Utilizing Ensemble Learning for Enhanced Fruit Freshness Classification

Krish Nair (50%)
Penn State Computer Science
Penn State Harrisburg

Casey Detwiler (50%)
Penn State Computer Science
Penn State Harrisburg

***Abstract*—AI usage in large-scale food production requires a high-degree of accuracy. AI models often have many pitfalls that make their use in such situations questionable. One approach to increase model accuracy for these safety-critical systems is an ensemble model, where multiple AI models are trained in parallel to hopefully attain better overall performance than any single model alone. We trained VGG16, InceptionNetV3, and EfficientNetV2 to classify fruit as fresh or rotten independently, then combined them in 5 distinct ways. InceptionNetV3 performed the best out of any solo model, reaching 96.90% on the test data set by itself. Our best combination model, which utilized both InceptionNetV3 and VGG16, performed slightly better, reaching a testing accuracy of 97.77%. Many other combinations of models performed similarly well.**

***Index Terms*—Ensemble Learning, Classification, Neural Network, Food Safety**

I. Introduction

In large-scale industrial food production, a degree of quality assurance is vital. Machinery can process vast quantities of food very quickly, and ensuring that spoiled or other samples unfit for human consumption are not used in production is important for customer safety. Human overseers are usually given the final call, but humans are not perfect. They can be tired, zone-out from repetitive actions, and cost a wage. Monotonous tasks that are prone to human error are prime targets for artificial intelligence systems, but they often suffer from their own issues.

An ensemble model is one such approach to mitigate the issues that AI systems often face. An ensemble model consists of training multiple other models separately, and combining their outputs in some meaningful way to reach a final prediction. The hope is that multiple AI models will be able to "cover" for each others' mistakes, increasing the overall performance of the combined model.

In this paper we create an ensemble model for classifying the freshness of fruit. We trained 3 separate models on the same data set: VGG16, InceptionNetV3, and EfficientNetV2. With these models, we combined them in 5 distinct ways, including the common method of averaging, and 4 of our own novel approaches.

Our best performing model achieved 97.77% accuracy using InceptionNetV3 and VGG16 with one of our novel combination approaches, barely edging out the best performing solo model, InceptionNetV3, at 96.90%. Several other combinations of models reached over 97% accuracy, but the combination method seemed to be less important than the quality of the models used.

II. Related Works

There have been different methods for food grading over the years. Before the use of deep neural networks, computer vision systems were created to determine food freshness through a color scale [1]. Image segmentation was used to differentiate the fruits from the background and principal component analysis is used to reduce the dimensionality of the features gathered from the images. A model is then fit to the data to develop a color scale based on the food images. While this was a helpful method for automating food grading, newer applications deep learning help to achieve more detailed analysis of the foods.

For deep learning models, a lot of data is needed to train the models to learn features about the data. For food grading, available datasets are composed of hundreds of images of various foods. Thus, data augmentation is very common to achieve a wider variety of training images [2] [3] [4].

Various deep learning models have been applied to food grading including convolution neural networks. Custom CNN's [2] and more complex CNN's, such as EfficientNet and MobileNet [3], have been applied to food grading tasks where the model predicts the quality of a single type of food based on images of various food qualities. The results from these models on the used test sets are very promising having accuracies ranging from 94% to 99% in some cases.

Support vector machines have also had good success in food grading [3] [4] [5] . SVM's were found to outperform simpler models by a large margin by achieving accuracies of around 98% for different datasets. Complex models, such EfficientNet, still achieved a slightly higher accuracy when compared in the study. However, in terms of application of the models, SVM showed higher promise due to its efficiency.

III. Methodology

A. Overview

In this paper we utilized an ensemble model to hopefully enhance any individual model's classification performance. We selected VGG16, InceptionNetV3, and EfficientNetV2 as our models. We utilized transfer learning, trained them on our data set of fruits, and then combined them in 5 distinct ways. We utilized a more standard approach for combining their outputs including average pooling and 4 novel ones. We evaluated all possible combinations of models with all possible combination methods for a total of 20 distinct combinations.

For implementation we utilized the Keras library for Python. Our code is available here.

B. Data Set

For training our models we utilized Meshram and Patil's data set of fresh and rotten fruits [6]. The data set includes a variety of fruits but we focused our study specifically on apples, bananas, and oranges. The final data set contains 6,865 total instances of apples, bananas, and oranges divided into fresh and rotten categories. We designed our model to distinguish between fresh and rotten fruit, so we considered all fruit types within the data set and only cared about the fresh and rotten labels. In total, there are 4804 instances available for training, 1030 instances for validation, and 1031 instances for testing. The data set contains a wide variety of camera angles, lighting angles, and environments. This will allow the networks to learn more generalized cases. Figure 1 shows samples from the fresh class and Figure 2 shows samples from the rotten class.

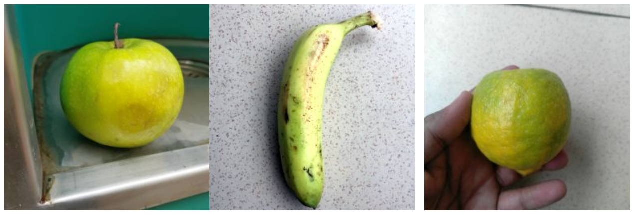
Fig. 1: Sample Fresh Instances [6]

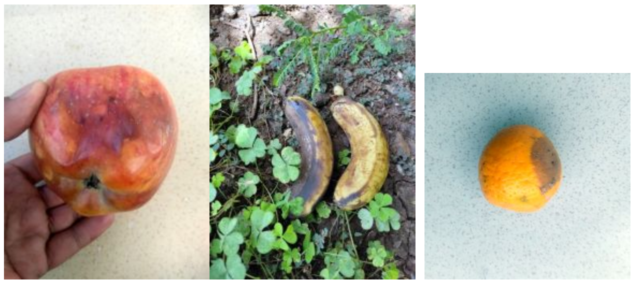
Fig. 2: Sample Rotten Instances [6]

C. Networks

We trained 3 neural networks for our ensemble model: VGG16, InceptionNetV3, and EfficientNetV2. They were all trained independently, and were combined in various ways after training.

1) VGG16: We utilized transfer learning from the pre-trained VGG16 on the ImageNet [7] dataset. We removed the top layer from this preset and froze all other layers before adding a single output node for our binary classification problem. Figure 3 shows the unmodified structure of VGG16. Our slightly modified version contains 14,715,201 parameters.

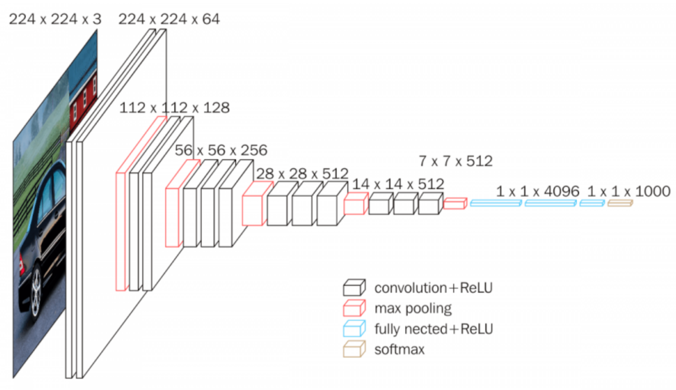

Fig. 3: Unmodified VGG16 Structure [8]

2) InceptionNetV3: Similarly to VGG16, we utilized transfer learning from the pretrained InceptionNetV3 on the ImageNet [7] dataset. We removed the top layer from this preset and froze all other layers before adding a single output node for our binary classification problem. Figure 4 shows the unmodified structure of InceptionNetV3. Our slightly modified version contains 21,802,784 parameters.

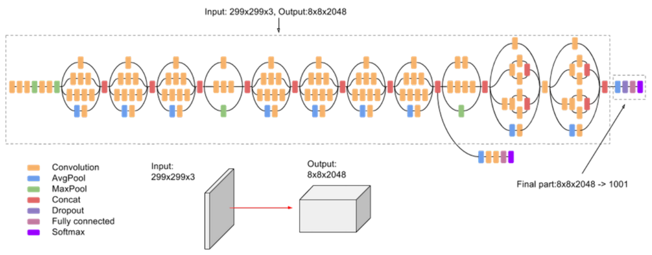

Fig. 4: Unmodified InceptionNetV3 Structure [9]

3) EfficientNetV2: Like the other networks, we utilized transfer learning from the small preset of EfficientNetV2 pretrained on the ImageNet [7] dataset. We removed the top layer from this preset and froze all other layers before adding a single output node for our binary classification problem. Figure 5 shows the unmodified structure of EfficientNetV2. Our slightly modified version contains 20,331,360 parameters.

D. Training

Each model was trained and evaluated separately to gauge their individual performances before combination. VGG16 was trained for 25 epochs, InceptionNetV3 was trained for 15 epochs with early stopping, and EfficientNet was trained for 20 epochs with early stopping. We utilized image augmentation for the training data set with a 15 degree rotation range, a horizontal shift range of 0.2, a zoom range of 0.8x to 1.2x and occasional horizontal flipping. Figures 6, 7, and 8 showcase the training of VGG16, InceptionNetV3, and EfficinetNetV2 respectively.

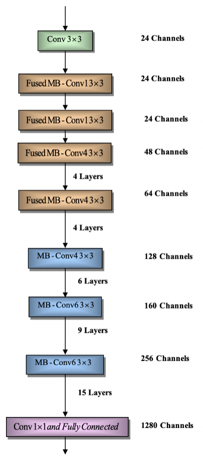

Fig. 5: Unmodified EfficientNetV2 Structure [10]

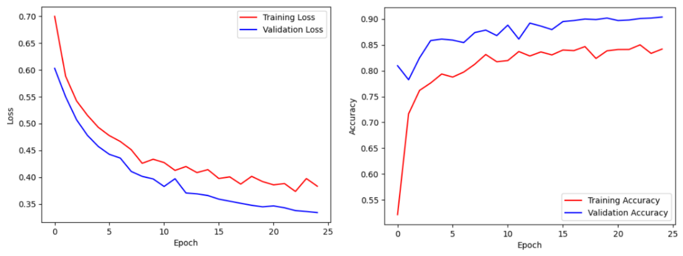

Fig. 6: VGG16 Training

E. Combination

With 3 models at our disposal, there are numerous ways to possibly combine them. We considered all possible sets containing at least 2 models and utilized five distinct methods for pooling the individual networks' results to produce the final prediction. Figure 9 showcases the overall architecture of our model for when all three models are included. The structure is identical when combining only two models.

1) Average Voting: Average voting simply takes the average of all model's predictions from 0 to 1 and selects the class that is closer to the final average. Mathematically, the prediction of the overall model on an input, denoted $P(i)$, is given by

$$P(i) = round(\frac{\sum_{m \in Models} \hat{y}_m}{|Models|})$$

where $Models$ represents the set of all models used for this combination, $\hat{y}_m$ represents model m's prediction ranging between 0 and 1, and $round()$ rounds to the nearest integer.

2) First Past the Post Voting: We utilized simple first-past-the-post voting for our first novel combination method. In this method, each network's prediction is weighed the same, and whichever option is voted the most is the class chosen. This model serves as a baseline, as it does not take into account models' validation accuracy or confidence when combining them. With this combination method, $P(i)$ is given by

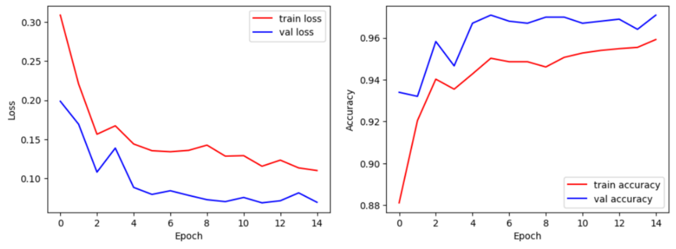

Fig. 7: InceptionNetV3 Training

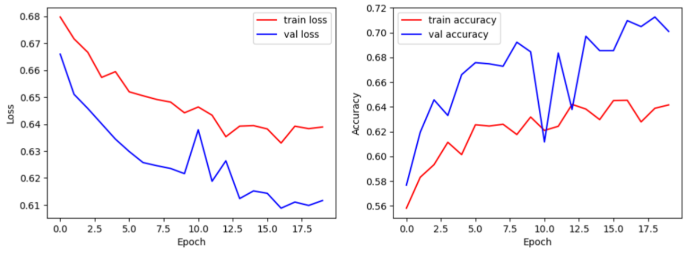

Fig. 8: EfficientNetV2 Training

$$P(i) = \begin{cases} 0 & \text{if } \sum_{m \in Models} round(\hat{y}_m) \leq \frac{|Models|}{2} \\ 1 & \text{Otherwise} \end{cases} \tag{1}$$

Note that when evaluating this metric on 2 models, if the models disagree, the calculation defaults to 0 (fresh).

3) Confidence Voting: To improve upon this naive approach, we weighed each network's prediction on its confidence. Confidence measures how certain a model is about the class it predicts. There are multiple ways that this can be calculated, but we decided to utilize the following formula:

$Conf = 4 * (\hat{y} - 0.5)^2$

This formula yields approximately 0 whenever $\hat{y}$ is near 0.5, as this indicates high uncertainty. Whenever $\hat{y}$ is close to 0 or 1, it outputs a real number around (but never above) 1, indicating high confidence. This formula grants significantly greater weight to predictions that are close to 0 or 1, as this indicates the model is very certain about its prediction. Meanwhile, uncertain models' predictions will be around 0.5, so they will have very little impact on the output, especially if other models produced more confident predictions. For shorthand, we will also define $G(\hat{y})$ as

$G(\hat{y}) = 2(round(\hat{y}_m) - 0.5)$

$G(\hat{y})$ simply outputs -1 if $\hat{y} < 0.5$ or 1 if $\hat{y} \geq 0.5$, indicating whether a prediction should raise or lower the total.

The overall prediction is given by the formula:

$$P(i) = \begin{cases} 0 & \text{if } \sum_{m \in Models} Conf(\hat{y}_m) * G(\hat{y}_m) < 0 \\ 1 & \text{Otherwise} \end{cases} \tag{2}$$

This formula gives greater weight to the models with higher confidence, while minimizing the influence of models with weak confidence. This should improve performance over the

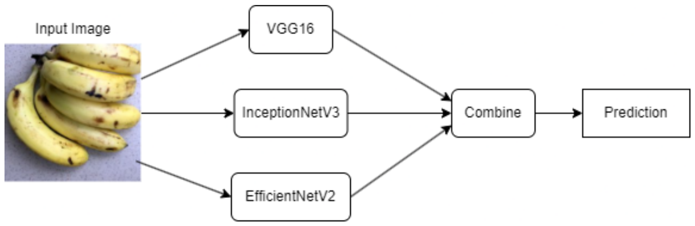

Fig. 9: Combined Model Structure

naive method by reducing the impact of models that are uncertain in their classification.

4) Competence Voting: Alternatively to using the models' confidence, we weighed each model's prediction based on its performance on the validation data set. This gives greater weight to the better performing models, while lessening the impact of models with worse performance. A model's competence is given by the formula $Comp(m) = V_m - 0.5$ Where V_m corresponds to model m's accuracy on the validation set. With a formula for measuring the competence of a model, the combined prediction is given by the formula

$$P(i) = \begin{cases} 0 & \text{if } \sum_{m \in Models} Comp(m) * G(\hat{y}_m) < 0 \\ 1 & \text{Otherwise} \end{cases} \tag{3}$$

This combination method prioritizes models that have shown their aptitude on the validation set, while still giving the lesser performing models a chance to weigh in if the best-performing models are uncertain.

5) Weighted Voting: Lastly, we combined the two previous methods for combining results to consider each model's confidence and performance when making the final prediction. This method takes into account the models' respective validation accuracies and confidence on the input sample. The final prediction is given by the formula

$$P(i) = \begin{cases} 0 & \text{if } \sum_{m \in Models} Comp(m) Conf(\hat{y}_m) G(\hat{y}_m) < 0 \\ 1 & \text{Otherwise} \end{cases} \tag{4}$$

IV. Results

A. Individual Model Results

Figure 12 showcases the individual models' performances on both the validation set used in training and the testing data set.

InceptionNetV3 achieved the highest independent testing accuracy, reaching 96.90% without the assistance of either other model. VGG16 performed a bit worse, only reaching 93.89%. EfficientNet performed significantly worse, only reaching 67.12% accuracy on the testing data set.

B. Combined Model Results

Figure 13 shows the performance of each possible combination of models with each combination method on the testing data set.

Out of every possible combination of model and method, the highest overall testing accuracy was achieved by InceptionNetV3 and VGG16 together using our novel weighted combination method. Other combination methods of these two models scored similarly well, with the worst only being as low as 96.02%. Including EfficientNet in the ensemble slightly decreased the combined model's performance, going from 97.77% to 97.67% for weighted voting. This decrease was fairly minimal as the competence metric applied to EfficientNet prevented it from being able to impact the results much.

Figure 10 lists the number of instances that each exact combination of models got correct, that is, each row corresponds with how many instances that only the model(s) got correct. For example, the row labeled "Efficient, VGG" indicates how many instances EfficientNetV2 and VGG correctly classified, but InceptionNetV3 failed to correctly classify.

Correct Classification Breakdown	
Combination	Total
Efficient, Inception, VGG	645
Efficient, Inception	24
Efficient, VGG	12
Inception, VGG	301
Efficient	2
Inception	31
VGG	6
No model	10

Fig. 10: Correct Classification Breakdown

Figure 11 showcases this data geometrically, with overlapping circles showcasing how many instances each model correctly classified. Region sizes are not to scale.

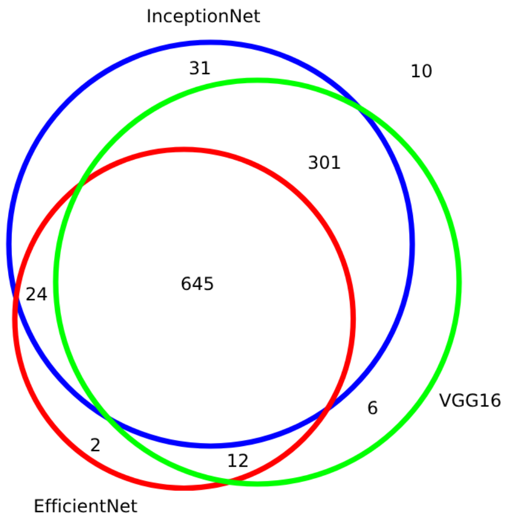

Fig. 11: Correct Classification Breakdown

Individual Model Metrics				
Model	Validation Loss	Validation Accuracy	Training Loss	Training Accuracy
EfficinetNet	0.6116	70.10%	0.6065	67.12%
InceptionNet	0.0691	97.09%	0.0617	96.90%
VGG	0.3340	90.39%	0.3043	93.79%

Fig. 12: Individual model validation and testing metrics

Combined Model Testing Accuracy					
Combination	Averaging	FPTP	Confidence	Competence	Weighted
Efficient, Inception, VGG	**97.67%**	95.25%	**97.58%**	95.25%	97.67%
Inception, VGG	97.28%	**96.02%**	97.67%	**97.09%**	**97.77%**
Efficient, VGG	91.95%	90.30%	91.85%	93.50%	93.02%
Efficient, Inception	97.58%	92.82%	97.48%	**97.09%**	97.38%

Fig. 13: Combined model testing accuracy

V. Discussion

The best-performing ensemble model was the weighted combination of InceptionNetV3 and VGG16, which factored in their respective validation accuracies and their confidence. Both of these models achieved over 90% testing accuracy individually, so the resulting combination reaching nearly 98% accuracy is not surprising.

EfficientNet performed significantly worse than the other two models individually. As a result, it tended to worsen performance when included in an ensemble model. However, competence voting and weighted voting both factor in models' validation accuracies when using models in predictions, so the impact was mostly negated.

In total, only 10 of the testing instances were incorrectly identified by all 3 models, meaning that 99.03% of instances were correctly identified by at least one model. This indicates that the networks are indeed "covering" for each other, as the majority of instances that any network got wrong were correctly identified by at least one other network. Ensuring that the correct opinion of each network are given the most weight proves challenging though, as even the weakest network, EfficientNetV2, still correctly identified 2 instances that both VGG16 and InceptionNetV3 got wrong. Finding a balance that allows the worst performing network to override the best-performing ones in rare edge cases like this is challenging. However, a bit more fine-tweaking of the parameters or combination formulas may have allowed us to reach over 98%.

We did not factor in running time when designing our model. As a result, when considering all three networks, our combined model has a total 56,849,345 parameters, or only 36,517,985 if you only include the overall best combination. This would likely struggle to run in real-time with any embedded hardware, especially since that any real-time application of this would need to examine several samples per second.

Even though our best combined model had nearly double the parameters of the best-performing solo model, the testing accuracy was not improved by even a full percent. This trade-off would likely not be worth it for use in real-time systems, but in applications where performance is irrelevant and accuracy is all that matters, this marginal improvement may be worth the increased running time. This heavily depends on the application, but for classifying fresh fruit on an industrial scale, this trade-off would not be acceptable.

A possible better approach may have consisted of training several light networks instead of a few large ones. Going from 21,802,784 parameters for InceptionNetV3 to 36,517,985 with the inclusion of VGG16 increased the testing accuracy by less than a percent, so it seems evident that adding significantly more parameters does not guarantee significantly better performance. Thus, it seems like even InceptionNet's 21,802,784 parameters may be excessive for the application, and smaller networks could have achieved similar accuracy. Smaller networks would have allowed us to train more of them, and combine the resulting networks in many more ways.

VI. Conclusion

For industrial food production, ensuring the quality of the product is a safety-critical application. Technology allows us to process vast quantities of foods, and preventing the usage of rotten or spoiled samples in production is vital. Humans are usually selected to perform these tasks, but that comes with its own drawbacks, such as costing a wage or getting distracted. Monotonous tasks such as this are a prime candidate for automation, but AI systems offer suffer their own drawbacks that makes using them in safety-critical applications questionable. Ensemble learning is one approach to reduce the possibility of undesired operation.

An ensemble model consists of multiple separate AI systems working in tandem to achieve a common goal. When trained properly, they can compensate for the shortcomings of any single model and achieve greater results than any individual model could. They are computationally more expensive than a single model, however, so one must be mindful of the intended use case of the combined model when designing an ensemble model.

In this paper we created an ensemble model for diving images of fruit into two categories: fresh and rotten. We utilized pretrained versions VGG16, InceptionNetV3, and EfficientNetV2 and added our own output layer to suit the models to the task. We combined their predictions in 5 distinct ways, including the common method of averaging, and 4 novel approaches.

The best performing solo model, InceptionNetV3, reached 96.90% accuracy on the testing data set. Out of all possible model sets with all possible combination methods, our best performing model achieved 97.77% accuracy using InceptionNetV3 and VGG16 with one of our novel combination approaches. Several other combinations reached over 97% accuracy, but it seems that the base accuracy of the models is a far more influential on a combined model's output than the specific combination method used.

Ensemble learning has its own advantages and drawbacks. While our best combined model achieved greater accuracy than the best individual model, the performance increase was fairly marginal, while being significantly more computationally complex. It is unlikely that our model would be used in a real food production facility. However, if simpler models were used, many more models could be trained and executed in parallel without being too taxing on the system. Over 99% of instances in the testing data set were correctly identified by at least one of our models, so the models ultimately did end up covering each others' weaknesses. Unfortunately, with only 3 models at our disposal, it's difficult to truly see this effect. This method of machine learning holds promise, however, it seems better suited for many smaller models rather than a few large models when real-time viability is a concern.

References

[1] Y. Chherawala, R. Lepage, and G. Doyon, "Food grading/sorting based on color appearance trough machine vision: the case of fresh cranberries," in *2006 2nd International Conference on Information & Communication Technologies*, vol. 1, 2006, pp. 1540–1545.

[2] Y. Li, X. Feng, Y. Liu, and X. Han, "Apple quality identification and classification by image processing based on convolutional neural networks," *Scientific Reports*, vol. 11, 08 2021.

[3] D. Pandit, N. Prakash, S. S. M, S. H. Dahale, and S. Tripathi, "Computer vision-based automated cashew kernel grading," *Proceedings of the 2023 5th International Conference on Image, Video and Signal Processing*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259177792

[4] L. Zhu and P. Spachos, "Support vector machine and yolo for a mobile food grading system," *Internet of Things*, vol. 13, p. 100359, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660521000032

[5] V. Hemamalini, S. Rajarajeswari, S. Nachiyappan, M. Sambath, T. Devi, B. K. Singh, and A. Raghuvanshi, "Food quality inspection and grading using efficient image segmentation and machine learning-based system," *Journal of Food Quality*, vol. 2022, no. 1, p. 5262294, 2022. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/5262294

[6] V. Meshram and K. Patil, "Fruitnet: Indian fruits image dataset with quality for machine learning applications," *Data in Brief*, vol. 40, p. 107686, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340921009616

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[8] V. P., "Vggnet-16 architecture: A complete guide," 2020, [Online], https://www.analyticsvidhya.com/blog/2021/12/traffic-signs-recognition-using-cnn-and-keras-in-python/.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2015. [Online]. Available: https://arxiv.org/abs/1512.00567

[10] S. AlTakrouri, N. Noor, N. Ahmad, T. Justinia, and S. Usman, "Image super-resolution using generative adversarial networks with efficientnetv2," *International Journal of Advanced Computer Science and Applications*, vol. 14, 01 2023.