ASSESSING THE QUALITY OF GRAIN VISUALLY

Rohit Pandey, Vanitha DSilva and Vanjuli Kulshrestha

MSDSP 498- Capstone Report

Northwestern University

Mar 6, 2023

Background and Objective

Manual inspection of the quality of pre-processed food grains is very time-consuming, cumbersome and prone to errors. This calls for the need to look for alternative techniques which can be deployed as part of smart farming or farm automation. Automating methods of assessing grain quality can be done for wholesale suppliers as truckloads of grains/pulses are entering the vicinity. It can also enable creation of online auctioning for grains/pulses hence expanding the buyer-seller space. This can also streamline the logistics involved and reduce waste. Visual assessment of quality can be done using apps on handheld devices and hence can also be used to recommend pricing with higher quality demanding higher prices and lower quality sold for animal consumption.

The objective of our project is to develop an end to end application for assessment of grain quality.

Motivation and Opportunity

Corn or maize is a major cereal crop, with global food consumption being the third highest after rice and wheat Global human food consumption of maize amounts to 18.5 kg/capita/year , 11% of the average annual cereal human consumption of 175 kg globally (2014–18, excluding beverages – FAOStat, 2021).

A similar trend is followed in India where it contributes to ~9% of the grain production. Production of maize has been growing at an average CAGR of at least ~2.5% since 2010. While factors such as adaptability to diverse agro-climatic conditions, lower labour costs and lowering of water table in the rice belt of India have contributed to the increase in acreage, the yield is still half of the

global average. Extreme changes in climate causing drought or flooding, excess spread of disease and insects, inadequate irrigation,lack of right agro-technology such as single-cross hybrid. Throughout the production process corn can be damaged through processing or transportation- this makes the ratio of pure to impure or damaged corn kernels vital to the determination of harvest or yield quality. All this and lack of resources available to farmers leads to higher wastage. And lower pricing rates for existing harvests. There is a dearth of efficient, consistent, accurate and economical estimated of the pure to impure. This paper is a contribution to provide the farmer with decisive information to accurately ascertain the quality of his harvest and charge a fair price through a real time digital application.

Goals

Therefore, the main goals of this project are as follows:

1. Effectively calculate the proportion of healthy corn kernels vs impure to provide an estimate for pricing through image processing with a minimum accuracy of 95%
2. Correctly detect images in the dataset, identifying methods of augmenting through zoom, flip, rotating where necessary.
3. Building out a pipeline for combining detection and classification (1 and 2)
4. Provide interpretation and insights that may be understandable, including the identification of important characteristics
5. Ensure the solution is light, concise, efficient enough to work on a handheld device to enhance the usability of the model
6. Complete the project deliverables for the agreed upon scope within time and budget

Literature Review

While there are many methods of image classification or grain counting, the most common methods have been that of extracting features and then classifying them. Traditional image segmentation largely focused on simplifying the image by limiting the pixels of an image (thresholding), altering the pixels to change the size, colour, shape, depth etc (filtering) or topologically interpreting image boundaries (watershedding). These were frequently applied on fresh produce such as apples, tomatoes etc. More nuanced feature extraction approaches in object detection methods used feature descriptors such as HOG (in a sliding window) and SIFT (matching local regions). They computed image gradients through orientation of image and magnitude, divided the image region into spatial bins, and created a histogram of the gradient magnitudes according to their orientations and spatial location. Then, a classifier was trained such as a support vector machine (SVM) or AdaBoost. For instance, Kuo *et al.* classified grains sparse-representation-based classification (SRC). Kambo and Yerpude distinguished the variety of grain using K-NN and Principal Component Analysis (PCA). Guo et al. employed a quadratic-SVM to detect and count sorghum from aerial images. In short, machine learning is an oft used method in object detection of various grains and produce. However these tend to be rigid and cannot handle low image resolutions, large variations in image scale and orientations.

Certainly, SIFT and HOG features are limited and unlike deep learning methods do not use hierarchical layer-wise representation learning. This is one of the reasons why CNNs are more flexible and can learn a function to model more abstract representations of data. This enables it to localize objects in a given image and determine which category each object belongs to. Methods such as single shot detection (SSD), you only look once (YOLO) and fast R-CNN extract nuanced feature descriptors with significant improvement in accuracy. When first introduced, this was at a

cost of longer training time, being data hungry and resource intensive, reducing feasibility for real time deployment in the field..

In 2019, Ghosal et al. leveraged a RetinaNet model with significant improvement in detection and counting work by Guo et al. Various other deep learning models have also been proposed in disease detection, quality assessment and detection and counting of various crops . Leveraging transfer learning, models can be pre-trained on such datasets and have their information transferred to detect similar objects without the need for long training times Ni et al. in 2018 and Li et al. in 2019 both utilized deep learning to count individual corn kernels While they were successful , it was less accurate at distinguishing overlapping kernels or corn on the cob where it was densely distributed, disallowing usage for decision making at harvest time. Roy et al, 2022 overcomes this problem by modifying the YOLOv4 algorithm to optimize it through a CSP1 and DenseNet block, however there are still limitations in feature extraction. Liu, Wang 2019 YOLOv3 network was modified to bring it to near real time. It focused on making the features more robust , however does not provide class level ratio.

Some other vision-based grain classification approaches proposed include a pattern recognition based technique to extract the shape, colour and geometry of corn kernels. The kernels are classified using a backward propagation neural network to identify the class to which they belong. Others such as Ekashwarii et al, 2022 involve identifying the variety of corn system based on color and texture features to identify the quality of corns. Combination methods have also been approached such Khaki etal 2020 who used a deep learning-based technique to identify defective types of corns by using segmentation to separate a group of touching corns, extracting colour and shape features and lastly using a maximum likelihood estimator to classify normal and defective

corns. Another similar approach, the authors Wang et al, 2022 use pre-trained deep learning CNN models to overcome the dual touching kernel problem.

None of these however, solve for empowering the farmer with decision making power based on the quality of his harvest. Due to the difficult nature of this problem and the demand for corn kernel count estimates in the market, we propose a pipelined deep learning approach to detect and count corn kernels and then classify them in a pure-impure ratio. The focus of this solution is to provide a mobile, light, efficient, concise pipeline that both detects and classifies the images.

Methodology

The proposed approach consists of two stages, firstly the elements in the given image are detected and then they are classified into some of the following categories: pure corn, impure corn (including rotten, broken and foreign objects). Both stages are based on deep learning approaches, the success of the pipeline is dependent on the quality of the training dataset through annotation, augmentation and labeling and the efficiency of the algorithm.

The YOLO (You Only Look Once) is a series of object detection models popular for achieving the right balance between accuracy and small model size. YOLOv8 was launched on January 10th, 2023 and compared to previous versions of YOLO is more suited for real time object detection. Its backbone network DarkNet-53 (53 deep layers) surpasses YOLOv7 in speed and accuracy. Yolov8 makes bounding box predictions by pixels through an anchor free detection head. As with previous versions, it shows a larger feature map which improves accuracy, reduces the amount of time it takes to train the model and can help to reduce overfitting. YOLOv8 also uses feature pyramid networks (different scales of feature maps), which helps to better recognize objects of different sizes, which improves its overall accuracy. Given YOLOv8n is the fastest and smallest

model available in YOLOv8 series, it was used as part of the pipeline. Post this, classification on a pre-trained model was performed and the number of pure versus impure kernels in the dataset was outputted. Different pre-trained models were tried. This section discusses the theoretical background of the approaches that have been considered for feature extraction. The approaches to feature extraction include ResNet50, VGG16, MobileNet, Xception, Inception, and EfficientNet.

ResNet50 approach was introduced in the 2015 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). This model is a residual learning framework that can alleviate the vanishing gradient problem of Deep Convolutional Neural Networks during deeper networks' training. Zahisham et al. (2016) proposed a ResNet50-based Deep Convolutional Neural Network (DCNN) for food recognition with ~40% accuracy.

The VGG-16 approach was introduced by Simonyan and Zisserman at the 2014 ILSVRC conference.. The VGG-16 is trained on the ImageNet dataset with over fifteen million high-resolution images and 22,000 image classes. A comparison of CNN tolerance to the intraclass variety in food recognition got an accuracy of 84.48%

Howard et al. proposed MobileNet, a low-latency, low-computation model for on-device and embedded applications. Its architecture is based on depthwise separable convolution that significantly reduces computation and model size while maintaining classification performance similar to that of large-scale models, such as Inception. Following that, the paper in [7] implemented FD-MobileNet-TF-YOLO as an embedded food recognizer.

Chollet introduced the Xception model, a modified depthwise separable convolution model based on the Inception model but it reduces the number of model parameters and makes more efficient

use of them, allowing for the learning of richer representations with fewer parameters. Yao et al. conducted a study on the classification of peach disease with validation accuracy of 92.23%.

The Inception architecture uses a sparse structure of a convolutional network with one-by-one convolution dimensions to reduce dimensionality. GoogLeNet is a deep-learning model that uses the Inception architecture Singla et al. demonstrated the feasibility of the Inception network for food category with an accuracy of 83.6% when tested against the Food-11 dataset.

Tan and Le proposed the EfficientNet (EFFNet) model, which utilizes a simple and effective compound coefficient to scale up CNN structurally. In comparison to conventional neural network approaches, EFFNet uses a fixed set of scaling coefficients to scale each dimension of depth, width, and resolution uniformly.

Approach

An illustration of the approach is detailed below

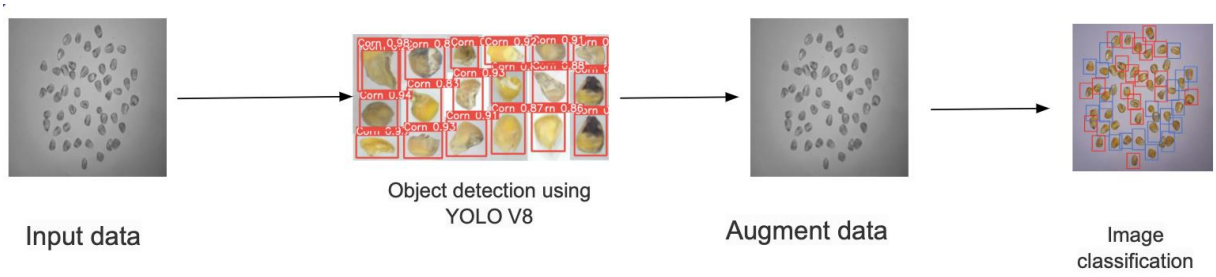


Fig1: Illustrative workflow of the approach used

The first step was to collect data from various sources to make the training dataset as rich as possible. The dataset was built up from multiple sources to ensure there were sufficient impure cases especially. The image below shows sample dataset of good and bad corn images

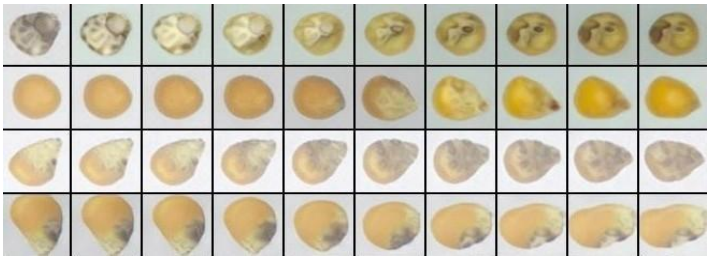


Fig2: Sample images of the corn kernels dataset

Next, various transformations were applied on the dataset to augment the data as well as to perform an initial analysis. A trial was conducted on an image of grain to visualize various processing outcomes:

Fig3:(From left to right) Various transformations applies Resize image of broken grain;Grayscale;Image denoising;Contours;Image Segmentation (Watershed Algorithm)

Further, YOLOv8n was applied to detect corn in the images. YOLOv8 tends to be data hungry when training so the training dataset was further enhanced with annotated datasets from Roboflow:

1. Data Source 1- has classes belonging to Fusarium rotation, Healthy rotation, Others rotation
2. Data Source 2 - has classes Broken, Damage, Healthy, Immature, Shriveled, Weeveled, Fm-inorganic, Fm-organic
3. Data Source 3 - has classes Broken, Damage, Fm-inorganic, Fm-organic, Fungus, Healthy,Immature, Shriveled, Weeveled
4. Data Source 4 - has classes Bad, Good

5. Data Source 5 - has classes Broken, Corpos Estranhos, Damage-Discolor, Danificado, Fungus, Healthy, Immature, Inorganic foreign material, Inteiro, Não Canjicado, Não Degerminado, Organic foreign material, Other Edible Seeds, Padrão, Película, Ponto Preto, Shriveled, Trincado, Weeveled, bad, broken, damage, fm-inorganic, fm-organic, fungus, fusarium rotation, good, healthy, healthyrotation, immature, shriveled, weeveled
6. Data Source 6 - has classes Chana-Desi, Inorganic Foreign Material, Inorganic foreign material, Maize-Yellow Maize, Organic Foreign Material, Organic foreign material, Rice-Basmati, Wheat, fm

Below shows bounding box annotation results to detect corn kernels. The results are shared in the table below:

Dataset	**Images**	**Instances**	**BoxP**	**R**	**mAP50**
Validation	256	8345	0.967	0.963	0.974
Test	82	2953	0.966	0.952	0.977

Table1: YOLOv8n object detection results

The images below show examples of correctly detected corn kernels of varying quality with the last images illustrating the bounding boxes.

Fig4:(From left to right) Images of corn fed to YOLOv8 and the resulting bounding boxes

Object detection helped to distinguish kernels that were close to or overlapping each other The resulting model outputs bounding boxes was leveraged in a pre-trained image classification model. One challenge with transfer learning was that most of these models have been used for human or transport detection and not specifically corn kernels as featured might not be easily transferred.. For this reason, multiple models were tried from a baseline to pre-trained to customized pre-trained models. The test results are given below.

				Impure				Pure			
Model Name	**Model category**	**Accuracy**	**Loss**	**Precision**	**Recall**	**F1 score**	**Support**	**Precision**	**Recall**	**F1 score**	**Support**
EfficientNet	Pretrain	0.96803	0.832	0.96	0.98	0.97	1475	0.98	0.96	0.97	1500
EfficientNet	No-pretrain	0.968796	0.832	0.96	0.97	0.96	1475	0.97	0.96	0.96	1500
MobileNet	Pretrain	0.96623	0.090549	0.96	0.98	0.97	1475	0.98	0.96	0.97	1500
MobileNet	No-pretrain	0.97605	0.705	0.97	0.97	0.97	1475	0.97	0.97	0.97	1500
VGG16	Pretrain	0.601	0.5022	0	0	0	1475	0.5	1	0.67	1500
VGG16	No-pretrain	0.94487	0.1489	0.97	0.89	0.93	1475	0.9	0.97	0.94	1500
ResNet50	Pretrain	0.86996	0.346394	0.92	0.8	0.85	1475	0.83	0.93	0.87	1500
ResNet50	No-pretrain	0.920195	0.219788	0.98	0.86	0.92	1475	0.88	0.98	0.93	1500
InceptionNet	Pretrain	0.960685	0.962901	0.96	0.97	0.96	1475	0.97	0.96	0.96	1500
InceptionNet	No-pretrain	0.970094	0.080148	0.97	0.98	0.97	1475	0.98	0.97	0.97	1500
VGG19	Pretrain	0.92994	0.187585	0.93	0.93	0.93	1475	0.93	0.93	0.93	1500
VGG19	No-pretrain	0.5042	0.693112	0	0	0	1475	0.5	1	0.67	1500

Exception Net	Pretrain	0.963542	0.891802	0.98	0.95	0.96	1475	0.98	0.96	0.97	1500
Exception Net	No-pretrain	0.96875	0.947865	0.98	0.96	0.97	1475	0.96	0.98	0.97	1500

Table2:Model results on test dataset

MobileNet, InceptionNet and EfficientNet are among the most accurate models with a F1 score of 97% each for testing dataset. The validation and test ROC curves and the confusion matrices for the same are illustrated below. It had an AUC of 0.99 for test with majority classes identified accurately.

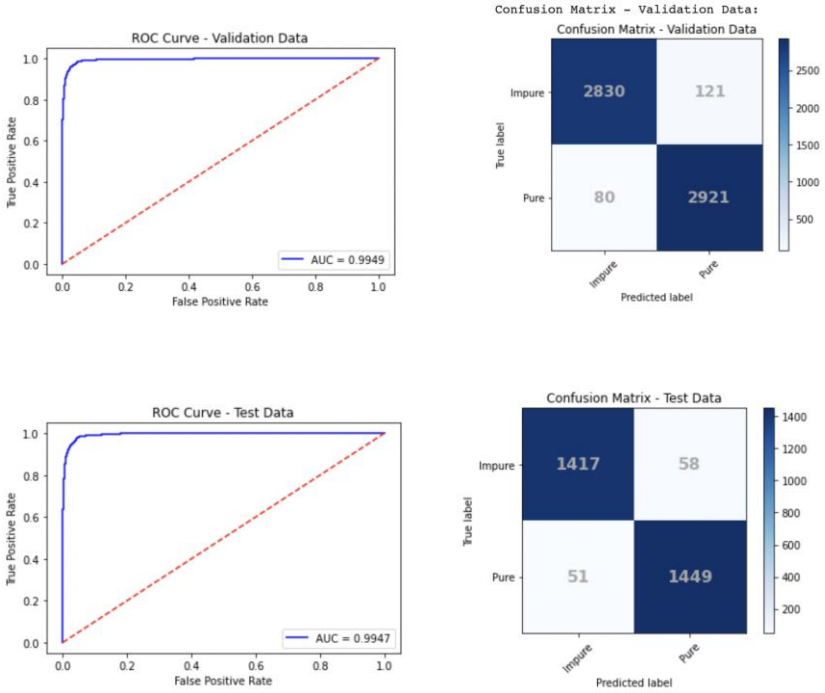


Fig5:(From top to bottom) Validation ROC curve and confusion matrix; Test ROC curve and confusion matrix

However, the time taken and size of the model are also important factors. The chart below shows time taken against the number of epochs. The size of each bubble is representative of the model

size. A lighter model would be easier to deploy. Due to all these reasons we can conclude that the best model would be MobileNet.

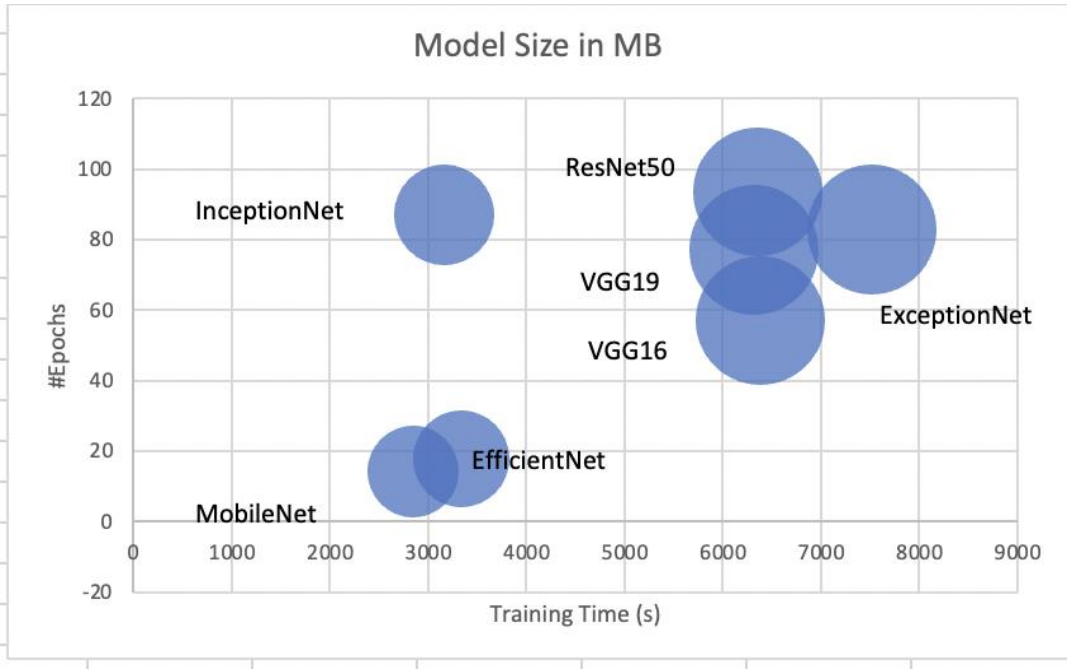


Fig6:Bubble chart depicting the training time and number of epochs for each pre-trained model. The size of the bubble shows the size of model

Conclusion

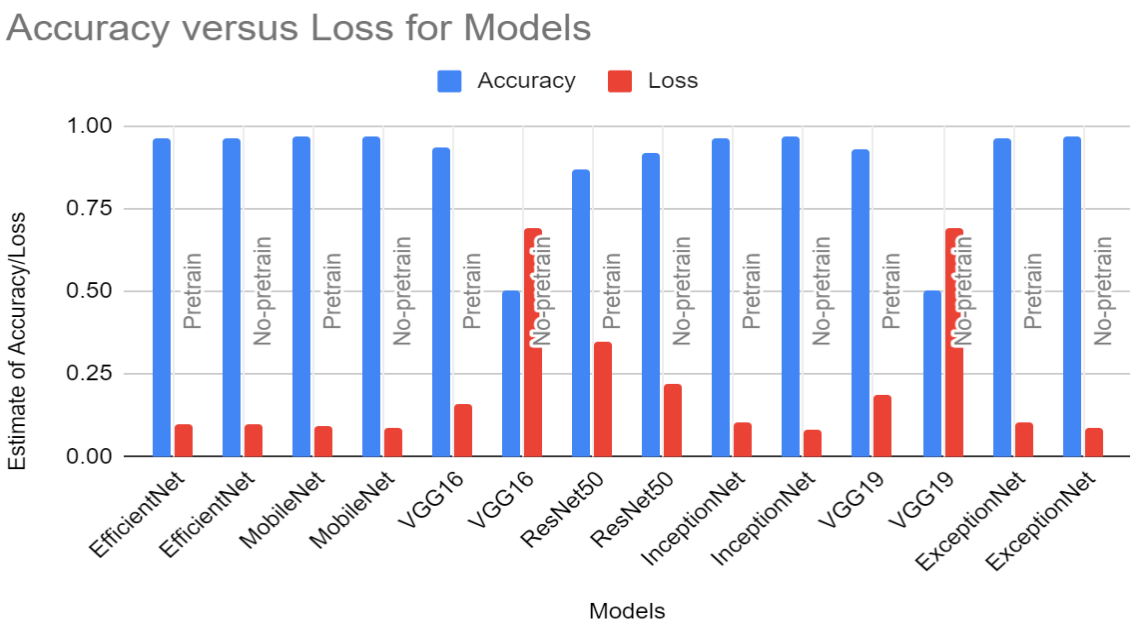
In this paper, we proposed a kernel quality evaluation through detection, classification and counting method based on deep learning. YOLOv8n was used first to create bounding boxes that would help with the classification, It is the lightest version of the YOLO series of models and was selected keeping in mind its speed (a challenge with the other versions of YOLO models), the classification model subsequently used was able to bolster minor reduction in accuracy at the cost of speed. Due to the effectiveness of the YOLO + pretrained MobileNet CNN classifier, this approach does not make any assumptions on the lighting conditions, orientation of kernel or the

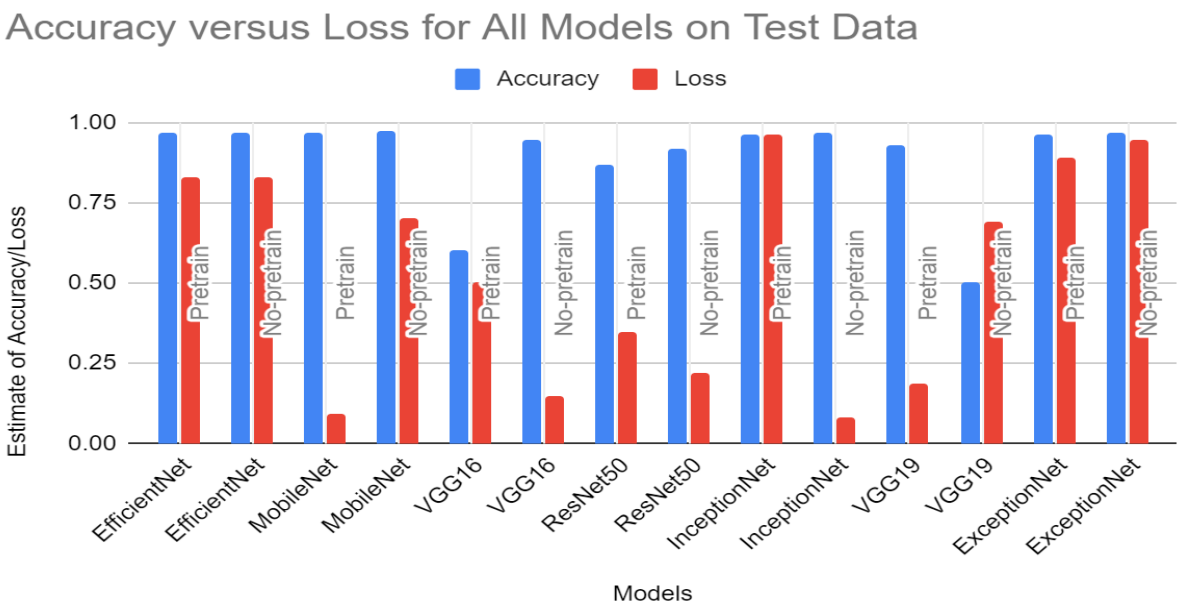
background quality or the number of kernels. Removing these limitations allows farmers and agronomists to use this in-field to estimate the quality of harvest, giving them additional decision making power when it comes to their crop. A limitation in the approach was when multiple kernels were overlapping and perhaps enhancing the YOLOv8n with a sliding window segmentation would help. This approach could be extended for disease detection and quality assessment of corn with additional data sources..

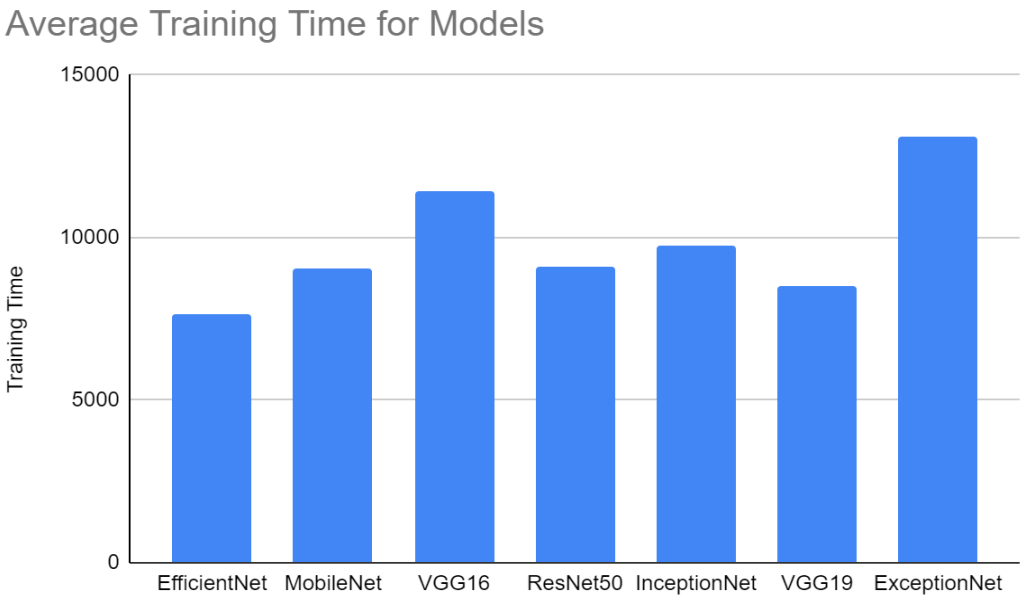
Appendix

Results

Classification Model Validation results







S. No.	Model Name	Type	Validation	Class 0				Class 1			
		Accuracy	Loss	Precision	Recall	F1 score	Support	Precision	Recall	F1 score	Support
Efficient	Pretrain	0.96203	0.09623	0.96	0.97	0.96	2951	0.97	0.96	0.96	3001

Net											
EfficientNet	No-pretrain	0.96203	0.09623	0.96	0.97	0.96	2951	0.97	0.96	0.96	3001
MobileNet	Pretrain	0.96623	0.090549	0.97	0.96	0.97	2951	0.96	0.97	0.97	3001
MobileNet	No-pretrain	0.96959	0.088444	0.97	0.97	0.97	2951	0.97	0.97	0.97	3001
VGG16	Pretrain	0.932628	0.158131	0.97	0.89	0.93	2951	0.9	0.97	0.94	3001
VGG16	No-pretrain	0.5042	0.69311	0	0	0	2951	0.5	1	0.67	3001
ResNet50	Pretrain	0.86996	0.346394	0.92	0.8	0.86	2951	0.83	0.94	0.88	3001
ResNet50	No-pretrain	0.920195	0.219788	0.98	0.86	0.91	2951	0.87	0.98	0.93	3001
InceptionNet	Pretrain	0.960685	0.102901	0.96	0.97	0.96	2951	0.97	0.96	0.96	3001
InceptionNet	No-pretrain	0.970094	0.082148	0.97	0.97	0.97	2951	0.97	0.97	0.97	3001
VGG19	Pretrain	0.92994	0.187585	0.93	0.93	0.93	2951	0.93	0.93	0.93	3001
VGG19	No-pretrain	0.5042	0.693112	0	0	0	2951	0.5	1	0.67	3001
ExceptionNet	Pretrain	0.963542	0.101802	0.98	0.95	0.96	2951	0.95	0.98	0.96	3001
ExceptionNet	No-pretrain	0.96875	0.089786	0.98	0.96	0.97	2951	0.96	0.98	0.97	3001

Classification Model Training times

Model Name		Standard LR	Standard Epochs	Training Time	Model Size in MB	#Epochs
EfficientNet	Pretrain	0.001	20	3339	17.9	11
EfficientNet	No-pretrain	0.001	20	3662	49	13
MobileNet	Pretrain	0.001	20	2841	14.1	10
MobileNet	No-pretrain	0.001	20	6231	38.7	20
VGG16	Pretrain	0.001	20	6377	57	20
VGG16	No-pretrain	0.001	20	5051	2.8	14
ResNet50	Pretrain	0.001	20	6351	93.5	20

ResNet50	No-pretrain	0.001	20	2765	273.4	8
InceptionNet	Pretrain	0.001	20	3168	87	12
InceptionNet	No-pretrain	0.001	20	6594	242	13
VGG19	Pretrain	0.001	20	6314	77.2	20
VGG19	No-pretrain	0.001	20	2195	230.1	6
ExceptionNet	Pretrain	0.001	20	7520	83	20
ExceptionNet	No-pretrain	0.001	20	5558	242	16

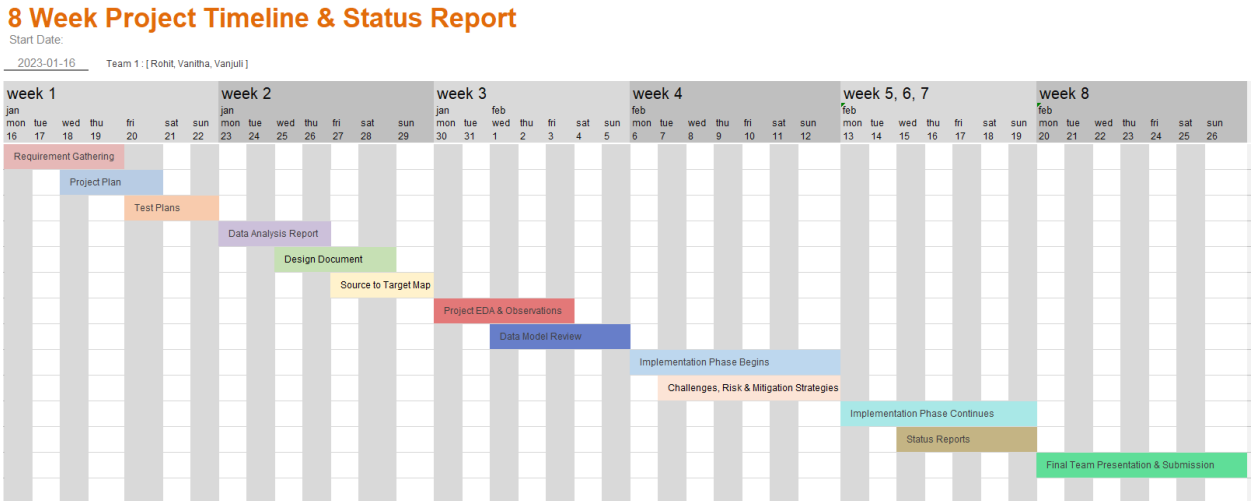
Confusion Matrix test, validation

		Validation Dataset					Test Dataset				
Type	Validation	Impure - True	Impure - False	Pure - True	Pure - False	Accuracy	Impure - True	Impure - False	Pure - True	Pure - False	Accuracy
Efficient Net	Pretrain	2850	101	2876	125	96%	1441	34	1441	59	97%
Efficient Net	No-pretrain	2850	101	2876	125	96%	1441	34	1441	59	97%
MobileNet	No - Pretrain	2851	100	2920	81	97%	1436	32	1468	39	98%
MobileNet	Pretrain	2830	80	2921	121	97%	1417	51	1449	58	96%
VGG16	No-Pretrain	0	0	2951	3001	50%	0	0	1500	1475	50%
VGG16	Pretrain	2631	81	2920	320	93%	1351	40	1460	124	94%
ResNet50	No-Pretrain	2529	53	2948	422	92%	1273	25	1475	202	92%
ResNet50	Pretrain	2372	195	2806	579	87%	1182	108	1392	293	87%
VGG19	No-Pretrain	0	0	2951	3001	50%	0	0	1475	1500	50%
VGG19	Pretrain	2756	222	2779	195	93%	1373	113	1387	102	93%

ExceptionNet	Pretrain	2793	59	2942	158	96%	1414	33	1467	61	97%
ExceptionNet	No-pretrain	2825	60	1941	126	96%	1420	25	1475	55	97%
InceptionNet	Pretrain	0	0	2951	3001	50%	0	0	1475	1500	50%
InceptionNet	No-pretrain	2756	222	2779	195	93%	1373	102	1387	113	93%

Timeline

The below is the proposed project timeline to achieve the week wise objectives described for the upcoming 8 weeks. Every week has the respective weekly deliverables like Requirement and Project Plan, Test Plan, Design Document etc.



Unit Testing

mputer Vision Project Team 1	**Name:**	**ASSESSMENT**			
	Module Name:		**Test Designed date:**	**23-01-2023**	
	Release Version:		**Test Executed by:**	27/02/2023	
			Test Execution date:		
Pre-condition	Availability of the quality dataset of the grains				
Dependencies	Dataset and compute dependencies				
Test Priority	Test priority is set as per the execution level of the Test Case in sequential order i.e. pre-training, post-training, model component and integration test followed by model evaluation test				

Test Case#	**Test Title**	**Test Summary**	**Test Steps**	**Expected Result**	**Actual Result**	**Status**	**Notes**
1	Pre-training Test	Series of tests before the actual model is trained	1. Check whether model output shape is proper or not which should be with respect to number of classes of the grain 2. Check for no missing labels in training or test datasets for all defined categories of the grain 3. Ensure no Test and Validation Data Leakage 4. Check to ensure final prediction is within expected range of outputs	Model should pass all test steps under Test Case - "Pre-training Test"	14/02 - validation accuracy higher	Pass	18/02- shuffle step introduced

			5. Make assertions about the data					
2	Labelling Test	Series of tests conducted after the model is trained	1. Test whether labeling is as expected 2. Test for consistency i.e. cases of clear defects, cases of unclear defects 3. Test for balance within the dataset	Model should pass all test steps under Test Case - "Labeling Test"	23/1- edge cases found. Additional training data added	Pass	23/1 - resolved	
3	Post-training Test	Series of tests conducted after the model is trained	1. Test whether model prediction is as expected 2. Test for robustness of model i.e. underlying biaseness, noise impact on the model 3. Test for reproducibility with fixed random seed and version control 4. Invariance test to see if the trained model performs correctly, the slight changes in the input should not affect the model predictions.	Model should pass all test steps under Test Case - "Post-training Test"		Pass		
4	Model Component & Integration Test	Series of test conducted for unit, regression and integration testing of the DL Model	1. Unit test or Minimal function test- Check the correctness of individual model components 2. Regression test - Check whether your model breaks and test for previously encountered bugs 3. Integration test - Check whether the different components work with each other within your deep learning pipeline 4. Comparison test -Check different algorithms	Model should pass all test steps under Test Case - "Model Component Test"	08/02- Pipeline not functional	Pass	20/02- resolved	

			performance through the tests detailed above to conclude on the best method					
5	Model Evaluation Test	Series of test on trained model to evaluate it's working	1. Evaluate model based on the test or evaluation criteria or function 2. Test model on validation datasets for performance metrics of the model like accuracy, precision both of which should be above 95%. 3. Test model using creation of plots which summarize performance on validation or test sets accurately	Model should pass all test steps under Test Case - "Model Evaluation Test"		Pass		
						Pass		

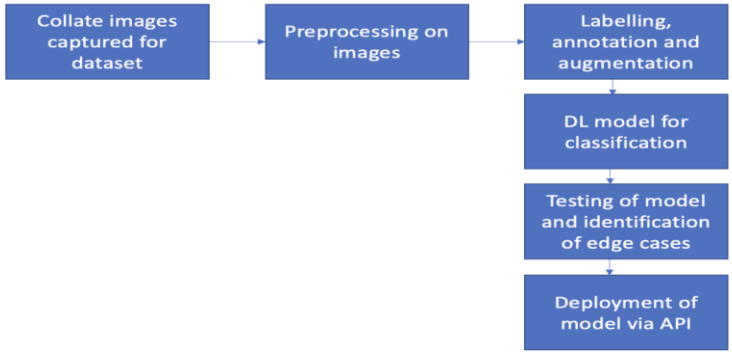
Risk Matrix

Risk and Issues	Description	Status	Solution
Integration risk and Deployment	Tflite and porting on app		Created initial tflite but still a major risk given time and automating pipeline challenges. Will keep a plan B in mind
Pipeline	Creation of proper pipeline connecting the object detection model to CNN model which gives the final output of pure vs impure grains		Research, multiple iterations and reaching out to SMEs for help
Model Accuracy	Accuracy of the model - slight differences in background, shape can impact the classification		Included images with variation in background, arrangement and different number of grains to increase richness of training dataset
Number of classes	YOLO was used for object detection which became more inaccurate with larger number of classes		Reduced number of classes to allow for YOLO to be more accurate given dearth of time
Object Detection	Identifying different classes correctly		Approached with different methods in an agile fashion, simultaneously researching solutions
Data	Identifying the right dataset		Agile ,Iterative approach

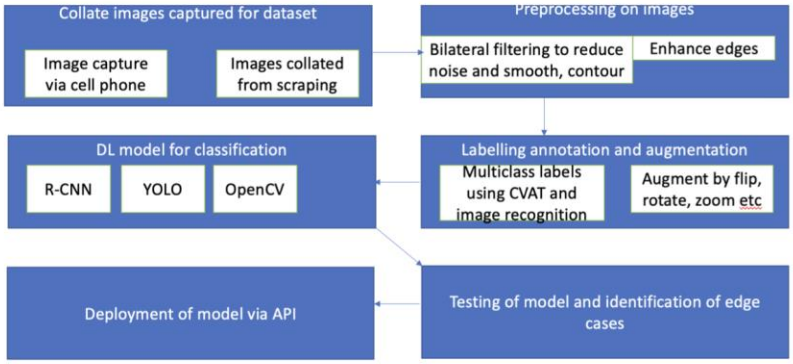
Process flow

We have the below process diagram for the target system with the goal of assessment of grain quality using Computer vision and Deep Learning model. This diagram depicts the activities of model building from data collection, data processing, data labeling, data augmentation to classification, testing and deployment.

UML Diagram of the Grain Quality Assessment System



The high level UML diagram of the proposed system is shown below



Modeling Activities for Grain Quality Assessment System