

**Software Engineering Department**

**Braude College of Engineering**

**Real time fruit grading using data mining and machine learning**

**Project code: 25-2-D-6**

**Capstone Project Phase A – 61998 Jul 2025**

**Lidor Anfinger**

**Dror Hershcovish**

**Git repository link:** <https://github.com/Drorh473/fruit_grading.git>

**Supervisor: Mr. Ilya Zeldner**

Table of Contents

[Abstract](#Abstract) 4  
[Key words](#Keywords)4  
[**1 Introduction**](#Introduction) **4  
2 Background and Related Work 5** 2.1 Fruit Grading in Agriculture 5  
 2.2 Computer Vision for Fruit Quality Assessment   
 2.2.1 Deep Learning for Fruit Quality Assessment  
 2.2.2 Cross-View Identification  
 2.3 Current Solutions for Fruit Quality Assessment  
 2.4 CNN  
 2.5 ShuffleNet  
 2.5.1 Key Features of ShuffleNet  
 2.5.2 How ShuffleNet Works  
 2.5.2.1 Input Processing  
 2.5.2.2 Convolutional Layers  
 2.5.2.3 Activation Functions  
 2.5.2.4 Hierarchical Feature Extraction  
 2.5.2.5 Flattening in Neural Network Architectures  
 2.5.2.6 Fully Connected Layers  
 2.5.2.7 Backpropagation  
 2.5.3 Applications  
 2.6 ShuffleNet V2  
 2.6.1 Core Building Block  
 2.6.2 Network Architecture and Staging  
 2.6.3 ShuffleNetV2 Architecture Diagram

**3 Research / Engineering Process**  
 3.1 Development Methods  
 3.1.1 Python for AI and Machine Learning as Server  
 3.1.2 React for the Frontend  
 3.1.3 MongoDB for Database  
 3.2 3D Rotation Estimation Fruits Dataset  
 3.3 Preprocessing Methods  
 3.3.1 Gaussian Filter  
 3.3.2 CLAHE Settings  
 3.3.3 Preprocessing Code  
 3.4 Choose the CNN Model  
 3.4.1 Performance Matrix  
 3.4.2 Chosen Model  
 3.5 Methods to Measure Precision  
 3.5.1 Sensitivity   
 3.5.2 Specificity   
 3.5.3 Accuracy

# Abstract

The labour-intensive nature of manual fruit grading in agricultural supply chains creates significant challenges for small-scale producers. This project introduces an innovative real-time visual inspection system that combines multi-view camera inputs with state-of-the-art deep learning models to automate fruit quality assessment. Our solution incorporates a custom-designed conveyor belt system featuring a specialized inspection box equipped with strategically positioned cameras and LED strip lighting for optimal image capture. Our approach integrates lightweight convolutional neural network (CNN) architectures to process temporal sequences of fruit images, detecting defects and classifying ripeness levels across various fruit types. The system efficiently segments multiple fruit instances from images using advanced computer vision techniques and applies sophisticated image enhancement to ensure consistent performance across varying lighting conditions.

Key words: Computer vision, Deep learning, CNN, Fruit classification, Image processing, Quality control.

# 1 Introduction

With increased consumer demand for high-quality fruits across global markets, there arises a critical need for advanced automated fruit grading systems. Traditional manual inspection approaches have demonstrated significant limitations in accuracy, consistency, and efficiency due to the labour-intensive nature of visual inspection and human-induced variability. The agricultural industry faces persistent challenges in fruit classification, grading, and quality assessment processes while attempting to meet the growing demand for consistently high-quality fresh produce in global markets.

As noted by Ismail and Malik, "traditional manual visual grading of fruits has been one of the important challenges faced by the agricultural industry due to its laborious nature as well as inconsistency in inspection and classification process" [[2]](#Reference2). Their research demonstrates that automated defects detection using computer vision and machine learning has become a promising area with direct impact on visual inspection domains.

The relevance of such technology is particularly pronounced given the agricultural labour shortages reported across major producing regions. The labour is the U.S. agricultural sector’s third largest production expense [[3]](#Reference3).

The economic impact of manual fruit grading is substantial. According to data from the Economic Research Service, labour accounts for approximately 42% of the variable production expenses for U.S. fruit and vegetable farms [[4]](#Reference4). This labour-intensive process is becoming increasingly costly as farm wages rise at a faster rate than non-farm wages, with agricultural wage growth outpacing non-agricultural wages (16% vs. 5% growth) between 2001 and 2019 [[5]](#Reference5).

Our proposed solution directly addresses these challenges by implementing an end-to-end automated inspection system that combines advanced computer vision techniques with efficient deep learning models through a comprehensive pipeline: image acquisition via a specialized mechanical conveyor system developed with a mechanical engineering student; feature extraction using ShuffleNetV2, a lightweight CNN with Rectified Linear Unit (ReLU)activation functions that balances computational efficiency with classification accuracy through innovative channel shuffle operations ; temporal processing with time-distributed feature flattening and pooling for multi-view assessment; integration of feature vectors from multiple cameras when applicable; and final classification through fully connected layers with Softmax activation function to generate probability distributions across quality classes. This integrated hardware-software approach ensures high-accuracy fruit grading capabilities while operating within the processing limitations of embedded systems commonly deployed in agricultural environments, providing consistent, objective quality assessments that exceed traditional manual inspection methods in both accuracy and efficiency.

By reducing dependency on specialized labour for quality control, our system allows agricultural operations to reallocate their limited workforce to tasks that truly require human expertise and judgment.

# 2 Background and Related Work

**2.1 Fruit Grading in Agriculture**

Fruit grading is a decisive operation within the agricultural supply chain, entailing the classification of fruits based on parameters such as appearance, size, color, and the presence of defects. Traditional manual grading methods are labor-intensive and prone to inconsistencies due to human subjectivity and fatigue. These inconsistencies can adversely affect product pricing, market acceptance, and consumer satisfaction. Consequently, there is a growing emphasis on the development and implementation of automated fruit grading systems.

Automated grading technologies, leveraging advancements in computer vision and deep learning, offer enhanced accuracy and consistency in quality assessment. Such systems can process large volumes of produce efficiently, reducing reliance on manual labor and minimizing human error. The integration of these technologies not only streamlines the grading process but also ensures allegiance to stringent quality standards demanded by global markets.

Moreover, the adoption of automated systems facilitates better traceability and transparency within the supply chain, thereby bolstering consumer confidence and satisfaction. As the agricultural industry continues to evolve, the role of automated fruit grading systems becomes increasingly integral to meeting the demands of quality assurance and operational efficiency.

**2.2 Computer Vision for Fruit Quality Assessment**

Computer vision technology has revolutionized fruit quality assessment by providing objective, consistent, and efficient evaluation methods. The application of computer vision in fruit grading typically involves a sequential process:

1. **Image Collection**: Capturing high-quality images of fruits under controlled lighting conditions to ensure consistency.
2. **Preprocessing**: Enhancing image quality through noise reduction and contract adjustments.
3. **Feature extraction**: Identifying characteristics such as color, texture, and shape that correlate with quality parameters.
4. **Classification**: Assigning quality grades based on extracted features using machine learning models.

**2.2.1 Deep Learning for Fruit Quality Assessment**

Deep learning has transformed fruit grading systems by providing automated, highly accurate, and efficient quality assessment, surpassing the limitations of conventional computer vision techniques. CNN have emerged as the dominant architecture for visual fruit quality inspection, demonstrating remarkable ability to detect subtle defects, color variations, and ripeness indicators that might be imperceptible to human inspectors. Unlike conventional image processing techniques that rely on hand-crafted features, these models learn hierarchical representations directly from training data, eliminating the need for manual feature engineering while achieving superior classification accuracy.

**2.2.2 Cross-View Identification**

Cross-view identification in fruit grading represents a significant advancement that addresses the inherent limitations of single-viewpoint analysis. By capturing and analyzing multiple perspectives of each fruit sample, these systems can construct comprehensive three-dimensional representations that reveal defects or quality attributes otherwise hidden from singular vantage points, particularly for asymmetrical fruits or those with localized imperfections. Cross-view approaches typically employ feature fusion strategies that integrate information from multiple angles, either through early fusion at the image level or late fusion that combines independent predictions from each view. This methodology has proven especially valuable for differentiating between superficial defect and structural defects, a distinction crucial for accurate commercial grading and reducing unnecessary waste in the agricultural supply chain.

**2.3 Current Solutions for Fruit Quality Assessment**

Various agricultural technology firms have implemented deep learning solutions for fruit quality assessment with differing levels of effectiveness and accessibility. Claiming improved accuracy in defect detection and quality measurement, though real-world performance can vary considerably across different operating environments. These commercial systems generally process multiple images per fruit and apply neural network models to classify quality parameters. While these technologies represent significant advancements, they historically remained inaccessible to small-scale producers due to prohibitive costs for complete systems and requiring specialized technical knowledge for maintenance.

**2.4 CNN**

Convolutional Neural Networks (CNNs) have emerged as a cornerstone of modern computer vision, providing robust frameworks for tasks such as image classification, object detection, and semantic segmentation. By automatically learning spatial hierarchies of features directly from pixel-level data, CNNs bypass the need for manual feature extraction. Their architecture typically includes convolutional layers that identify localized patterns, pooling layers that reduce spatial dimensions, and activation functions like ReLU that introduce non-linearity. According to Krizhevsky et al. [[6](#Reference6)] - whose pioneering work on ImageNet classification catalyzed deep learning’s widespread adoption—CNNs excel due to their depth and ability to generalize across visual tasks.

This layered design allows CNNs to learn increasingly abstract representations, where shallow layers detect edges or textures, and deeper layers capture complex object structures. Despite their successes, CNNs still face critical challenges: they require vast labeled datasets, sensitive to adversarial interruption and often act as opaque "black boxes." As a result, recent advances aim to overcome these issues through techniques like transfer learning, adversarial defenses, and explainable AI models that enhance interpretability and trust. These innovations ensure CNNs remain vital tools in both academic research and real-world applications.

A diagram of a diagram of a layer

AI-generated content may be incorrect. Figure 1: Architecture of typical CNN [[10]](#Reference10)

**2.5 ShuffleNet**

ShuffleNet is a highly efficient CNN architecture specifically designed for mobile and embedded devices with limited computational resources. Introduced by Zhang et al [[7].](#Reference7) ShuffleNet addresses the high memory and computation cost typical of deep CNNs by employing two core techniques: pointwise group convolution and channel shuffle .The pointwise group convolution reduces computation by dividing channels into groups, while the channel shuffle operation overcomes the information flow bottleneck by permuting feature channels to ensure inter-group communication. This architecture enables remarkable accuracy and speed trade-offs, achieving lower error rates than comparable lightweight models like MobileNetV3 under the same computational constraints. Subsequent adaptations, such as ShuffleNetV2 and its use in domains like Image classification, further validate its effectiveness across tasks requiring both efficiency and accuracy

**2.5.1 Key Features of ShuffleNet**

ShuffleNet characterized by several design innovations including channel split, channel shuffle, and depthwise separable convolutions [[7]](#Reference7), which collectively reduce computational complexity without compromising accuracy. By reorganizing feature maps across groups through the *channel shuffle* operation, the model ensures effective information flow between feature subsets. Its design adheres to practical guidelines for reducing memory access cost (MAC[)[7]](#Reference7), enhancing parallelism, and minimizing fragmentation, making it particularly suitable for resource-constrained environments.

**2.5.2 How ShuffleNet Work**

**2.5.2.1 Input Processing**

Input data in ShuffleNetV2 is formatted similarly to standard CNNs, where images are represented as 3D tensors composed of height, width, and channel depth (e.g., RGB). This structured input enables convolutional layers to process spatial and color information efficiently through learned filters.

**2.5.2.2 Convolutional Layers**

Unlike traditional CNNs, ShuffleNet introduces a novel architectural design that leverages pointwise group convolutions and a channel shuffle operation to enhance efficiency. Instead of applying standard convolutions across all channels, ShuffleNet divides feature maps into groups, significantly reducing computation. However, to address the limited information exchange between channel groups—a drawback of grouped convolutions—it incorporates a channel shuffle mechanism [[7]](#Reference7), which permutes the output channels across groups to ensure effective inter-group communication. This design enables ShuffleNet to retain strong representational capacity while achieving substantial reductions in computational cost and memory usage, making it well-suited for mobile and low-power applications.

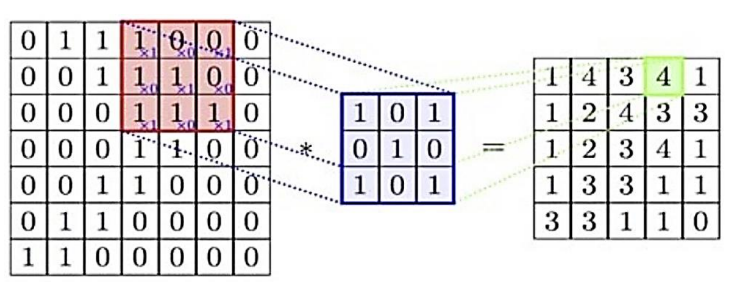


Figure 2: illustration of the convolution operation. [[10]](#Reference10)

**2.5.2.3** **Activation Functions**

Activation functions are indispensable in CNN’s as they introduce non-linearity, allowing these models to capture and represent complex patterns that linear transformations alone cannot address. This capability is vital for tasks like image recognition, where data contains intricate spatial relationships and hierarchical features that require non-linear modeling to detect patterns across various scales and positions. Without activation functions, a CNN would be limited to a sequence of linear operations, rendering it incapable of handling the complexity inherent in real-world data. Positioned typically after convolutional layers and before pooling operations, activation functions enhance feature selectivity by amplifying significant activations and suppressing less relevant ones, thereby improving the network's ability to discern meaningful information. By maintaining gradient flow during backpropagation, they also ensure effective learning, making them a cornerstone of CNN performance across diverse applications.

**2.5.2.4 Hierarchical Feature Extraction**

ShuffleNet's hierarchical feature extraction framework implements an innovative approach to computational efficiency while maintaining representational power through its distinctive channel shuffle operation. At lower layers, the network's point-wise group convolutions extract elementary visual patterns while significantly reducing computational complexity compared to standard convolutions. As features progress through the network, the channel shuffle mechanism facilitates cross-group information exchange, enabling features from different groups to interact and form more complex representations despite the parameter-efficient grouped convolution design. This strategic information flow, combined with bottleneck blocks that carefully balance channel reduction and expansion, allows ShuffleNet to construct increasingly abstract feature hierarchies with substantially fewer operations than traditional architectures. The resulting feature pyramid efficiently captures multi-scale patterns from low-level textures to high-level semantic concepts while maintaining exceptional inference speed on resource-constrained devices.

**2.5.2.5 Flattening in Neural Network Architectures**

Flattening constitutes a critical transformation operation in neural network architectures, particularly within CNN. This preprocessing step serves as the essential bridge between the hierarchical feature extraction components—comprised of convolutional and pooling layers—and the subsequent classification modules that typically employ fully connected layers.

The primary function of flattening is to restructure the multidimensional feature representations (typically three-dimensional tensors) into one-dimensional vectors suitable for processing by dense neural layers. This transformation preserves the informational content encoded by preceding network components while altering only the structural representation. Specifically, flattening maintains the integrity of spatially encoded features while converting them into a vectorized format that facilitates global pattern recognition.

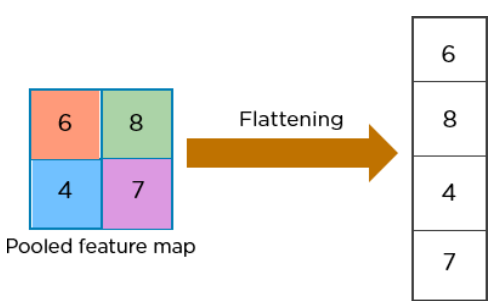
****

Figure 3: example of flattening. [[10]](#Reference10)

**2.5.2.6 Fully Connected Layers**

Fully connected (FC) layers in ShuffleNet serve as the final stage of the network, where high-level features extracted by the convolutional blocks are mapped to the output classes. While the architecture prioritizes computational efficiency through pointwise group convolutions and channel shuffle mechanisms, a standard FC layer is retained at the network's end to perform classification. This component converts the flattened feature representation into a fixed-size output vector.

**2.5.2.7 Backpropagation**

Backpropagation in ShuffleNet follows the conventional gradient-based optimization framework used in deep convolutional neural networks. During training, the network computes the error between predicted outputs and ground truth labels, and this error is propagated backward through all layers—including grouped pointwise convolutions, depthwise convolutions, and the channel shuffle operation—using the chain rule of calculus. Despite ShuffleNet’s architectural innovations for computational efficiency, its backpropagation process remains compatible with standard stochastic gradient descent (SGD) and its variants. The grouped and shuffled structures require appropriate gradient routing, but these operations are differentiable and thus seamlessly integrated into the learning pipeline[[7]](#Reference7).

**2.5.3 Applications**

ShuffleNet has numerous applications across diverse domains, leveraging its computational efficiency for resource-constrained environments:

● **Mobile vision systems**: Powers real-time image classification and object detection on smartphones with minimal battery consumption.

● **Agricultural monitoring**: Enables on-device fruit quality assessment, pest detection, and crop disease identification in field conditions.

● **Medical diagnostics**: Facilitates portable skin lesion classification, electrocardiogram (ECG) analysis, and medical image screening on low-power clinical devices.

● **Smart retail**: Supports product recognition and inventory tracking on edge devices in store environments.

● **IoT sensors**: Enables visual processing on battery-powered IoT devices for extended deployment periods.

ShuffleNet’s architecture makes it particularly valuable in scenarios requiring energy efficiency while maintaining acceptable accuracy.

**2.6 ShuffleNet V2**

ShuffleNetV2 is a lightweight convolutional neural network architecture designed to achieve a better trade-off between speed and accuracy, particularly for deployment on mobile and embedded devices. Building on the original ShuffleNet, the V2 version introduces critical refinements aimed at addressing practical efficiency limitations such as memory access cost (MAC), fragmentation, and parallelism[[11]](#Reference11). Its design was driven by empirical guidelines rather than theoretical assumptions, emphasizing actual hardware performance rather than only Floating Point Operations Per Second(FLOPs). ShuffleNetV2 maintains the use of pointwise group convolution and channel shuffle operations but simplifies the block structure to enhance runtime performance and reduce memory overhead on real devices.

**2.6.1 Core Building Block**

The central unit of ShuffleNetV2 is a split-branch structure. Each block divides the input feature map into two branches: one branch remains unchanged (identity), while the other undergoes transformation via depthwise separable convolutions and pointwise convolutions. After processing, the two branches are concatenated, and a channel shuffle operation is applied to enable cross-branch information exchange. This design ensures rich feature propagation while maintaining low computational cost. Additionally, the architecture avoids fragmentation and excessive element-wise operations, which are known to hinder actual inference speed.

A diagram of a computer flowchart

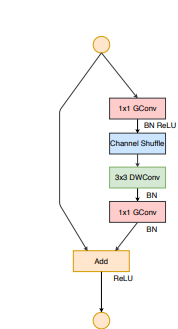
AI-generated content may be incorrect.

Figure 4: example of ShuffleNet V1 Block.[[11]](#Reference11) Figure5: example of ShuffleNet V2 Block . [[11]](#Reference11)

**2.6.2 Network Architecture and Staging**

The full ShuffleNetV2 model is constructed by stacking these core blocks in stages. Each stage includes downsampling units and non-downsampling blocks arranged to increase feature depth and spatial abstraction[[11]](#Reference11). The downsampling blocks utilize stride-2 operations with concatenation and channel shuffle to halve spatial dimensions while preserving critical information. Intermediate stages vary based on model scale (e.g., 0.5x, 1.0x, 1.5x, 2.0x), allowing flexibility across hardware constraints and performance targets.

**2.6.3 ShuffleNetV2 Architecture Diagram**

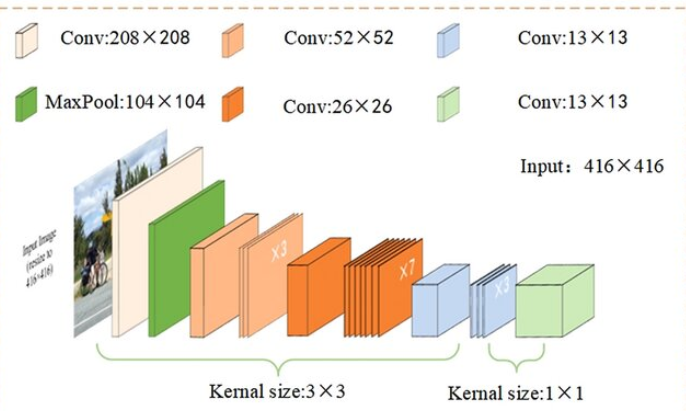
****

Figure 6: ShuffleNet V2Architecture[[12]](#Reference12).

# 3 Research / Engineering Process

**3.1 Development Methods**

**3.1.1 Python for AI and Machine Learning as server**

Python offers numerous benefits; its simplicity and readability are ideal for rapid development and prototyping. The language boasts a rich ecosystem of libraries and frameworks, such as OpenCV and PyTorch, which provide pre-built tools for computer vision and machine learning tasks. Additionally, Python's extensive community support and a wealth of online resources make problem-solving and collaboration much easier. Its versatility and ability to integrate seamlessly with other technologies make Python a powerful and efficient choice for developing advanced AI and neural network-based applications[[13]](#Reference13) .

**3.1.2 React for the Frontend**

We are going to use React for the frontend because it provides a flexible and efficient way to build user interfaces. React's component-based architecture promotes reusability and maintainability, making it easier to manage complex UIs. Its virtual Document Object Model (DOM) ensures high performance by minimizing updates to the real DOM. Additionally, React has a strong developer community and a rich ecosystem of tools and libraries, which accelerates development and enhances productivity[[14]](#Reference14).

**3.1.3 MongoDB for Database**

MongoDB's document-oriented architecture effectively accommodates the heterogeneous nature of agricultural data, including visual information, measurements, and classification results. Its flexible schema design adapts seamlessly to evolving agricultural parameters without requiring structural modifications, while robust indexing enables quick retrieval based on multiple attributes. Additionally, MongoDB's horizontal scalability ensures the system can handle increasing data volumes during peak seasons while maintaining consistent performance across distributed environments[[17]](#Reference17).

**3.2 3D Rotation Estimation Fruits Dataset**

The 3D Rotation Estimation Fruits dataset offers an excellent foundation for automated fruit quality assessment, providing comprehensive visual representation through images captured from multiple angles as fruits rotate. This dataset is particularly valuable due to its diverse collection of fruit specimens with varying shapes, textures, and color profiles, which enables the development of classification algorithms that can perform well in real industrial settings. A significant advantage of this dataset is its rotational completeness, which helps our model develop orientation-independent feature recognition—essential for practical inspection systems where fruits are randomly positioned on conveyor belts. The well-structured nature of the dataset, with each fruit carefully documented through consistent angular increments, provides an ideal testing environment for validating our classification accuracy and ensures the system can reliably assess quality metrics across different fruit types in production environments[[18]](#Reference18).

**3.3 Preprocessing Methods**

Effective preprocessing of digital fruit images plays a crucial role in ensuring accurate quality assessment through computer vision techniques. By standardizing data through normalization and spatial resizing, the system establishes consistency across diverse image sources, enabling more reliable feature extraction. Noise reduction via Gaussian filtering preserves essential texture details while eliminating acquisition artifacts that could otherwise interfere with detection of subtle surface exceptions. The application of Contrast Limited Adaptive Histogram Equalization (CLAHE) in the LAB color space significantly improves the visibility of defining characteristics by enhancing local contrast while maintaining color fidelity across varying illumination conditions.

**3.3.1 Gaussian Filter**

The implementation of Gaussian filtering in the preprocessing pipeline represents a critical noise reduction strategy for enhancing fruit quality assessment. By implementing a carefully tuned Gaussian filter, the system reduces unwanted high-frequency noise while maintaining the critical shape characteristics and surface textures that signal fruit quality. This preprocessing step creates a foundation for subsequent contrast enhancement techniques by smoothing random intensity variations that could otherwise compromise the accuracy of defect detection algorithms. Ismail and Malik (2021) demonstrated in their real-time visual inspection system that selective noise filtering techniques like Gaussian blur significantly improve feature extraction quality, particularly when coupled with adaptive contrast enhancement for agricultural produce with variable surface characteristics [[2]](#Reference1).

**3.3.2 CLAHE Settings**

Clip limit parameter in CLAHE controls the threshold for contrast enhancement to prevent over-amplification of noise in homogeneous regions. After experimentation with various settings, we determined that a clip limit of 2.0 provides an optimal balance between enhancing subtle surface defects and maintaining natural fruit appearance. Higher values tend to introduce artificial textures and amplify noise, while lower values provide insufficient contrast for detecting minor blemishes.

A comparison of the moon

AI-generated content may be incorrect.

A comparison of a planet

AI-generated content may be incorrect. Figure 7: demonstrating the CLAHE effect for the chosen Clip-Limit.

Figure 8: demonstrating the CLAHE effect for lower Clip-Limit.

**3.3.3 Preprocessing Code**

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 9: Image preprocessing implementation for the fruit grading system.

**3.4 Choose the CNN model**

The choice of an appropriate CNN architecture is fundamental to the development of an efficient and accurate automated fruit grading system, particularly under constraints imposed by edge deployment environments. Among leading lightweight models, MobileNetV3, EfficientNet-Lite, and ShuffleNetV2 each provide compelling advantages for low-resource applications. MobileNetV3 is characterized by its use of inverted residual blocks and linear bottlenecks, which effectively reduce latency and computational load while maintaining robust performance on mobile platforms [[15]](#Reference15). EfficientNet-Lite extends the EfficientNet framework by applying compound scaling to optimize the network’s depth, width, and resolution, resulting in a highly accurate yet compact model suitable for edge inference[[16]](#Reference16). ShuffleNetV2, in contrast, is explicitly designed with empirical efficiency guidelines, incorporating mechanisms such as pointwise group convolutions, channel splitting, and channel shuffle to minimize memory access cost and improve hardware-level parallelism [[7]](#Reference7).These architectural innovations collectively position each model as a strong candidate for real-time, resource-aware fruit grading applications.

**3.4.1 Performance Metrices**

|  |  |  |  |
| --- | --- | --- | --- |
| Feature / Metric | ShuffleNetV2 | MobileNetV3 | EfficientNet-Lite |
| Model Size | ~5.0 MB (1.0× variant) | ~14 MB | ~20–25 MB (Lite0–Lite2 variants) |
| Parameters (M) | ~1.4M (ShuffleNetV2 1.0×) | ~3.4M | ~4.7M (EfficientNet-Lite0) |
| FLOPs | ~146M (1.0×) | ~300M | ~390M (Lite0), scalable |
| ImageNet Top 1 Accuracy | ~72.6% | ~71.8% | ~76.3% (EfficientNet-Lite0) |
| Inference Speed | High | Moderate | Lower than others due to deeper layers |
| Hardware Efficiency | Excellent (minimized memory access cost, low fragmentation) | Good (optimized for mobile devices) | Fair to good (better accuracy but more memory and compute intensive) |
| Inference Speed (ms) | ~155ms | ~157ms | ~210ms |
| Framework Support | PyTorch (primary), TensorFlow | TensorFlow (primary), PyTorch | TensorFlow (primary), PyTorch |

Based on the comparative analysis ShuffleNetV2 demonstrates exceptional memory efficiency with its low parameter count (~2.3M for 1.0×) and minimal memory access cost, making it particularly suitable for highly constrained environments like embedded systems, though it achieves a more modest accuracy (69.4% for 1.0×). MobileNetV3 presents a balanced profile with its hardware-optimized architecture, offering higher accuracy (75.2% for large variant) while maintaining reasonable efficiency, though its performance can vary significantly across different neural processing units and inference engines. EfficientNet exhibits superior classification accuracy (77.1% for B0) and excellent feature transferability across domains, but requires substantially more computational resources with higher FLOPs (~390M) and larger model size (~29MB for B0), potentially limiting it’s deploy ability on low-power edge devices used in agricultural settings.

**3.4.2 Chosen Model**

ShuffleNetV2 was selected as the optimal architecture for our system due to its exceptionally low memory footprint and minimal computational requirements, enabling deployment on affordable edge devices. The architecture's channel shuffle mechanism and balanced convolutional design significantly reduce inference latency to under 160ms on standard mobile processors, meeting our critical requirement for high-throughput grading. This efficiency-focused approach aligns perfectly with one of our project's core objective of making the system accessible to producers with limited technological resources while maintaining grading accuracy sufficient for commercial standards. By prioritizing speed and affordability through ShuffleNet's lightweight design, we can deliver a cost-effective solution that processes fruits in real-time without requiring expensive specialized hardware.

**3.4.3 Comparison Between Models**

**3.4.3.1 Comparison of ShuffleNetV2, MobileNetV3 and EfficientNet-Lite**

### **3.4.3.1.1 Methodology and Evaluation Framework**

To establish the optimal neural network architecture for our real-time fruit grading system, we conducted a comprehensive comparative analysis of three state-of-the-art lightweight convolutional neural networks: ShuffleNetV2, MobileNetV3, and EfficientNet-Lite0. The evaluation was performed using our custom preprocessing pipeline on a representative test dataset comprising tomatoes, oranges, and mandarins from our MongoDB-structured image database. Each model was assessed across four critical performance dimensions essential for embedded agricultural applications: inference latency, feature extraction efficiency, model size and memory requirements (model parameters).

### **3.4.3.1.2 Performance Analysis Results**

The experimental results demonstrate significant performance variations among the evaluated architectures, with ShuffleNetV2 exhibiting superior computational efficiency across all measured metrics. ShuffleNetV2 achieved an average inference time of 30.581 milliseconds and feature extraction time of 1 millisecond (ms), representing approximately 73% and 70% faster processing compared to MobileNetV3 (108.232ms inference, 38.809ms feature extraction), 86% and 83% faster than EfficientNet-Lite (217.999ms inference, 70.99ms feature extraction), respectively. These timing advantages are particularly crucial for our conveyor belt system, where fruits must be classified while traveling through the inspection box containing the cameras. The system architecture demands that complete fruit classification be determined before the fruit reaches the end of the conveyor belt, creating stringent temporal constraints that necessitate rapid processing capabilities. The substantial performance gap between ShuffleNetV2 and the alternative architectures directly impacts the feasibility of real-time classification within these physical system constraints.

### **3.4.3.1.3 Resource Utilization and Model Efficiency**

From a resource utilization perspective, ShuffleNetV2’s architectural efficiency is evident in its compact model footprint of 8.754 MB and 2.3 million parameters, substantially lower than both MobileNetV3 (21.01 MB, 5.5 million parameters) and EfficientNet-Lite (17.907 MB, 4.7 million parameters). This reduced computational complexity translates directly to lower power consumption and memory usage, critical factors for deployment on resource-constrained embedded systems typical in agricultural environments. The channel shuffle operations inherent in ShuffleNetV2’s design enable efficient information flow between channel groups while maintaining computational efficiency, making it particularly well-suited for our multi-view temporal processing pipeline where features from multiple camera angles must be integrated rapidly.

### **3.4.3.1.4 Hardware Configuration and Testing Environment**

The comparative analysis was conducted on a MacBook Pro equipped with an M4 Pro processor, representing a high-performance computing environment with advanced neural processing capabilities and substantial memory bandwidth [[19]](#Reference19). This testing platform provided optimal conditions for model evaluation, allowing each architecture to demonstrate its maximum potential performance without hardware-imposed constraints. The M4 Pro's dedicated neural engine and high-speed unified memory architecture ensured that the measured performance differences accurately reflected the inherent computational characteristics of each model rather than system bottlenecks.

### **3.4.3.1.5 Target Hardware Deployment Considerations**

The actual fruit grading system will be deployed on a Raspberry Pi 5, which represents significantly more constrained computational resources compared to the testing environment [[20]](#Reference20). This embedded platform features lower processing power, reduced memory bandwidth, and limited parallel processing capabilities typical of edge computing devices used in agricultural applications. Under these resource-constrained conditions, the performance differences between architectures will become more pronounced, with lighter models like ShuffleNetV2 maintaining acceptable processing speeds while heavier architectures may experience substantial degradation in real-time performance. The computational efficiency advantages demonstrated by ShuffleNetV2 in our high-performance testing environment will translate to even greater practical benefits in the target embedded deployment, where every millisecond of processing time directly impacts the system's ability to classify fruits within the conveyor belt's temporal constraints.

### **3.4.3.1.6 Implications for Real-Time Fruit Grading Applications**

These performance characteristics align with the operational requirements of our conveyor belt inspection system, where the combination of real-time processing constraints, limited computational resources, and the need for consistent performance across varying lighting conditions necessitates an architecture that balances accuracy with computational efficiency. The empirical evidence supports ShuffleNetV2 as the optimal choice for our fruit grading application, providing the necessary processing speed for real-time operation while maintaining a minimal resource footprint suitable for embedded deployment in agricultural settings.

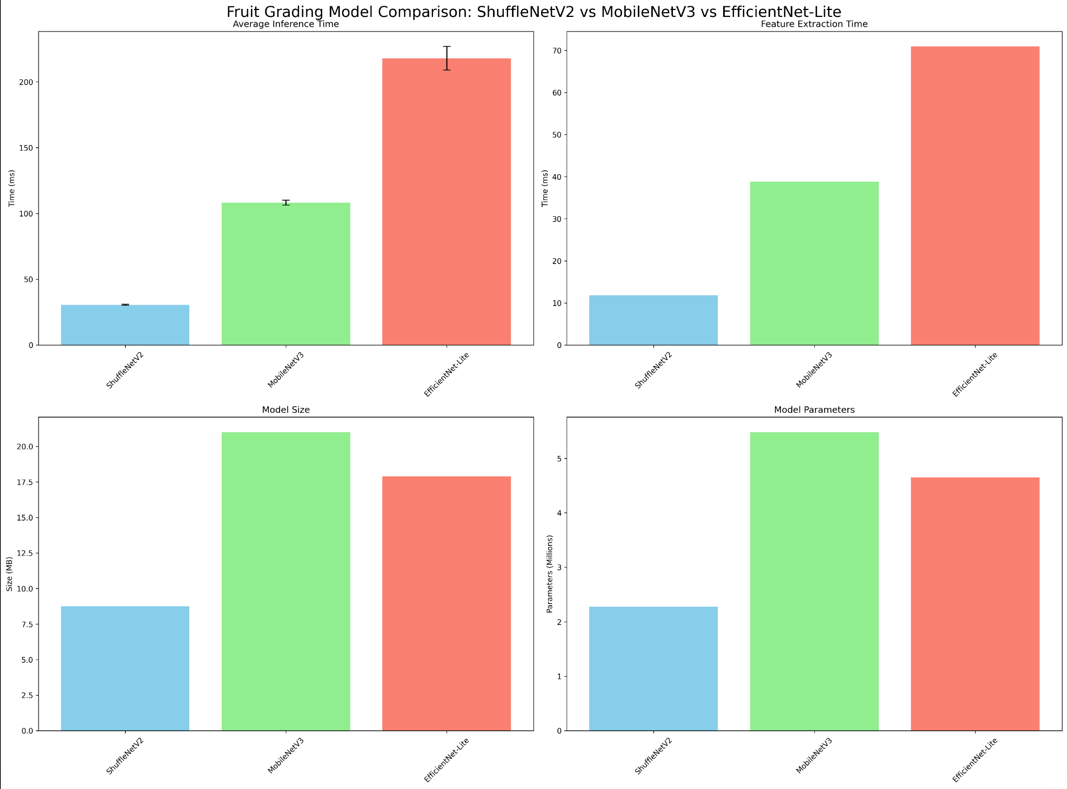
**3.4.3.1.7 Test Results**

Figure 10: Performance comparison showing ShuffleNetV2's superior efficiency in inference time, feature extraction, model size, and parameters versus MobileNetV3 and EfficientNet-Lite.

### **3.5 Activation Functions Overview**

### **3.5.1 Rectified Linear Unit (ReLU)**

The ReLU is the most prevalent activation function in CNNs, prized for its simplicity, computational efficiency, and effectiveness in training deep networks. Defined mathematically as f(x) = max(0, x), ReLU outputs the input value directly if it is positive, and zero otherwise, activating neurons selectively based on this threshold[[8]](#Reference8). This property fosters sparsity in the network, as many neurons produce zero outputs, which can enhance computational efficiency and reduce overfitting by creating more concise representations. ReLU also combats the vanishing gradient problem—a challenge in deep learning where gradients shrink during backpropagation—by providing a constant gradient of 1 for positive inputs, thus enabling smoother and more robust training. However, ReLU has its limitations, notably the "dying ReLU" issue, where neurons can become permanently inactive if they consistently receive negative inputs.

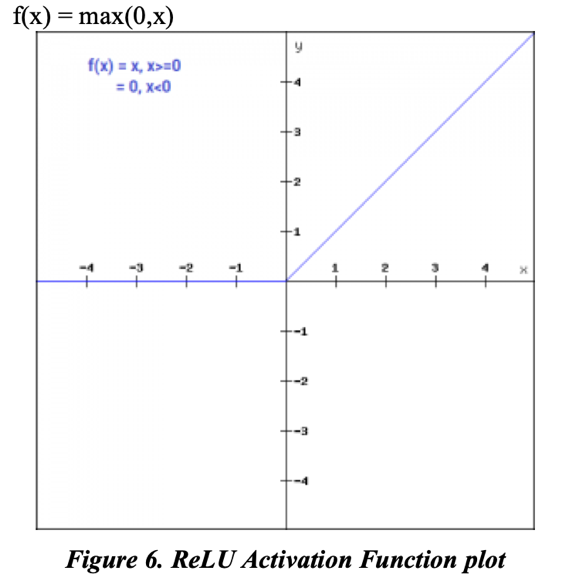


Figure 11: ReLU Activation Function Plot [[8]](#Reference8).

### **3.5.2 Sigmoid**

The sigmoid activation function is a mathematical function commonly used in neural networks, especially for binary classification tasks. It takes any real-valued number and maps it to a range between 0 and 1, which can be interpreted as a probability [[8]](#Reference8). Its mathematical formulation is given by:

where e is the base of the natural logarithm. This function produces an S-shaped curve, often referred to as the logistic function, and is differentiable, making it compatible with backpropagation in neural network training. However, a key limitation is the vanishing gradient problem: for very large positive or negative inputs, the gradient becomes nearly zero, slowing down learning in deep networks.

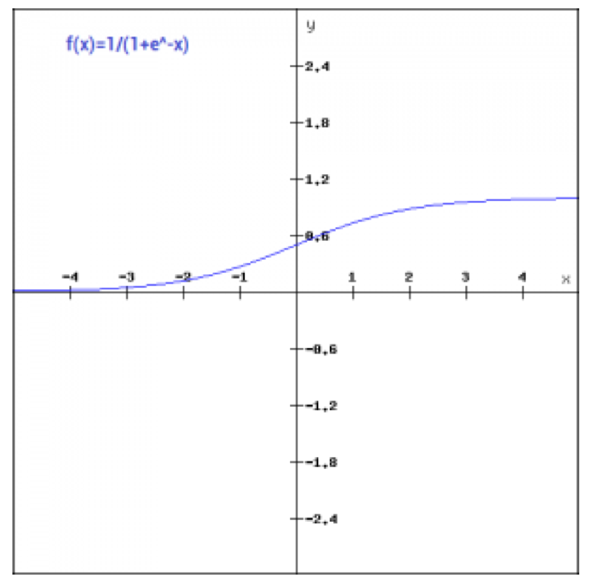


Figure 12: Sigmoid Activation Function Plot [[8]](#Reference8).

### **3.5.3 Hyperbolic Tangent (Tanh)**

The hyperbolic tangent (tanh) activation function is a zero-centered function that maps real-valued inputs to the interval (-1, 1) [[8]](#Reference8). Its mathematical expression is:

Being zero-centered, tanh can improve learning dynamics compared to sigmoid by reducing bias in gradients during backpropagation. However, it also suffers from the vanishing gradient problem for large positive or negative inputs, which can impede training in deeper architectures.

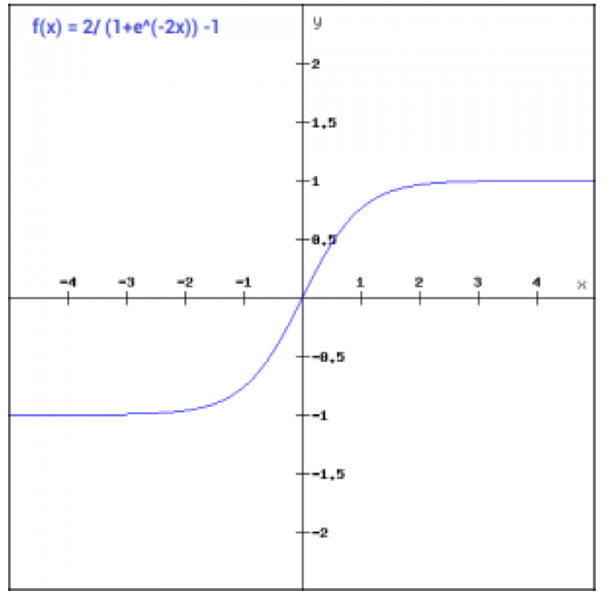


Figure 13: Tanh Activation Function Plot [[8]](#Reference8).

### **3.5.4 Softmax**

The softmax activation function is commonly used in the output layer of neural networks for multi-class classification tasks. It converts a vector of real-valued logits into a probability distribution over Kclasses, ensuring all outputs are non-negative and sum to one:

This allows the model to express confidence in each class prediction, which is critical for distinguishing between similar fruit types. Softmax emphasizes higher input values, making it effective for identifying the most probable class.

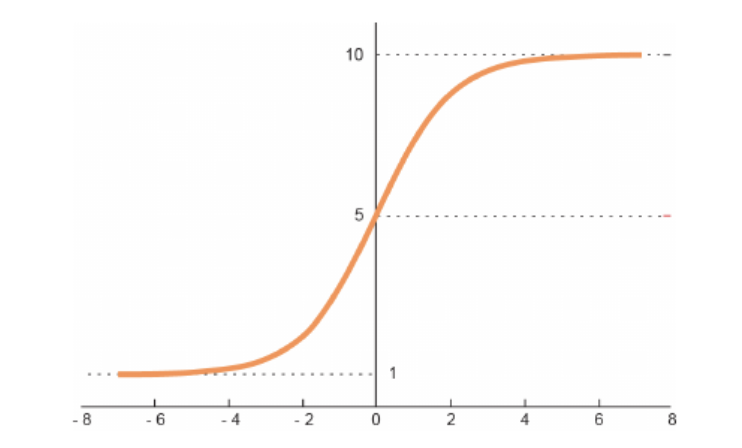


Figure 14: Softmax Activation Function Plot [[21]](#Reference21).

**3.5.5 Chosen Activation Functions**

In the project, the ReLU activation function was employed for the hidden layers, while the Softmax activation function was utilized in the output layer. ReLU outputs zero for negative inputs and retains positive values unchanged. This property mitigates the vanishing gradientproblem, thereby facilitating more efficient and stable training in deep neural networks. Additionally, its computational simplicity contributes to faster convergence, rendering it a practical and widely adopted choice for hidden layers. The Softmax activation function was selected for the output layer due to the multi-class classification nature of the task. Softmax converts raw output scores into a probability distribution over multiple classes, allowing the model to assign varying levels of confidence to each class. Unlike activation functions such as sigmoid or tanh, which are typically suited for binary classification tasks (i.e., distinguishing between two classes), Softmax is specifically designed to handle scenarios where inputs must be classified into more than two mutually exclusive categories, making it particularly suitable for this project.

**3.6 Methods to measure precision**

**3.6.1 Sensitivity (True Positive Rate)**

Sensitivity, also known as the true positive rate, refers to the ability of a prediction model to correctly identify instances belonging to a specific class within a dataset (Ismail & Malik, 2021) [[2]](#Reference1). In our evaluation framework, sensitivity serves as a critical performance metric, offering insight into the model’s effectiveness in detecting samples from targeted quality categories. This becomes particularly important in contexts such as agricultural quality assessment, where class imbalance can distort performance evaluations. As highlighted by Ismail and Malik, incorporating sensitivity into the assessment enables a more nuanced understanding of classification accuracy, ensuring that underrepresented but important categories are not overlooked.



Figure 15: The Sensitivity formula [[2]](#Reference1) .

**3.6.2 specificity (True Negative Rate)**

Specificity, also known as the true negative rate, measures the model’s ability to correctly identify and exclude instances that do *not* belong to a particular class (Ismail & Malik, 2021) [[2]](#Reference1). This metric is especially important in quality assessment tasks where false positives—incorrectly labeling substandard items as acceptable—can have significant downstream consequences. In the context of agricultural quality control, high specificity ensures that defective or non-conforming samples are reliably filtered out, thus safeguarding product integrity. As noted by Ismail and Malik, incorporating specificity into model evaluation enables a more balanced view of classification performance, particularly in datasets with imbalanced class distributions.



Figure 16: The Specificity formula [[2]](#Reference1) .

**3.6.3 Accuracy (Overall Classification Performance)**

Accuracy represents the proportion of correct predictions—both positive and negative—out of the total number of cases. It is a commonly used metric to evaluate the overall effectiveness of classification models, such as those applied in agricultural quality assessment. While useful for a quick performance overview, accuracy can be misleading in imbalanced datasets, where correct classification of the dominant class inflates the metric. As emphasized by Ismail and Malik (2021), combining accuracy with other metrics like sensitivity and specificity provides a more reliable evaluation of model performance (Ismail & Malik, 2021) [[2]](#Reference1).



Figure 17: The Accuracy formula [[2]](#Reference1) .

# 4 Work Artifacts

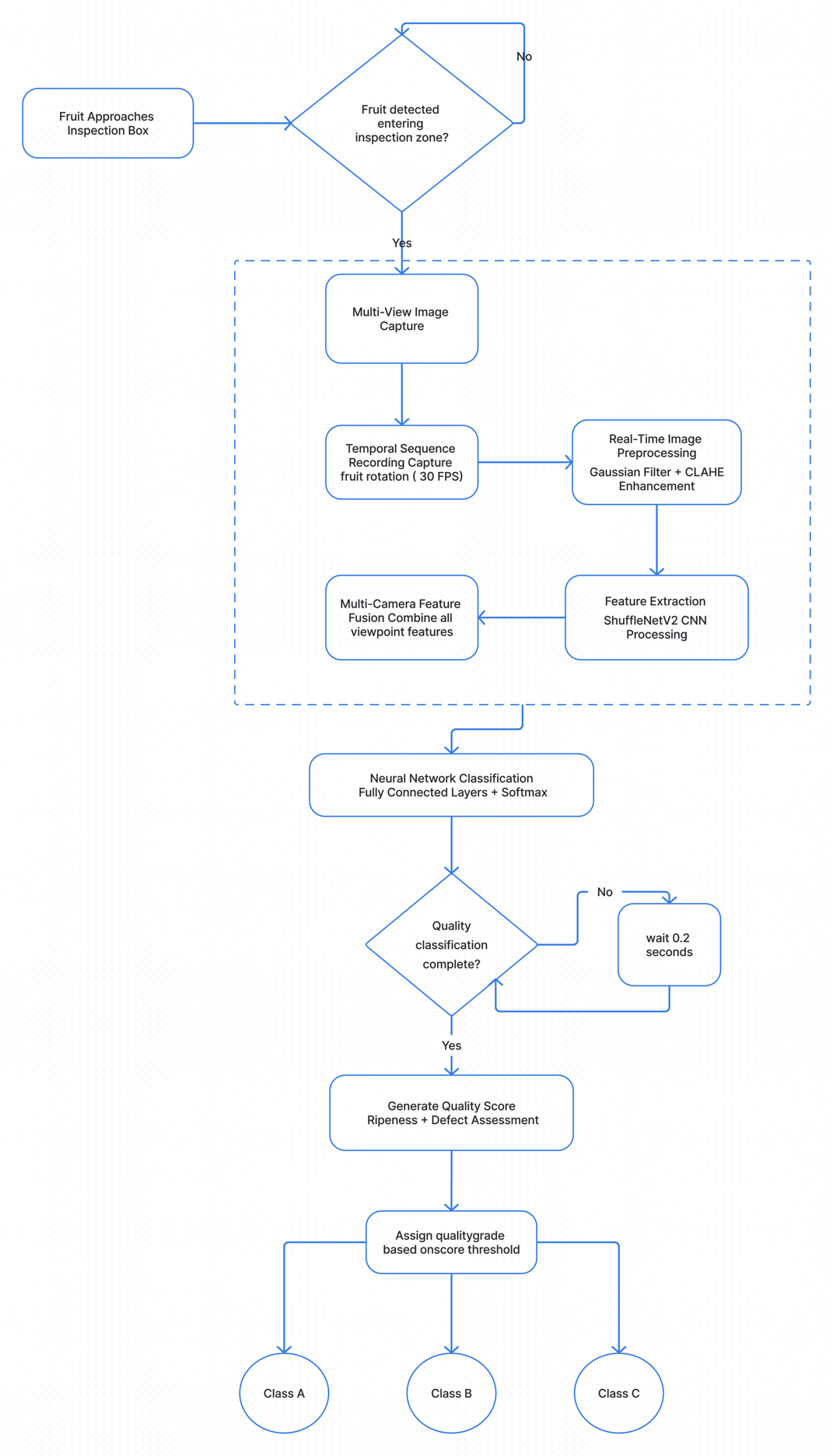
**4.1 System Flow**

Figure 18: System Flow

References

[2] N. Ismail and O.A. Malik, "Real-time visual inspection system for grading fruits using computer vision and deep learning techniques," Information Processing in Agriculture, vol. xxx, pp. 1-13, 2021.

[3] Jacques, B. J. (2018). A qualitative study exploring labour challenges and constraints and the labour hiring and management decisions on fruit, vegetable and horticulture specialty farms [Master's thesis, North Carolina A&T State University]. ProQuest Dissertations & Theses Global.

[4] Labor-Intensive U.S. Fruit and Vegetable Industry Competes in a Global Market. USDA Economic Research Service, 2010.

[5] U.S. Fruit and Vegetable Industries Try To Cope With Rising Labor Costs. USDA Economic Research Service, 2022.

[6] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). *ImageNet Classification with Deep Convolutional Neural Networks*. Advances in Neural Information Processing Systems, 25, 1097–1105.

[7] Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). *ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6848–6856.

[8] Kumar, S., & Sharma, A. (2021). *Activation Functions in Neural Networks*. International Journal of Engineering, Applied Sciences and Technology, 5(12), 310-316.

[9] Hu, Y., Huber, A., Anumula, J., & Liu, S. (2018). *Overcoming the vanishing gradient problem in plain recurrent networks*. arXiv preprint arXiv:1801.06105v3.

[10] Gupta, R. S., & Pateriya, R. K. (2022). *Convolutional Neural Network and its Architectures*. International Research Journal of Engineering and Technology (IRJET), 9(1), 653-656.

[11] Ma, N., Zhang, X., Zheng, H., & Sun, J. (2018). *ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design*. Proceedings of the European Conference on Computer Vision (ECCV), 116-131.

[12] Khebbache, S., Messai, M., & Mahdid, M. (2021). *L-Net: lightweight and fast object detector-based ShuffleNetV2*. International Journal of Computing and Digital Systems, 10(1), 1277-1287.

[13] Python <https://www.python.org/>

[14] React <https://react.dev/>

[15] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 4510-4520.

[16] Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. Proceedings of the 36th International Conference on Machine Learning (ICML), 6105-6114.

[17] <https://www.mongodb.com/>

[18] <https://github.com/alalbiol/3d-rotation-estimation-fruits/tree/main/data>

[19] Apple Inc. (2024). "Apple introduces M4 Pro and M4 Max." Apple Newsroom. Retrieved from https://www.apple.com/newsroom/2024/10/apple-introduces-m4-pro-and-m4-max/

[20] Raspberry Pi Foundation. (2024). "Raspberry Pi 5 Technical Specifications and Performance Benchmarks." Official Documentation. Retrieved from <https://www.raspberrypi.com/documentation/computers/processors.html#bcm2712>

[21] Es-sabery, F., Hair, A., Qadir, J., & Sainz de Abajo, B. (2021). Sentence-level classification using parallel fuzzy deep learning classifier. IEEE Access, PP (99), 1–1. https://doi.org/10.1109/ACCESS.2021.3053917