at Table 7-9 and Figures 4-6.

Table 7. *Performance of Xception Model Along Evaluation Metrics for 10 Epochs*

Epoch	Loss	Accuracy	Val_loss	Val_accurancy
1/10	0.9124	0.9279	0.0333	0.9944
2/10	0.0897	0.9717	0.0315	0.9944
3/10	0.0647	0.9775	0.0336	0.9888
4/10	0.072	0.976	0.0568	0.9846
5/10	0.053	0.9837	0.0185	0.9958
6/10	0.0371	0.9872	0.0135	0.9972
7/10	**0.038**	**0.9863**	**0.017**	**0.9972**
8/10	0.037	0.9876	0.0305	0.9944
9/10	0.0477	0.9843	0.0284	0.9888
10/10	0.0353	0.9881	0.0172	0.9958

Figure 4. *(Left) Accuracy Curve and (Right) Loss Curve for Xception Model*

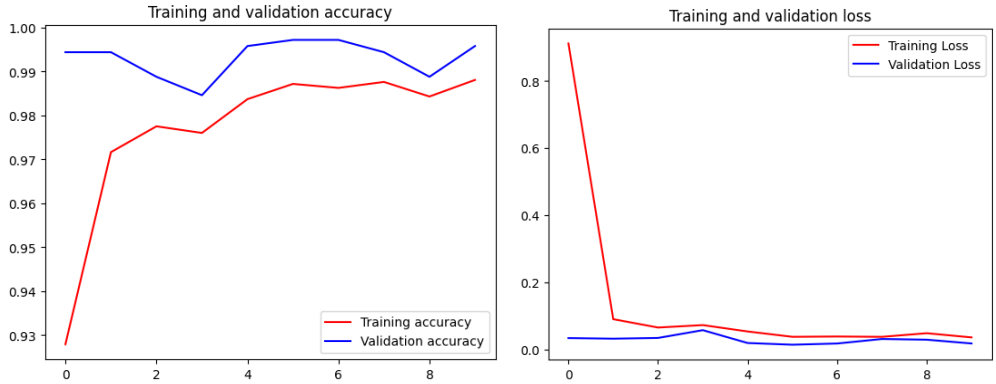


Table 8. *Performance of Inception Resnet V2 Model Along Evaluation Metrics for 10 Epochs*

Epoch	Loss	Accuracy	Val_loss	Val_accurancy
1/10	1.3467	0.9193	0.0366	0.9902
2/10	0.0659	0.9781	0.0379	0.986
3/10	0.0887	0.9691	0.0375	0.9888
4/10	**0.0433**	**0.9873**	**0.019**	**0.9958**
5/10	0.0555	0.9784	0.0255	0.993
6/10	0.0692	0.9771	0.0494	0.9818
7/10	0.0424	0.9861	0.024	0.993
8/10	0.0328	0.9896	0.0283	0.9888
9/10	0.0307	0.9919	0.0251	0.9902
10/10	0.036	0.989	0.0326	0.9902

Figure 5. *(Left) Accuracy Curve and (Right) Loss Curve for Inception Resnet V2 Model*

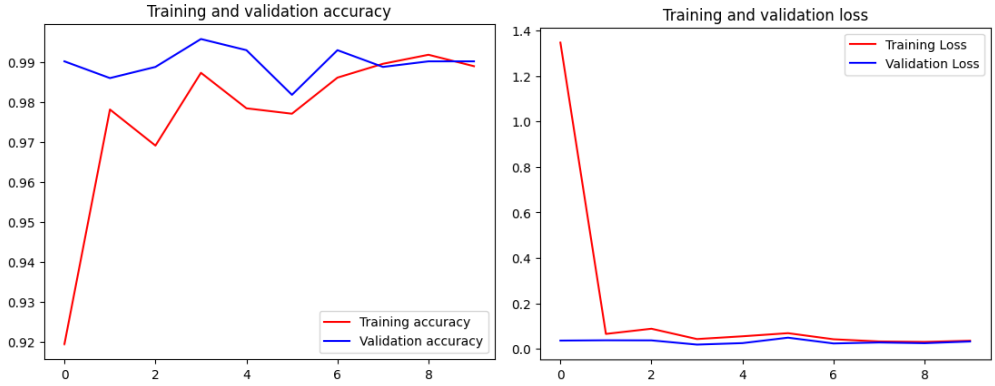
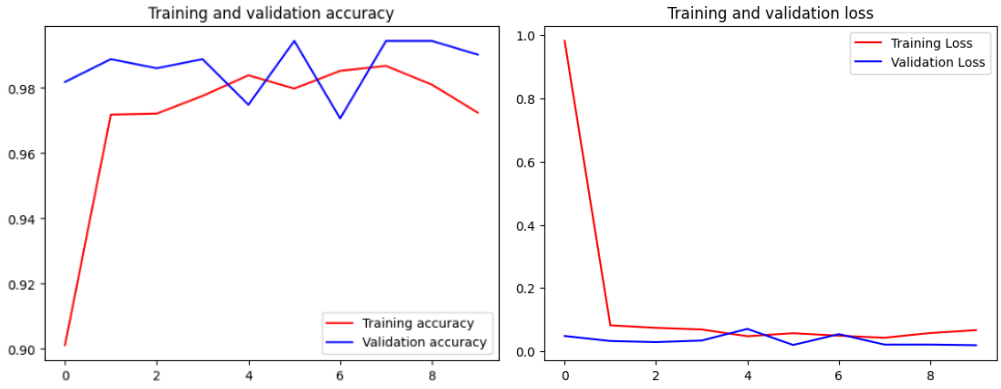


Table 9. *Performance of Inception V3 Model Along Evaluation Metrics for 10 Epochs*

Epoch	Loss	Accuracy	Val_loss	Val_accurancy
1/10	0.9824	0.9011	0.048	0.9818
2/10	0.0818	0.9718	0.0324	0.9888
3/10	0.0739	0.9721	0.0289	0.986
4/10	0.0688	0.9775	0.0339	0.9888
5/10	0.0473	0.9839	0.0707	0.9748
6/10	**0.057**	**0.9798**	**0.0197**	**0.9944**
7/10	0.049	0.9852	0.0541	0.9706
8/10	0.0425	0.9867	0.0209	0.9944
9/10	0.0576	0.981	0.0209	0.9944
10/10	0.0667	0.9724	0.0189	0.9902

Figure 6. *(Left) Accuracy Curve and (Right) Loss Curve for Inception V3 Model*



It was observed that the Inception V3 algorithm in Google Colab used T4 GPU with 13.7 GB RAM, 5.1 GB System RAM, and 29.0 GB disk space for 15m11s; Inception Resnet V2, 13.7 GB RAM, 5.1 GB System RAM, and 30.0 GB disk space for 15m37s; Xception used 13.7 GB RAM, 4.5 GB System RAM, and 28.1 GB disk space FOR 15m13s. This shows that Inception Resnet v2 requires higher computational resources, and it requires more time.

The same procedure is followed for the second dataset (2078 images). At Tables 10-12 it is shown that when we increase the dropouts and learning rate the overall

accuracy is decreased. The best combination in this case is using dropouts 0.2 and learning rate 0.001.

Table 10. *Performance of Xception Model Along Evaluation Metrics for the Epoch with Highest Accuracy*

Xception Model	Epoch	Loss	Accuracy	Val_loss	Val_accurancy
Dropout 0.2; Learning rate 0.001	4/10	0.0792	0.9845	0.1326	0.9855
Dropout 0.5; Learning rate 0.001	10/10	0.0431	0.9818	0.1103	0.9758
Dropout 0.2; Learning rate 0.0015	5/10	0.2124	0.9823	0.1162	0.9807

Table 11. *Performance of Inception Resnet V2 Model Along Evaluation Metrics for the Epoch with Highest Accuracy*

Inception Resnet V2 Model	Epoch	Loss	Accuracy	Val_loss	Val_accurancy
Dropout 0.2; Learning rate 0.001	9/10	0.0259	0.992	0.0218	0.9903
Dropout 0.5; Learning rate 0.001	8/10	0.0648	0.9754	0.0446	0.9855
Dropout 0.2; Learning rate 0.0015	10/10	0.0224	0.9914	0.0285	0.9903

Table 12. *Performance of Inception V3 Model Along Evaluation Metrics for the Epoch with Highest Accuracy*

Inception V3 Model	Epoch	Loss	Accuracy	Val_loss	Val_accurancy
Dropout 0.2; Learning rate 0.001	10/10	0.029	0.9893	0.0062	0.9969
Dropout 0.5; Learning rate 0.001	10/10	0.0643	0.9823	0.0184	0.9952
Dropout 0.2; Learning rate 0.0015	9/10	0.0366	0.9888	0.0032	0.9969

Also, it is needed to compare these algorithms to check which of them is more efficient, when using the optimal dropout and learning rate values.

This information is shown in table 13 where we notice that with a smaller dataset Inception V3 achieves higher accuracy, followed by Inception Resnet V2 and Xception model.

Table 13. *Model Comparison Using Optimal Dropout Layers and Learning Rates*

Model	Epoch	Loss	Accuracy	Val_loss	Val_accurancy
Xception	4/10	0.0792	0.9845	0.1326	0.9855
Inception Resnet V2	9/10	0.0259	0.992	0.0218	0.9903
Inception V3	10/10	0.029	0.9893	0.0062	0.9969

More detailed information for each algorithm is shown at Tables 14-16 and Figures 7-9.

Table 14. *Performance of Xception Model Along Evaluation Metrics for 10 Epochs*

Epoch	Loss	Accuracy	Val_loss	Val_accurancy
1/10	1.2551	0.923	0.606	0.9614
2/10	0.3646	0.9663	0.1715	0.9758
3/10	0.2531	0.9727	0.5258	0.9565
4/10	**0.0792**	**0.9845**	**0.1326**	**0.9855**
5/10	0.0922	0.9818	0.2557	0.971
6/10	0.1447	0.9802	0.259	0.9662
7/10	0.0676	0.9872	0.067	0.9758
8/10	0.0389	0.9866	0.0819	0.9758
9/10	0.0618	0.9829	0.1716	0.9614
10/10	0.038	0.9882	0.0926	0.9758

Figure 7. *(Left) Accuracy Curve and (Right) Loss Curve for Xception Model*

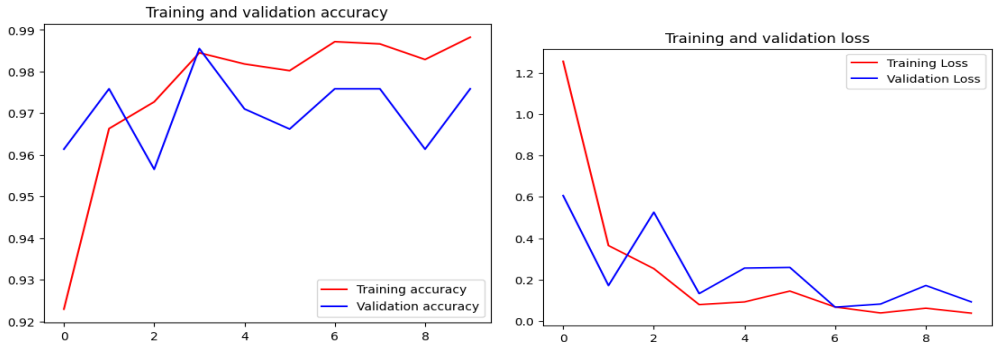


Table 15. *Performance of Inception Resnet V2 Model Along Evaluation Metrics for 10 Epochs*

Epoch	Loss	Accuracy	Val_loss	Val_accurancy
1/10	1.3621	0.9208	0.2102	0.9565
2/10	0.189	0.9663	0.0418	0.9855
3/10	0.1033	0.9679	0.0543	0.9903
4/10	0.0546	0.9813	0.0559	0.971
5/10	0.0366	0.985	0.0555	0.9807
6/10	0.0447	0.9861	0.0408	0.9807
7/10	0.0403	0.9856	0.0427	0.9855
8/10	0.0425	0.9861	0.0542	0.9807
9/10	**0.0259**	**0.992**	**0.0218**	**0.9903**
10/10	0.0314	0.9872	0.0496	0.9855

Figure 8. *(Left) Accuracy Curve and (Right) Loss Curve for Inception Resnet V2 Model*

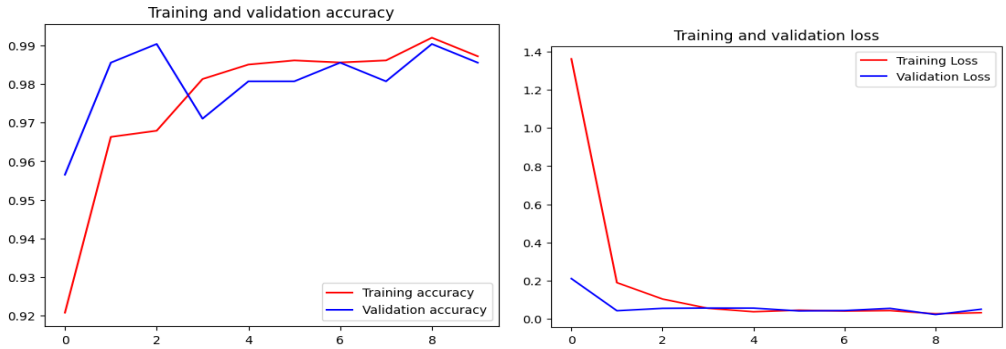
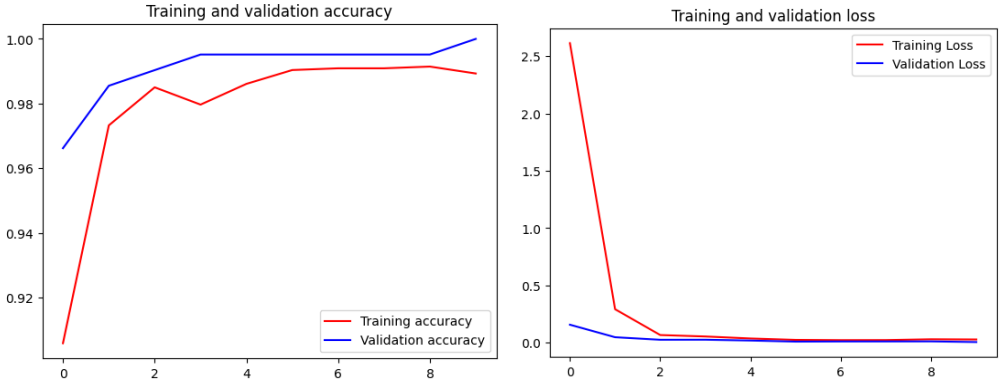


Table 16. *Performance of Inception V3 Model Along Evaluation Metrics for 10 Epochs*

Epoch	Loss	Accuracy	Val_loss	Val_accurancy
1/10	2.6139	0.9058	0.1576	0.9662
2/10	0.2928	0.9732	0.0484	0.9855
3/10	0.0675	0.985	0.0263	0.9903
4/10	0.0549	0.9797	0.0268	0.9952
5/10	0.0376	0.9861	0.0196	0.9952
6/10	0.0253	0.9904	0.01	0.9952
7/10	0.023	0.9909	0.0113	0.9952
8/10	0.0236	0.9909	0.0115	0.9952
9/10	0.0306	0.9914	0.0121	0.9952
10/10	**0.029**	**0.9893**	**0.0062**	**0.9969**

Figure 9. *(Left) Accuracy Curve and (Right) Loss Curve for Inception V3 Model*



It was observed that the Inception V3 algorithm in Google Colab used T4 GPU with 13.7 GB RAM, 6.2 GB System RAM, and 31.8 GB disk space for 4m29s; Inception Resnet V2, 13.7 GB RAM, 6.2 GB System RAM, and 32.1 GB disk space for 4m50s; Xception used 13.7 GB RAM, 6.5 GB System RAM, and 32.8 GB disk space for 4m27s. This shows that Inception V3 model requires lower computational resources.

Discussion

Throughout this research, we delved into the efficacy of optimizing Inception architectures for automated quality control in binary classification within manufacturing industries. Our main aim was to improve the performance of these models in discerning between defective and non-defective products by utilizing advanced data augmentation techniques and parameter optimization.

The findings revealed that through fine-tuning Inception architectures, significant enhancements in model performance were attainable with a value of accuracy, more than 99.50%. Specifically, employing transfer learning with pre-trained Inception models such as InceptionV3 and Xception facilitated robust initial feature extraction, which proved crucial for handling the complexity of the image data in our dataset.

Conclusions

Our research highlights the significant potential of Inception architectures in automating quality control processes in manufacturing industries. Through systematic optimization involving parameter tuning and data augmentation, we were able to greatly improve the performance of these models. Our study reaffirmed the suitability of Inception architectures, especially InceptionV3, Inception Resnet V2 and Xception, for binary classification tasks in quality control, owing to their advanced feature extraction capabilities.

Fine-tuning parameters is critical for achieving optimal model performance. Factors such as the learning rate (0.001), dropout rate (0.2), and dense layer configurations significantly influenced the models' accuracy and generalization. The

use of data augmentation techniques proved crucial in diversifying the training dataset and enhancing the robustness and generalization of the models. Our findings indicated that Xception outperformed Inception Resnet V2 and Inception V3 in terms of validation accuracy, with 99.72% compared to 99.58% and 99.44% of the other respective algorithms in dataset 1. However, it's important to note that in smaller datasets Inception V3 demonstrated a clear advantage in accuracy and computational efficiency. The selection between these models should be driven by the specific requirements of the use case. While Xception offers higher accuracy for large data sets, Inception V3 may be preferable in scenarios with lower dataset where computational resources are a crucial consideration, with an accuracy of 99.69% followed by Inception Resnet V2 99.03% and Xception 98.55%. It's important to evaluate the specific use case and priorities, as each algorithm possesses distinct strengths suitable for various applications.

References

Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 1251–1257.

Dabhi R (2020) *Casting product image data for quality inspection*. Kaggle.com.

Emir Y (2022) *Lemon Quality Dataset*. Kaggle.com.

Fan Y, Li J, Bhatti UA, Shao C, Gong C, Cheng J, et al. (2023) A multi-watermarking algorithm for medical images using inception V3 and DCT. *Computers, Materials & Continua* 74(1): 1279–1302.

Habibzadeh Motlagh M, et al. (2018) Automatic White Blood Cell classification using pre-trained deep learning models: ResNet and inception. In *Tenth International Conference on Machine Vision (ICMV 2017),* 1–3.

Jing J, Wang Z, Ratsch M, Zhang H (2020) Mobile-Unet: An efficient convolutional neural network for fabric defect detection. *Textile Research Journal* 92(1–2).

Korra A, et al. (2023) Efficiency and Performance Evaluation of Inception-V3 and Inception-ResNet-V2 in Image Classification. In *ICITEE Conference*, 2–4.

Lemon Quality Dataset (2022) Available at: https://www.kaggle.com/datasets/yusufemir/lemon-quality-dataset.

Patel M, Shah M (2023) Transfer learning with fine-tuned deep CNN model for COVID-19 diagnosis from chest X-ray images. *International Journal of Advanced Technology and Engineering Exploration* 10(103): 720–740.

Poojary R, Pai A (2019) Comparative study of model optimization techniques in fine-tuned CNN Models. In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*, 1–3.

Praveen Gujjar J, Prasanna Kumar HR, Chiplunkar NN (2021) Image classification and prediction using transfer learning in Colab Notebook. In *Global Transitions Proceedings*, 3–4.

Szegedy C, et al. (2015) Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–3.

Szegedy C, et al. (2016) Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 1–6.

Szegedy C, et al. (2017) Inception-V4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4278–4282.

Zhou, Z, Deng W, Zhu Z, Wang Y, Du J, Liu X (2021) Fabric defect detection based on feature fusion of a convolutional neural network and Optimized Extreme Learning Machine. *Textile Research Journal* 92(7–8): 1–3.