



LwSANet: Light Weight Self-Attention Network Model to Recognize Fruits from Images

Gracia Nissi Sathyadhas^{1*}, Angelin Gladston¹, Khanna H. Nehemiah²

¹ Department of Computer Science and Engineering, Anna University, Chennai 60025, India

² Ramanujan Computing Centre, Anna University, Chennai 60025, India

Corresponding Author Email: gracianissi@gmail.com

Copyright: ©2025 The authors. This article is published by IETA and is licensed under the CC BY 4.0 license (<http://creativecommons.org/licenses/by/4.0/>).

<https://doi.org/10.18280/ts.420117>

ABSTRACT

Received: 13 May 2024

Revised: 2 August 2024

Accepted: 15 October 2024

Available online: 28 February 2025

Keywords:

fruit recognition, deep learning, Self-Attention Network, batch normalization

Recognition and counting of fruits play a vital role in harvest estimation, harvesting, categorizing good and bad fruits, cost estimation, and stock estimation in departmental stores. Nowadays deep learning algorithms play a major role in automatic object detection. Automating such mechanical robots faces challenges due to less accurate predictions because of background foliage, illuminations, and nightmares. In this computer vision task to detect and classify the intended objects, we designed and developed a Lightweight Self Attention Network (LwSANet) model. To reduce the amount of processing, and increase the object detection speed and performance, the Self-Attention Network Block was also introduced. LwSANet is simple to adopt and has obtained an accuracy of 99.25% and a loss of 0.003% for single fruit detection and classification. It has obtained an accuracy of 98.2% and a loss of 0.23% for the detection and classification of fruits from multiple and overlapped fruit images. When we compare with other state-of-the-art models the achieved accuracy is 1.68% better than other models. Further, the model performance is compared with various well-structured state-of-the-art architectures like LeNet, VGG-16, GoogLeNet, MobileNet, SqueezeNet, and ShuffleNet.

1. INTRODUCTION

Fruits are rich in nutrients, essential for a healthy body, maintain blood sugar level and reduces the number of calories and fats consumed. Day by day, the demand for fruits are increasing which leads to automation in food industries [1]. The rapid growth in the field of image processing and neural networks helps to design and develop the automated robot to do the fruit detection, estimation, counting, yield estimation, harvesting, harvest estimation, pre-grading, quality estimation by addressing challenges like occlusions, variations in poses, intra-class variations, lighting conditions and resource constraints etc. Convolutional Neural Network (CNN) is the base for Deep Learning which works well in image recognition. The general structure of a Deep learning model contains input layer, CNN layers are for automatic best feature extraction and output dense layer for classification. The output layer filter count will be equal to the number of classes to be classified. Each CNN Layer designed with neurons which processes the features extracted from the previous layers. Going deeper, the model extracts more features. Based on the features to be extracted, the number of hidden layers can also be implemented. Normalization and Pooling Layers also helps to improve the performance of recognition of object from images.

Currently, the object detection deep learning algorithms divided into two categories. Single stage object detection and two stage object detection. Two stage object detection algorithms first find the regions and then classifies the objects. One stage object detection algorithm extract feature from the

network and predicts the class directly.

When compared with machine learning models, deep leaning models works well for both supervised and unsupervised learning environments and provides better accuracy and reduces the training time [1]. Color image classification suffers due to occlusions, illuminations and poor visibility [2]. Using machine learning model to recognize object from an image accurately, the environment should be structured one [3]. Increasing demand in deploying the deep learning models in resource constraint devices like mobile and IoT devices, light weight models are encouraged. There are many successful pre-trained deep learning network models to detect and classify the fruits from images such as LeNet, VGG, and GoogLeNet. Even though many number of successful pretrained models proposed earlier, the increasing need of running high quality deep neural network in limited computational environment or resource constraint devices motivated to propose Light weight Self Attention Network to address the challenges of processing the low-quality images thought resource constraint devices of modern computing environments.

In this work, a complete analysis has been carried out on the LeNet, VGG-16, GoogLeNet, MobileNet, SqueezeNet and ShuffleNet with two publicly available datasets, the Fruits-360 dataset with 131 classes of fruits, grains, nuts, vegetables, and Fruits-and-Vegetables-for-Image-Recognition dataset with 36 classes of fruits and vegetables. Then a Light weight Self-Attention Network (LwSANet) model is proposed with data generation, preprocessing, image augmentation, feature

extraction and classification. Further analyzed the LwSANet with other models in detail, and the prediction accuracy achieved 96.10% accuracy.

The main contributions of this work is as follows:

1). A simplified vision like CNN model to recognize fruits from images were developed with feature extraction block and classification block.

2). The feature extraction block implemented with Convolution-Batch Normalization-Activation block (CBA), Self-Attention Block to form attention on channels and CBA with MaxPooling. It is used to achieve faster, accurate and stable training. Instead of dropout, Reshape is used to compress the global spatial information generated by CBA.

3). Instead of Flattening and Dropout, Global Average Pooling Layer is used, which maintains the dimensions of the previous layer. To minimise the size of the activations without affecting the performance of the network, it takes into account the average value in each feature map. Also it acts like channel descriptor.

4). Classification block implemented with Reshape, Fully Connected Layer-Batch Normalization-Activation (FBA) blocks, and Dropout. FBA consists of Dense layer, BatchNormalizaion and Activation Relu [4].

The rest of the paper is organized as follows: Literature survey section discusses in detail the recent works carried out related to fruit recognition. System design section elaborates the proposed LwSANet: Light weight Self Attention Network model to recognize fruits from images. Experimental design throws light on implementation setup, experimental results and results and discussion section discusses in detail the inferences drawn out of the various experimental results obtained.

2. LITERATURE SURVEY

The basic LeNet comprises of eight layers including the Input and output layers. Input layer processes 32x32 pixel size image that is the max size object detected from its database [5]. Then the inputs were normalized as -0.1 for background and 1.175 for ground truth object to be recognized. Second layer is a Convolution layer with six feature maps sizing 28x28 which prevents network from dropping the boundary data it contains 156 trainable parameters. Third layer is subbing sampling layer with six feature maps sizing 14x14. 2x2 non overlapping receptive fields were used in subsampling which reduces the input 28x28 into 14x14. Fourth layers are Convolution layer which doesn't connects all the subsampling layer connections to each neuron, because different set of input connections helps in extracting different features sets. The organization of feature extraction connections in this fourth Convolution layer designed as follows. First six feature maps extract inputs from every contiguous subset of three feature maps, next six takes every contiguous set of four and the third final one takes inputs of all subsampling layer inputs, cumulatively this layer has 1516 trainable parameters. Fifth layer is a Subsampling layer with 16 feature maps of size 6x6 and 32 trainable parameters. Sixth layer is a Convolution layer with 120 feature maps and each unit connected with 5x5 neighborhood of all 16 features maps of fifth subsampling layer. Seventh layer is a fully connected layer with 84 units connected with all the Sixth Convolution layer units. Eighth layer is the output layer of 10 units fully connected layers with 10164 trainable parameters. A simple loss function is used with the above network as

estimation criterion for training samples represented as in Eq. (1):

$$E(Ts) = \frac{1}{S} \sum_{s=1}^S Y_o(Xi, Ts) \quad (1)$$

The penalties of in correct classification due to more discriminative criterion is handles using the following Eq. (2):

$$E(Ts) = \frac{1}{S} + \sum_{s=1}^S (Y_o(Xi, Ts) + \log(e^{-m} + \sum_n e^{-Y_n(Xi, Ts)})) \quad (2)$$

The basicVGG-16 based model consists of 13 convolutional layers into five blocks [6]. The blocks are interconnected sequentially with its next layer along with pooling layer with 2x2 pixel window. Finally, three fully connected layers, first two layers have 4096 channels and last fully connected layer have channels equal to number of classes to be classified. The hidden layers used ReLU as an activation function. The input image size fixed as 224x224. The model also trained and tested with single scale image and multi-scale images. The multi scale images sampled between the range 256 and 512 and made it as 384 as fixed. For insufficient training image sets, data augmentation techniques used to increase the number of images to avoid overfitting. Dropout ratio set to 0.5 and model trained for 74 epochs. During testing it achieved 6.8% of error rate [7]. Automatic Fruit classification VGG-16 model trained with two datasets, first 18 classes of 1653 images and second with 15 classes of 2633 images. Due to a smaller number of training images, second dataset accuracy of prediction is low.

The basic GoogLeNet incorporates the structures of LeNet and AlexNet [8]. This model initially built with 22 layers along with the design of group convolutions. The group convolutions called as Inception module. Each inception module has three 1x1, 3x3 and 5x5 convolution layers and one parallel Max Pooling layer, which increases the number of output from level to level. Then dimensionality reduction convolutions were added before each 3x3 and 5x5. Two auxiliary layers were added to increase the gradient signal to propagate back and to provide additional regularization.

Yolo is another model with many numbers of versions for object detection [9]. The basic first version model itself contains 26 convolutional layers and 2 fully connected layers. The version 2 contains 30 layers and version 3 is with 106 layers. Even though the v2 have 30 layers, the prediction accuracy is still bad when the object is little bit small. In this one multiclass problem turned into multi label problem. With CPU the training takes nearly 3000ms. Version from 4 handles high resolution images for training and testing. The model can be trained with 200 like less number of images for 2000 iterations and batch size 21. The object detection accuracy in natural environment is high in Yolo when compared with other Models like Alexnet, Resnet101 [10]. The parameters of Proposed and predefined models considered for our experiment were shown in Table 1.

The trainable parameters are defined and optimized automatically by the model. The non-trainable parameters are hyper parameters, which are optimized manually and are not optimized according to its gradient. The various Fruit Detection applications, technologies used and their Solutions were listed in Table 2.

Table 1. Comparison of predefined networks parameters with proposed model

| Network | Layers | Trainable Parameters Data Set 1 | Trainable Parameters Data Set 2 | Non-Trainable Parameters | Duration |
|-------------|--|------------------------------------|------------------------------------|-----------------------------|----------|
| LeNet | Conv2D-3 Sampling-2 Dense-2 | 0.99M | 0.9M | 0 | ~90s |
| VGG-16 | Conv2D-13 Dense-3 | 14.7M | 14.7M | 0 | ~140s |
| GoogLeNet | Conv2D-15 (Except DI Block) MaxPooling-5 AvePooling-3 Dense-6 | 7.5M | 7.5M | 512 | ~290s |
| MobileNet | Fully Connected-2 Dropout-2 Conv-14 Conv dw-13 BN-27 | 4.2M | 4.2M | - | ~102s |
| SqueezeNet | Conv-101 MaxPooling-3 GlobalAvgPooling-1 | 1.25M | 1.25M | - | ~148s |
| Shuffle Net | Bottleneck-4 Group Conv-8 ShuffleNet Unit-4 (203 Layers) | 1.42M | - | 2088 | ~107 |
| LwSANet | Conv2D-6 Dense-7 | 0.25M | 0.25M | 1052 | ~100s |

Fruits-360 and Fruits-and-Vegetables-Image-Recognition datasets were used for all the model analysis.
All the models executed for 20 epochs and 50 epochs.

Table 2. Summary of literature review

| S. No. | Problem | Techniques and Data Set Used | Solution / Review |
|--------|---|--|--|
| 1 | Automatic Fruit Classification Using Deep Learning for Industrial Applications [7]. | Fine-tuned visual geometry group-16. Dataset 1-Consists of 18 classes of clear fruit images with 1653 color images. Dataset 2-supermarket fruit dataset, which contains 15 classes of 2633 images. Used Swish and Drop block. | Even though trained for 100 epochs, for second dataset the accuracy achieved is 88.35, because of low number of training images. |
| 2 | Circular Fruit and Vegetable Classification Based on Optimized GoogLeNet [8]. | Data set-6 classes of 6600, each class 1100, test set is 100/class. Compared with 5 Models and produced better performance with Optimized GoogLeNet. | Training accuracy of GoogLeNet as 96.88% the testing accuracy as 96%. |
| 3 | Real-Time Detection of Ripe Oil Palm Fresh Fruit Bunch Based on YOLOv4 [9]. | Data set-Own 240 Positive Images and 250 Negative Images. | Accuracy is 86% @1000 Iterations and 100% @ 2000 Iterations. Yolo tiny V4 takes less than 2 Hrs. |
| 4 | Real-Time Monitoring Method of Strawberry Fruit Growth State Based on YOLO Improved Model [11]. | CSP block in the YoloX network replaced with a self-designed feature extraction module C3HB block. Normalized Attention Module attached to improve detection accuracy. Own 5600 Strawberry images were used. | Accuracy-94.26% |
| 5 | A Single Stream Modified MobileNet V2 and Whale controlled Entropy based Optimization Framework for Citrus Fruit Diseases Recognition [12]. | MobileNet-V2 CNN model finetuned and the model trained using TL. Approximately 1000 Citrus Fruit Images were used. | Accuracy-99.7% |
| 6 | Identification and Depth Localization of Clustered Pod Pepper Based on Improved Faster R-CNN [13]. | Improved Faster-RCNN. ResNet as Backbone. Dataset-Own 328 augmented as 3062. Horizontal & Vertical Comparison using Faster R-CNN and YOLOv3 network. | Accuracy-87.30% Very limited number of images were used. |
| 7 | Multi-Task Cascaded Convolutional Networks Based Intelligent Fruit Detection for Designing Automated Robot [14]. | Image Fusion technique is used to improve the detection. Dataset-1800 own images, 316 from the internet and 511 from ImageNet. Using image augmentation, dataset size increased. | Images Labelled manually. With few other class images, the model can be trained easily. |

Table 3. Summary of state-of-the-art models

| Methods | Advantage | Drawback |
|--------------------|---|--|
| <i>LeNet</i> | Very small network | Accuracy is low |
| <i>VGG-16</i> | Small network | Need to train for many epochs |
| <i>GoogLeNet</i> | Better accuracy | Network depth is too high. Requires more time for training |
| <i>MobileNet</i> | Small network, Less computation | MFLOPs and parameters are too high |
| <i>SqueezeNet</i> | Good in accuracy with small dataset | Not suitable for large data set |
| <i>Shuffle Net</i> | Compact for resource constraint devices | Doesn't supports optimizing hyperparameters |
| <i>LwSANet</i> | Compatible for small and large devices. Accuracy is high. Supports optimizing hyperparameters for different dataset. No constraint for image resolution. Light weight model and less computation. | |

State-of-the-art Model Squeeze and Excitation network models mainly concentrate on accuracy. Because of its SEBlock, it needs additional computational cost and resources. It achieved good performance with benchmarked datasets because of its channel-wise feature responses and attention mechanism. MobileNet employs depth-wise separable convolutions to reduce computational complexity significantly while maintaining reasonable accuracy. It can be deployed in mobile and embedded devices because it is so compact than VGG and GoogLeNet and aims to be more suitable in mobile or embedded environment deployment with limited resources. LwSANet prefers attention mechanisms to maintain accuracy with a very less number of layers in the model compared with LeNet, VGG, and GoogLeNet, also the compressed network can be deployed in Mobile or embedded environments. A deep neural network model with less number of layers and an attention mechanism decreases computational cost and time. This motivates to design a Lightweight Self Attention Network Model. Also, the proposed Lightweight model is compared with other lightweight models' state-of-the-art architectures MobileNet, SqueezeNet, and Shuffle-Net. Details are listed in Table 3.

3. PROPOSED METHODOLOGY

3.1 Data acquisition and preprocessing

Two datasets were used for our model training and testing. The first dataset is a Fruits-360 dataset. It contains 90483 single-fruit images of 131 classes. Fruits-360 Dataset [15] was proposed on focusing high-quality datasets to resolve the modeling objects, human-robot interaction, and autonomous robots for harvesting and fruit estimation, created a dataset with 131 classes of images in white plain background. Preprocessing involved resizing all snapshots to a uniform resolution appropriate for the network entry, accompanied by normalization to standardize pixel values throughout the dataset. Data augmentation techniques which include random rotations, flips, and zooms were implemented to artificially extend the dataset size and enhance the model's robustness to versions in fruit presentation. Most of the other dataset contains noisy background. The dataset contains 106 varieties of fruits, 18 classes of vegetables and 7 nuts classes that are listed in Table 4.

Table 4. Fruits-360 data set summary [15]

| Fruit | Classes | Fruit | Classes | Vegetable | Classes | Nuts / Grains | Classes |
|--------------|---------|---------------|---------|-------------|---------|---------------|---------|
| Apple | 13 | Mulberry | 1 | Beetroot | 1 | Chestnut | 1 |
| Apricot | 1 | Nectarine | 2 | Cantaloupe | 2 | Corn | 2 |
| Avacado | 2 | Orange | 1 | Cauliflower | 1 | Hazelnut | 1 |
| Banana | 3 | Papaya | 1 | Cucumber | 2 | Nut Forest | 1 |
| Blueberry | 1 | Passion Fruit | 1 | Eggplant | 1 | Nut Pecan | 1 |
| Cactus Fruit | 1 | Peach | 3 | Ginger Root | 1 | Walnut | 1 |
| Carambula | 1 | Pear | 9 | Kohlrabi | 1 | | |
| Cherry | 6 | Pepino | 1 | Onion Red | 3 | | |
| Clementine | 1 | Physalis | 2 | Pepper | 2 | | |
| Dates | 1 | Pineapple | 2 | Potato | 4 | | |
| Fig | 1 | Pitahaya | 1 | | | | |
| Granadilla | 1 | Plum | 3 | | | | |
| Grape | 8 | Pomegranate | 1 | | | | |
| Guava | 1 | Pomelo | 1 | | | | |
| Huckleberry | 1 | Quince | 1 | | | | |
| Kaki | 1 | Rambutan | 1 | | | | |
| Kiwi | 1 | Raspberry | 1 | | | | |
| Kumquats | 1 | Redcurrant | 1 | | | | |
| Lemon | 3 | Salak | 1 | | | | |
| Lychee | 1 | Strawberry | 2 | | | | |
| Mandarine | 1 | Tamarillo | 1 | | | | |
| Mango | 2 | Tangelo | 1 | | | | |
| Mangostan | 1 | Tomato | 9 | | | | |
| Maracuja | 1 | Watermelon | 1 | | | | |
| Melon | 1 | | | | | | |

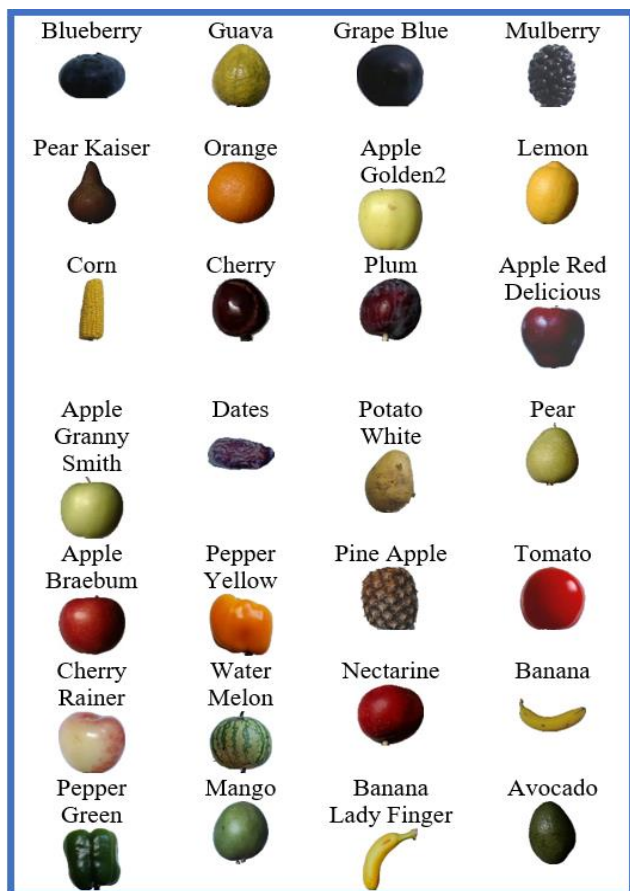


Figure 1. Sample images from fruits 360 data set

The fruits are placed on white paper and rotated using a simple motor to capture the 360-degree direction of the image. However, the fruits are placed on a white background, and due to various illuminations and lighting conditions, the background becomes noisy that is not uniform. To remove that background, a flood fill type algorithm is used. Sample images are listed in Figure 1.

Then the size of the image was scaled to 100×100 pixels. Another dataset like CIFAR uses 28×28 size images. High scaling helps in differentiating similar fruits. Finally, the dataset includes 67692 Training images, 22688 Testing images, and 103 multiclass images, a total of 90483 images.

Testing was done with test images of the dataset and also with external images of different size and types, randomly downloaded from Google. The types may be of jpg, tiff, and png. When the externally imported image size is big, it will be down-sampled. When a down-sampled image's intensity or pixel value is too low, then the image will be anti-aliased to avoid poor pixelization. This concept addresses the problem of exploiting the high and low-resolution input images.

Data Augmentation is also done with the first dataset images to generate a few more scaled, rotated, sheared, zoomed, and flipped images, which avoids overfitting during training. The augmented image sample is shown in Figure 2.

The second dataset is Fruit-and-vegetable-image-recognition contains 36 classes of fruits and vegetables images with average size of 1500×1500 pixels. Each class contains single fruit, multiple or overlapped fruits of same class and cut fruits also. Each class contains nearly 100 high quality images. But the number of images under each class is very low when we compare with first dataset. There are 3115 images of 36 classes for training and 359 images of 36 classes for testing.



Figure 2. Augmented images

The Fruit-and-Vegetable-Image-Recognition dataset [16] is a collection of images used for training and testing deep learning models to recognize and classify different types of fruits and vegetables. The images in this dataset were compiled using Bing Image Search for a personal project on food item image recognition. Please note that the creator does not own the rights to these images. If you are the copyright holder of any image in this dataset and have concerns about its use, please contact the creator to request removal. The creator will promptly honor such requests to ensure compliance with all legal obligations and respect intellectual property rights. Here are some key details about the dataset:

- Images: The dataset typically contains a large number of images of various fruits and vegetables, often with different angles, lighting conditions, and backgrounds.

- Classes: The dataset is usually labeled with multiple classes, each representing a specific type of fruit or vegetable (e.g., apple, banana, carrot, broccoli, etc.).

- Annotations: Each image is annotated with the corresponding class label, allowing machine learning models to learn from the data.

- Size: The dataset size can vary, but it's often in the range of thousands to tens of thousands of images.

- Source: The dataset may be created by researchers, collected from public sources (e.g., web scraping), or contributed by users.

- Applications: The dataset is useful for various applications, such as:

- Fruit and vegetable classification
- Quality inspection
- Automated harvesting
- Food recognition
- Nutrition analysis

Sample images are listed in Figure 3.

Overfitting is avoided using data augmentation with rescaling, shift, shear, zoom, and flip operations. The rescale set to 1./255, which will rescale the data between -0.5 to +0.5. The mean will be 0. The zoom range set as 0.2, will perform zoom in within the image size. The various augmentation operations and their properties were listed in Table 5.

Horizontal_flip set as True, will perform flip the image horizontally and generates augmented images during training to avoid overfitting. Also, it increases the generalization of the model [11].

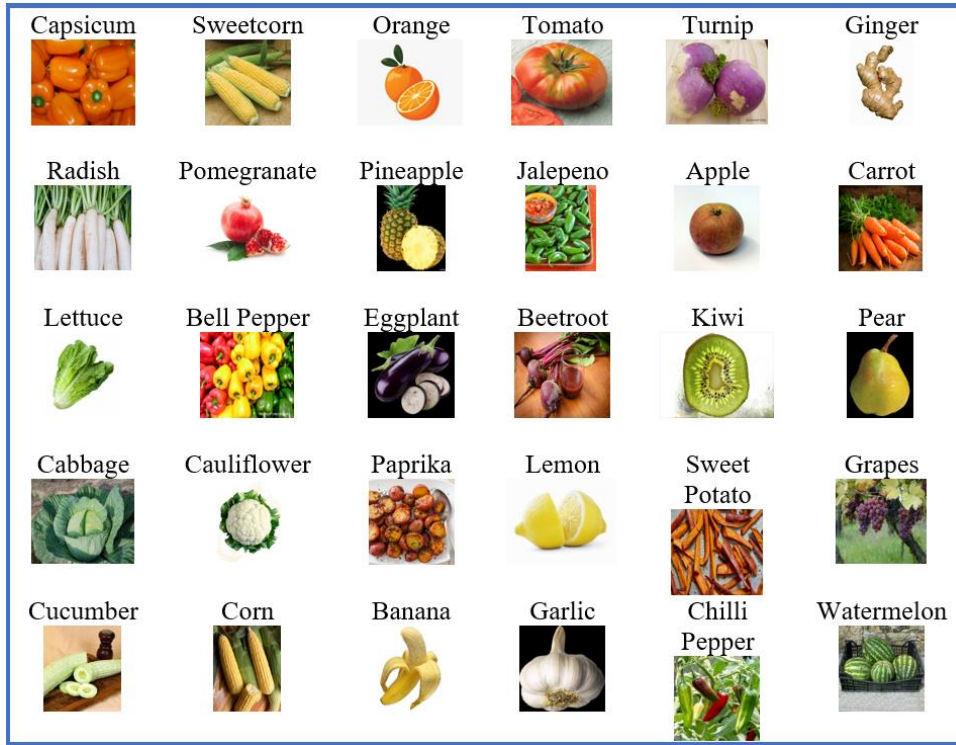


Figure 3. Sample images from fruits and vegetables image recognition data set

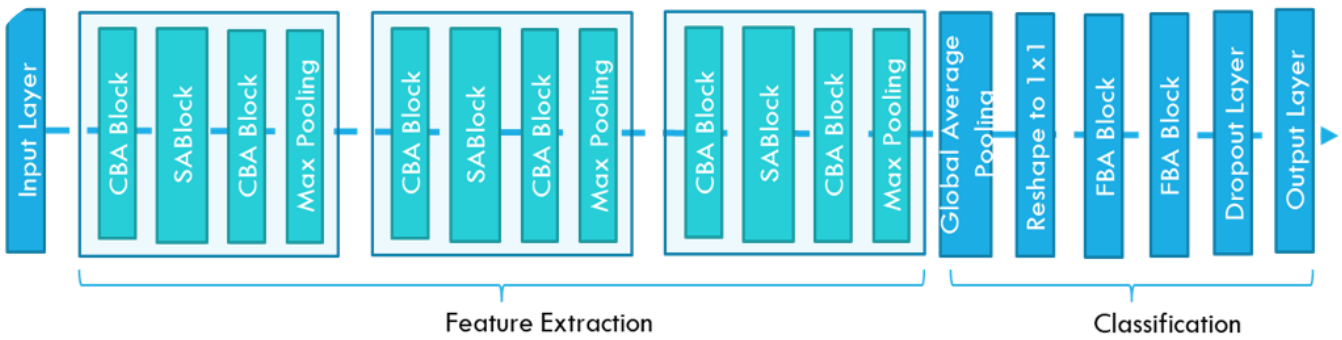


Figure 4. General structure of LwSANet

Table 5. Image augmentation

| No. | Operation | Properties |
|-----|-----------------|---|
| 1 | Resize | 224×224,3 |
| 2 | Rescale | To normalize the image as 1/255 |
| 3 | Shear Range | 2% in clock-wise and anti-clock-wise directions |
| 4 | Zoom Range | 2% in all sides |
| 5 | Horizontal Flip | True |

3.2 System design

The Proposed model is a Light weight Self Attention Network (LwSANet) Architecture shown in Figure 4. The network consists of two major blocks, backbone block for feature extraction and the second block is a classification block built with fully connected dense layers to classify the image one among 131 classes. The feature extraction block contains three units. Each unit built with a Convolution-Batch Normalization-Activation block (CBA), Self Attention block (SA) by referring [17-20], then CBA block and MaxPooling layer.

Classification block built with Fully Connected Layer-Batch Normalization-Activation (FBA). To compress the

global spatial information i.e., output of CBA Block, Reshape is used. FBA used twice then dropout is used in front of Output Dense Layer.

3.3 Feature extraction

3.3.1 CBA Block

The first unit CBA filter size is 32~128, kernel size is 3x3. The convolutional layers compute a dot product of input vector and weight vector with bias. All image's 2-dimensional pixel values are flattened and generated as vectors. The first layer inputs are multiplied with randomly generated weight values W_v .

$$F_i = \sum_{v=1}^{h \times w} (I_v \cdot W_v) + B_i \quad (3)$$

This weighted sum of the first layer is represented as F_i by finding sum of product of Input vector I_v and W_v as represented in Eq. (3). These features are normalized using Batch Normalization method [21]. This method helps to achieve faster and stable training through re-centering, rescaling and

length-direction decoupling. It also supports to use higher learning rates without affecting the gradients of the image pixels. When we use Batch Normalization with Dropout, it worse the performance of the network [22]. CBA block is built as Convolution layer with filters, kernel size, strides, batch normalization and activation function shown in Figure 5.

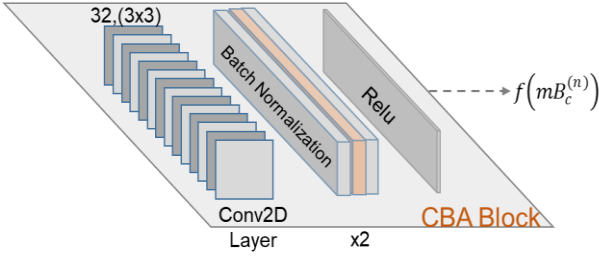


Figure 5. Architecture of CBA block

Each channel input features are normalized into zero mean (μ_c) and unit variance using computation of the mean and standard deviation (σ_c) as in Eq. (4):

$$\begin{aligned} \mu_c &= \frac{1}{N} \sum_{n=1}^N I v_c^{(n)}, \\ \sigma_c &= \sqrt{\frac{1}{N} \sum_{n=1}^N (I v_c^{(n)} - \mu_c)^2 + \alpha} \end{aligned} \quad (4)$$

where, Iv is an input vector α is a small constant. Considering the input layer over a mini batch normalization, the inputs N samples share the same channel are normalized together [17]. It can be represented as in Eq. (5):

$$mB_c^{(n)} = \frac{1}{\sigma_c} (I v_c^{(n)} - \mu_c) \quad (5)$$

During the complex scene evaluation, the normalization procedure fits the distribution of the input and recovers the features. The recovered features can be computed using linear transformation with channel parameter β as in Eq. (6):

$$Rf_c^n = M(mB_c^{(n)}; \theta) + \beta \quad (6)$$

These normalized features are then passed through the ‘relu’ activation function to produce the state of the layer L2. Relu is a non-linear activation function which maps the input values to either 1 or 0 directly. In convolutional neural network, maximum of 50 to 60 percent of the hidden units are activated, because the weight vector Wv defined randomly. It can be represented as in Eq. (7):

$$\begin{aligned} f(mB_c^{(n)}) &= \max(0, mB_c^{(n)}) = \frac{mB_c^{(n)} + |mB_c^{(n)}|}{2} \\ &= \begin{cases} mB_c, & \text{if } mB_c > 0 \\ 0, & \text{if } mB_c \leq 0 \end{cases} \end{aligned} \quad (7)$$

Except output layer all the other layers were incorporated with Relu activation function.

3.3.2 Self-Attention (SA) Network block

SA Block forms a self-attention on channels which contains two units, Reshape and Self-Attention Modules, it takes transformed input from CBA Block show in Figure 6. The reshape unit maps the input mB_c with the feature maps F to compress the global spatial information generated by CBA Block into a channel description. The output of reshaping unit is represented as F_r . The self-attention on channels increases the feature identification on images by convolutional layers also reduces the irrelevant noises [23].

The input $mB_c^{(n)}$ denoted as $F \in Re^{H \times W \times C'}$ mapped with the feature map and Reshaping operation is applied to the features with height, width and channel which results a channel descriptor by aggregating feature maps and spatial dimension of the input, represented as $R \in Re^{H \times W \times C'}$ and the transformed output R , as represented in study [17]. The convolution’s filter kernel $V=[v_1, v_2, \dots, v_c]$ used with parameter and generates the output $R=[r_1, r_2, \dots, r_c]$ as in Eq. (8), where,

$$r_c = v_c * F = \sum_{s=1}^{C'} v_c^s * Re^s \quad (8)$$

The interdependencies between the features increases the sensitivity of the informative features but there may be a chance for exploiting these due to transformation in the subsequent operation. This can be avoided by introducing global pooling averaging to generate channel wise statistics z achieved by as in Eq. (9),

$$z_c = F_{sq}(r_c) = \frac{1}{HxW} \sum_{i=1}^H \sum_{j=1}^W r_c(i, j) \quad (9)$$

The Self-Attention block captures channel-wise dependencies of features from the data received from Reshape block to create non-linear interaction between channels and the mutually exclusive relationship between non-linear must be learnt [24]. The output of Self-Attention block is attained by rescaling the Re with the activation as mentioned in Eq. (10).

$$\tilde{x}_c = F_{scale}(Re_c, s_c) = s_c Re_c \quad (10)$$

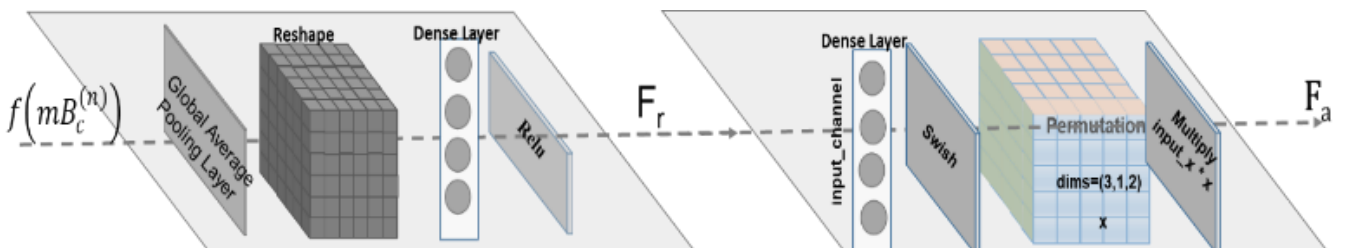


Figure 6. SANet block architecture

Its major functionality is built with a dense layer, permute and multiply functions. Self-Attention block recalibrates the features in the feature maps by identifying the discriminative features. Here the weight matrix of each layer is compressed and applied to dense layer to avoid improper conflict selection of features and packing. Also, it reduces the conflict features.

First Dense layer activated with Relu and the second dense layer in SA Block activated with swish activation function. Even though swish is slower than Relu, for getting more reliable and better results for complex data, it is used.

3.3.3 Classification

At the end of feature extraction block, Global Average Pooling layer is incorporated instead of Dropout and Flattening which takes average value of each feature map. Flattening divides the two dimensional image into one dimensional image which increases the length and processing cost [25]. Average Pooling layer maintains the size of the previous convolutional layer. The resulting feature size will be equal to the size of the previous layer feature map. It considers the average value in each feature map to reduce the size of the activations without compromising the performance of the network. Dropout is only two times, that is after three sets of pooling layers. Then the pooled data converted into a vector using reshape with 1x1.

3.3.4 FBA Block

FBA Block built as Fully connected layer with input_channels, kernel_initializer, BatchNormalizaion and Activation shown in Figure 7. The model has two FBA Blocks that is the second and third last layer with input_channels 270 and 350. Output layer is also a dense layer with input_channels 131 in case of Dataset1 and 36 in case of Dataset2. Which is equal to the number of classes to be classified. Because, the backbone Feature Extraction block contains CNN layers with filter sizes 32, 64 and 128.

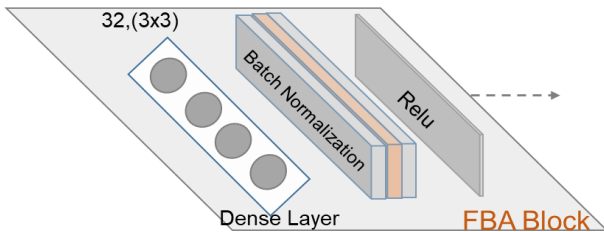


Figure 7. Architecture of FBA block

```
12/12 [=====] - 12s 984ms/step
array([[3.0907202e-03, 1.0379327e-03, 5.1989166e-05, ..., 2.6765601e-03,
        7.1900507e-04, 3.3482336e-02],
       [1.2793930e-03, 4.0204212e-04, 1.1050309e-03, ..., 3.0551510e-04,
        2.4171358e-03, 6.1459700e-03],
       [7.2111484e-06, 8.5225274e-06, 5.3606293e-04, ..., 1.2000942e-06,
        2.1255946e-04, 8.3640138e-05],
       ...,
       [1.3134971e-04, 2.8354359e-06, 3.1341463e-07, ..., 5.0387443e-03,
        3.2870670e-05, 9.7142786e-01],
       [1.7987770e-03, 8.9300466e-01, 3.3091626e-04, ..., 5.9029935e-06,
        4.3114534e-04, 7.3397241e-05],
       [8.2323082e-02, 2.6921165e-04, 1.8234755e-01, ..., 5.2331202e-03,
        1.3928839e-03, 1.2057469e-04]], dtype=float32)
```

Figure 8. Prediction probabilities of the test images

Fully connected layers use back propagation and feed forward computation to accomplish learning and inference on the entire image at once to predict the dense output from the entire image. The last output dense layer is incorporated with SoftMax, which transforms the raw vector values into vector of probabilities, to create a distribution of probabilities over the input classes. The prediction probabilities of the test images were shown in Figure 8.

$$S(O)_i = \frac{e^{O_i}}{\sum_{j=1}^N e^{O_j}} \quad (11)$$

The value of $e \approx 2.718$ [12]. All the O_i is the values of the input vector of the output image that values may be of positive, negative and zero, even though the input is always positive because of $e^{O_i} \sum_{j=1}^N e^{O_j}$ Normalizes the values and makes the sum of all values to 1. N is the number of Input classes in Eq. (11).

Dense layers are used to classify the images based on the features extracted. If there is one Dense layer, that uses the edge feature to classify the images. Here the data set contains 131 class of fruits. In apple category itself, 13 varieties are there. To classify these 13 categories differently, the in depth feature like texture, color are needed. To improve the classification more effectively, more number of Dense layers we need. Dense layer takes input of any size and produces output of corresponding size by resampling. There may be loss in last dense layer that can be represented as in Eq. (12),

$$lf(L_{ij}) = \sum_{ij} lf'(L_{ij}; \phi) \quad (12)$$

Loss function lf calculate on the layer L_{ij} by summing the gradient of its spatial components. This gradient descent on lf calculated on the whole image using feedforward and back propagation by considering the entire image by minibatch.

Activation Function:

The activation function supports to improve the training accuracy of the model. The SA Block dense layers use Relu and swish as an activation function. CBA Block and FBA Block also uses Relu as activation function. The function of Relu is defined as $f(x) = \max(0, x)$; where $f(x) = 1$ if $x > 0$ and $f(x) = 0$ if $x \leq 0$.

The max operation in Relu is faster than the tanh operation of sigmoid function. Therefore, Relu makes the hidden unit operations as light weight. Gradient at the infinity is not zero in Relu, that's what it converges faster and facilitate gradient disappearance [4].

The swish function also used in SA Block's second Dense layer. Swish is a smooth non-monotonic function which has lower bound and without upper bound. When compare to the Relu, Swish has very good convergence performance [8]. The function swish defined as $f(x) = x * (\text{sigmoid}(\beta x))$ [26]. The sigmoid function defined as $f(x) = 1 / (1 + e^{-x})$. In proposed system, the β defined as 1, also it acts as Sigmoid-weighted Linear Unit [13]. When $\beta = 1$, it acts as $f(x) = x * \text{sigmoid}(x)$, then the output ranges from -0.5 to ∞ [8].

The output layer uses SoftMax activation function as output classifier to represent the probability distribution over 131 output classes first dataset fruits-360 and 36 output classes for the second dataset Fruits and Vegetables for Image Recognition. It used for predicting a class from multiple disjunct classes and its probability lie between 0 and 1.

SoftMax squeezes the input vector probability between 0 and 1, the larger input vector will correspond to larger probability, that can be calculated by applying e^{z_i} exponential function to each component and normalizes these values by dividing the sum of all these exponentials.

4. ENVIRONMENTAL SETUP

Experiment done with Kaggle with accelerator GPU T4×2. The proposed model was developed using Python, TensorFlow, Keras and output visualized using matplotlib. Two data sets were used for train, test and validate. The first data set is fruits-360 [15], which contains 131 classes of 90483 images and splitted as 67692 for training and 22688 for testing. The second dataset Fruits and Vegetables for Image Recognition contains 3825 images of 36 classes, each image contains many numbers of fruits of same class. Among them 359 images used for Testing and 3466 images for Training. The images were resized into 224×224 pixels. Different dataset and class have different number of sample images, there is no balancing applied between classes. During training the learning rate was set to 0.001. During execution it will monitor val_loss, if the epoch continuously not showing any progress, it will reduce the learning rate with early stopping method. This slows down model learning and increases the likelihood of reaching a local or global minimum, causing the model to converge quickly [13]. For first dataset the model run for 20 epochs and for second dataset it was 50. The other parameter settings were listed in Table 6.

Simple and computationally efficient stochastic objective function optimizer ‘Adam’ with categorical cross-entropy used for gradient based optimization [27]. The loss may be different when the same content with different type of images. Residual-like connection used to take better advantage of multiscale attention features and contrastive loss with weight 0.1 is added with Cross-Entropy loss to reduce the loss [28].

Table 6. Hyperparameters

| S.No | Hyperparameters | Value Set |
|------|-----------------------|---------------------------|
| 1 | Input shape | 100×100 224×224 |
| 2 | Kernal Initializer | He_normal |
| 3 | Activations | Swish Relu |
| 4 | Stride | 1 |
| 5 | Filter Size | 32 64 128 |
| 6 | Epochs | 20 50 |
| 7 | Batch Size | 32 |
| 8 | Class Mode | Categorical |
| 9 | ReducedLr – min_delta | 0.0001 |
| 10 | Early Stopping | restore_best_weights=True |
| 11 | Optimizer | Adam |
| 12 | Leraning Rate | 0.001 |
| 13 | Output Classes | 131 36 |

4.1 Kernel initialization

The weight of the neurons in the network initialization plays important role in improving training accuracy during the staring stage of the training. At the initial stage, the convolution kernel and the training samples are independent to each other. Too large initialization results to exploding the gradients and too small initialization results to vanishing gradients problems. We didn’t use any pretrained model

weights for our proposed model initialization. When the depth of the network and the samples are high, he_normal works well when we compare with other initializers [29]. Based on result-based approach, he_normal kernel initializer is preferred for our proposed LwSANet Model.

4.2 Experimental results and discussion

To evaluate the performance of our LwSANet it is compared with LeNet, VGG-16 and GoogLeNet using Transfer Leaning. For training, dataset train images were used. For testing, test images from dataset and few images downloaded from internet with different size and resolution were used. The focus is not on state-of-the-art results, so we implemented the simple architectures of LeNet and GoogLeNet, also used the pre-trained models of VGG-16.

4.3 Improving training accuracy using dropout layer

Introducing dropout layer between the convolutional layers improves the training accuracy by dropping some neurons randomly from the hidden layer. It is an effective technique for model averaging in neural networks and it reduces complex co-adaptations on the training data [30].

It is a geometric mean of predictions from an exponential number of learnt models with shared parameters that is about equal weighted. Dropout makes filter value to 0 [31] and generates noisy input to the next fully connected layer and prevents them from developing co-dependency and overfitting [32].

4.4 Analysis using Fruits-360 dataset

Model trained and tested with single fruit images. Few multiple fruit images were also tested. The preprocessed 100x100 pixel size images were taken as input. The same set of sample images were taken for comparison with existing models LeNet, VGG-16, GoogLeNet, MobileNet, MicroNet, SqueezeNet and the proposed LwSANet model. All the models were trained for 20 epochs. Batch size set to 32. The validation accuracy is listed in Table 7.

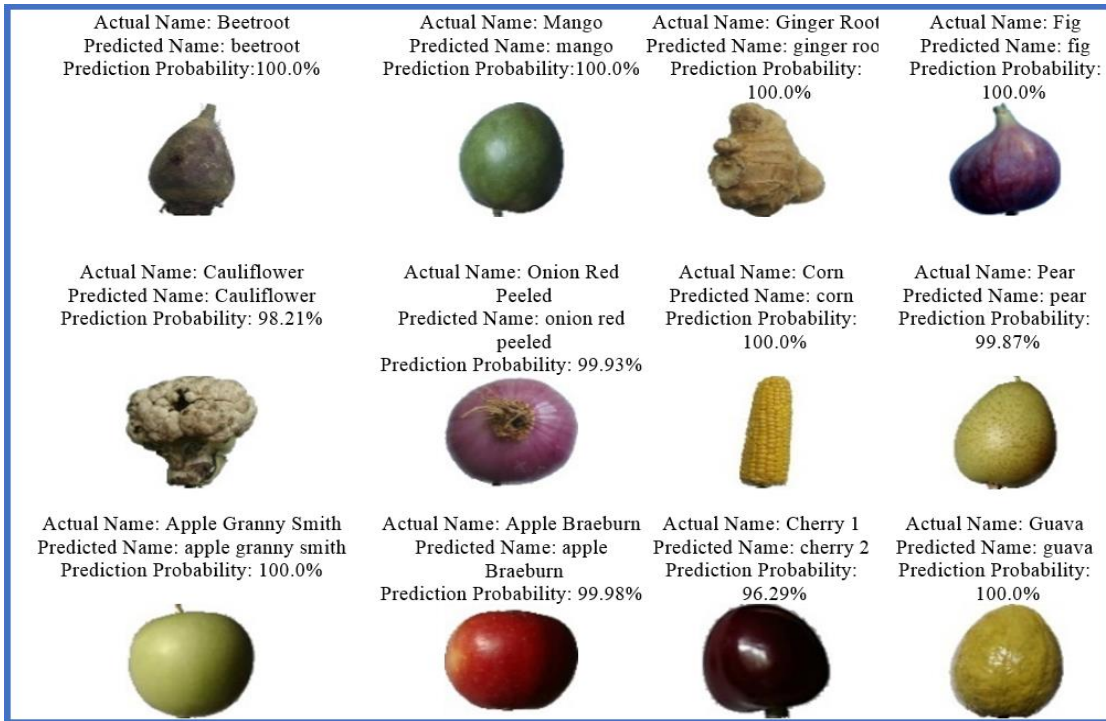
4.5 Analysis using fruits-and-vegetables-image-dataset

Each image in this dataset contains multiple number of fruit image of same class. Each class contains 100 training images and 10 testing images. Due to this limited number of images, data augmentation is applied on the training data to avoid overfitting. All the models trained for 50 epochs. Batch size set to 5 because of dissimilarity in images. The validation accuracy represented in Table 8.

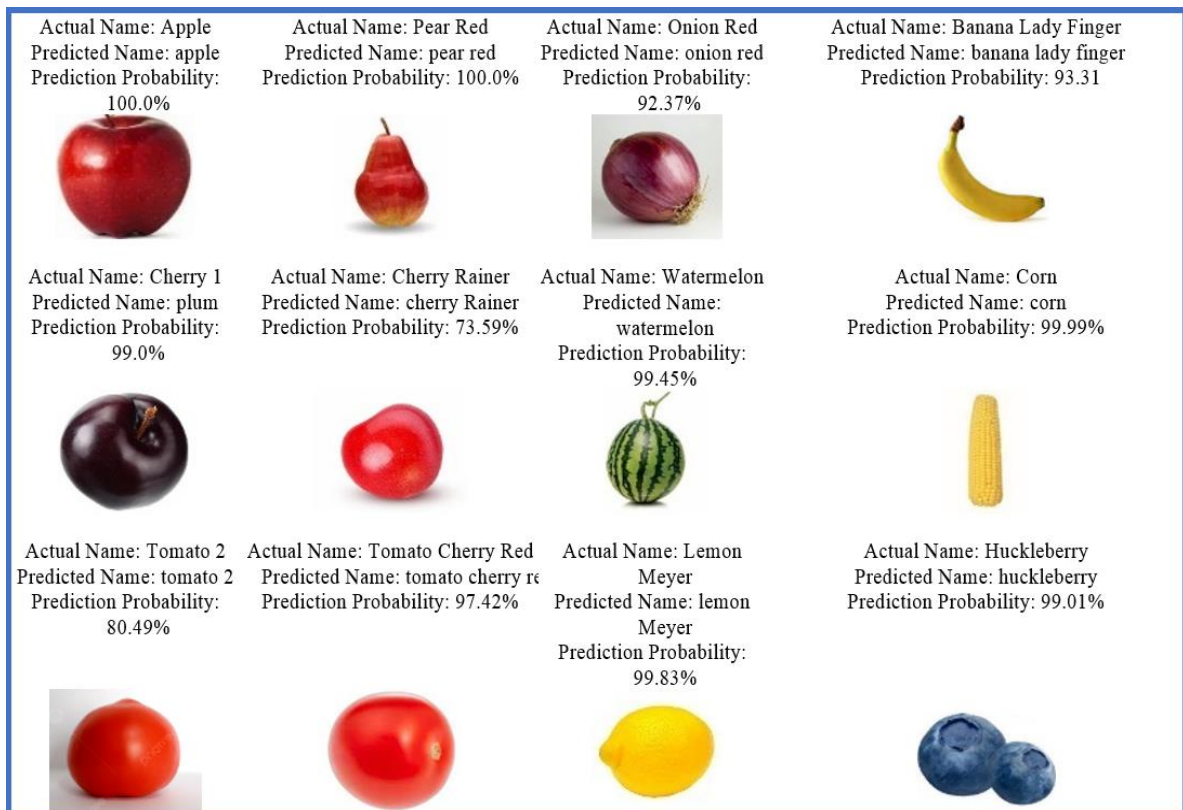
The prediction performance of proposed models is shown in Figures 9(a) and (b). Figure 9 (a) represents the images from the test data of the dataset 2. 12 sample images were tested and the predictions were represented with their probability of the prediction. Even though the prediction probability of the images were less, all the images were predicted correctly. The probability may be affected due to the size reduction of the high-quality images. The prediction of Google search images was represented in Figure 9(b). The tested 12 sample images of different classes of fruits and vegetables were predicted correctly with good probability.

Table 7. Accuracy of the models when training with dataset1-20 epochs













| Model | Val_Accuracy @Epoch-5 | Val_Accuracy @Epoch-10 | Val_Accuracy @Epoch-15 | Val_Accuracy @Epoch-20 |
|-------------|-----------------------|------------------------|------------------------|------------------------|
| LeNet | 99.04% | 99.79% | 100% | 100% |
| VGG-16 | 84.60% | 94.77% | 97.33% | 98.18% |
| GoogLeNet | 92.76% | 96.52% | 97.99% | 98.25% |
| MobileNet | 65.8% | 73.5% | 79.21% | 84.4% |
| SqueezeNet | 76.2% | 78.6% | 80.7% | 80.15% |
| Shuffle Net | 77.5% | 79.21% | 81.3% | 85.03% |
| LwSANet | 97.85% | 99.71% | 99.77% | 99.93% |















(a) Dataset-1 test images- predicted by LwSANet–after 20 epochs



(b) Google search images- predicted by LwSANet–after 20 epochs

| | | | |
|--|---|--|---|
| <p>Actual Name: Banana Predicted Name: banana Prediction Probability: 59.39%</p>  | <p>Actual Name: Bell Pepper Predicted Name: bell pepper Prediction Probability: 75.12%</p>  | <p>Actual Name: Cabbage Predicted Name: cabbage Prediction Probability: 94.96%</p>  | <p>Actual Name: Carrot Predicted Name: carrot Prediction Probability: 91.33%</p>  |
| <p>Actual Name: Sweet Corn Predicted Name: sweetcorn Prediction Probability: 67.83%</p>  | <p>Actual Name: Grapes Predicted Name: grapes Prediction Probability: 93.05%</p>  | <p>Actual Name: Kiwi Predicted Name: kiwi Prediction Probability: 89.19%</p>  | <p>Actual Name: Mango Predicted Name: mango Prediction Probability: 72.31%</p>  |
| <p>Actual Name: Pear Predicted Name: pear Prediction Probability: 63.9%</p>  | <p>Actual Name: Pineapple Predicted Name: pineapple Prediction Probability: 56.87%</p>  | <p>Actual Name: Pomegranate Predicted Name: pomegranate Prediction Probability: 98.71%</p>  | <p>Actual Name: Tomato Predicted Name: tomato Prediction Probability: 66.44%</p>  |

(c) Data set-2 test images- predicted by LwSANet – after 50 epochs

| | | | |
|---|---|--|---|
| <p>Actual Name: Apple Predicted Name: apple Prediction Probability: 86.3%</p>  | <p>Actual Name: Mango Predicted Name: mango Prediction Probability: 82.46%</p>  | <p>Actual Name: Grapes Predicted Name: grapes Prediction Probability: 52.11%</p>  | <p>Actual Name: Bell pepper Predicted Name: bell pepper Prediction Probability: 69.13%</p>  |
| <p>Actual Name: Pineapple Predicted Name: pineapple Prediction Probability: 76.43%</p>  | <p>Actual Name: Pear Predicted Name: pear Prediction Probability: 90.04%</p>  | <p>Actual Name: Grapes Predicted Name: grapes Prediction Probability: 94.9%</p>  | <p>Actual Name: Kiwi Predicted Name: kiwi Prediction Probability: 75.85%</p>  |
| <p>Actual Name: Potato Predicted Name: potato Prediction Probability: 87.49%</p>  | <p>Actual Name: Tomato Predicted Name: tomato Prediction Probability: 88.35%</p>  | <p>Actual Name: Bell pepper Predicted Name: bell pepper Prediction Probability: 51.49%</p>  | <p>Actual Name: Tomato Predicted Name: tomato Prediction Probability: 88.7%</p>  |

(d) Google search images- predicted by LwSANet–after 20 epochs

Figure 9. Performance of LwSANet on various image sets over different training epochs

Table 8. Accuracy of the models when training with dataset2-50 epochs

| Model | Val_Accuracy @Epoch-10 | Val_Accuracy @Epoch-20 | Val_Accuracy @Epoch-30 | Val_Accuracy @Epoch-40 | Val_Accuracy @Epoch-50 |
|----------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
| LeNet | 77.78% | 98.74% | 98.70% | 96.46% | 98.90% |
| VGG-16 | 69.18% | 70.63% | 70.63% | 70.63% | 70.63% |
| GoogLeNet (Aux1_Acc) | 49.05% | 55.87% | 63.66% | 67.36% | 70.68% |
| MobileNet | 91.17% | 96.87% | 97.72% | 97.92% | 97.72% |
| SqueezeNet | 24.69% | 32.05% | 38.16% | 47.06% | 56.22% |
| ShuffleNet | 62.82% | 68.59% | 72.14% | 77.75% | 80.16% |
| LwSANet | 64.35% | 71.03% | 91.09% | 95.54% | 98.10% |

4.6 Accuracy and loss function

The learning is based on the gradient function, which is much easier to smooth the image towards an increase in the gradient. This process reduces the noise in the image, improves learning rate and reduces the loss.

$$I_k = I_{k-1} - \epsilon \frac{\partial E(I)}{\partial I} \quad (13)$$

where, ϵ is a scalar constant. The real-valued gradient vector I is iteratively adjusted using the gradient decent algorithm $E(I)$ as in Eq. (13).

The training done until it reaches the accuracy of 99%, but that reduces prediction or validation accuracy. Most of the images tested from external source was predicted wrongly. The reason behind that is the model bi-hearted the images as such. That affects prediction accuracy. The Accuracy and Loss comparison between proposed model and predefined models were shown in Table 9.

4.7 Evaluation metrics

Set of predicted labels of images were compared with actual labels of that images from the dataset were compared. It is calculated by intersection of two labels by union of two labels for this Jaccard coefficient is utilized. The various parameters evaluated using this are, precision in Eq. (14), recall in Eq. (15) and F1 in Eq. (16).

$$Precision = \frac{TP}{TP + FP} \quad (14)$$

$$Recall = \frac{TP}{TP + FN} \quad (15)$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \quad (16)$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (17)$$

where, TP is True Positive, TN is True Negative, using these the metrics precision, recall f1 and accuracy can be calculated as in Eq. (17).

The existing successful models LeNet, VGG-16, GoogLeNet and proposed model LwSANet were tested in four sets of images. 12 single fruit images taken from Fruit-360 dataset and another 12 single fruit images taken from Google Search. The prediction results were shown in Figure 10.

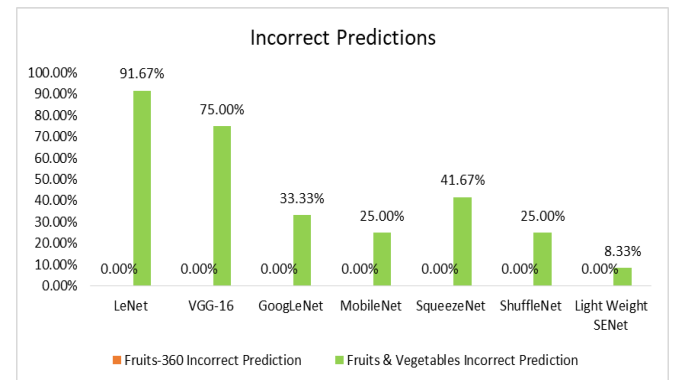
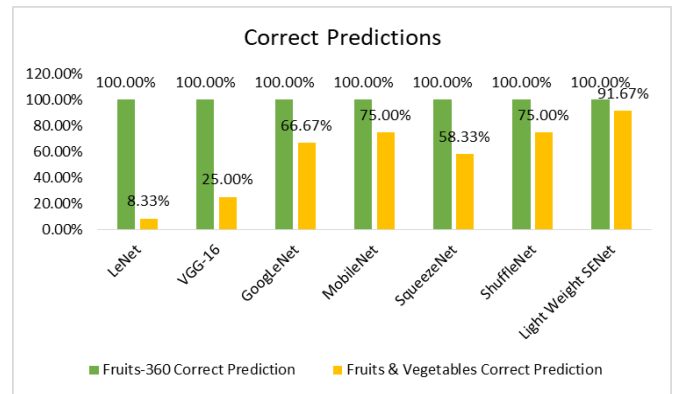
Accuracy score was compared while training using first dataset during 5,10,15 and 20 epochs. All the models were trained for 20 epochs. Because, when the accuracy reaches above 99%, the model bi-hearted the images and the prediction

accuracy was reduced. So, the average limit of 20 epochs were used. The accuracy score of Dataset 1 and Dataset 2 were shown in Figures 11 and 12.

Sample predictions were represented with the images depicted below. Because of the number large number of layers in Google, it took much amount of time for training. In epoch 50 it reached 70% accuracy, also the prediction accuracy is less. So, GoogLeNet only trained for 75 epochs. After that, it predicted 100% accurately. Sample images were listed in Figure 13.

Table 9. Accuracy and loss comparison

| Model | Fruits-360 Dataset | | Fruits-and-Vegetable Dataset | |
|----------------------|--------------------|---------------|------------------------------|-------------|
| | Accuracy | Loss | Accuracy | Loss |
| LeNet | 100% | 1.10 | 96.10% | 0.34 |
| VGG-16 | 96.61% | 0.12 | 70.63% | 4.82 |
| GoogLeNet (Aux1_Acc) | 98.25% | 0.05 | 71.19% | 0.91 |
| MobileNet | 84.4% | 0.14 | 97.72% | 0.24 |
| SqueezeNet | 80.15% | 0.54 | 56.22% | 5.21 |
| ShuffleNet | 85.03% | 0.2 | 80.16% | 0.42 |
| LwSANet | 99.93 | 0.0028 | 98.2 | 0.23 |

**Figure 10.** Correct and incorrect classification accuracy comparison with state-of-the-models

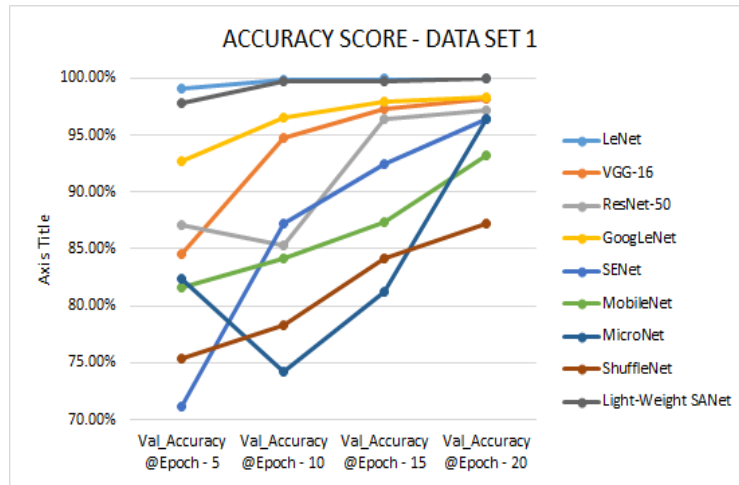


Figure 11. Accuracy score for dataset 1

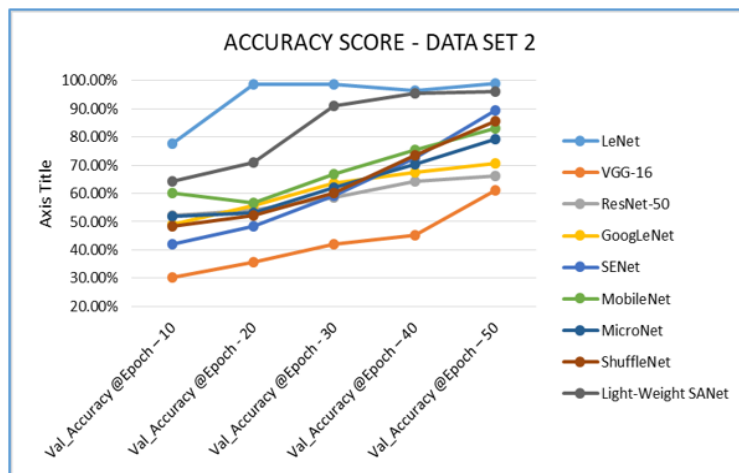
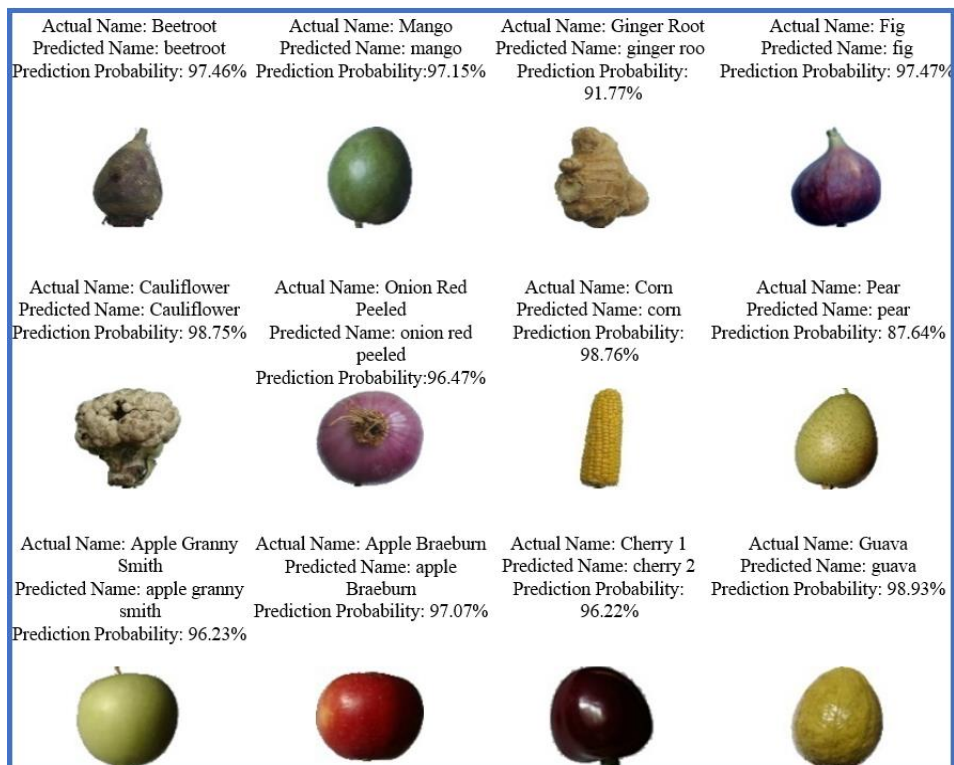


























Figure 12. Accuracy score for dataset 2



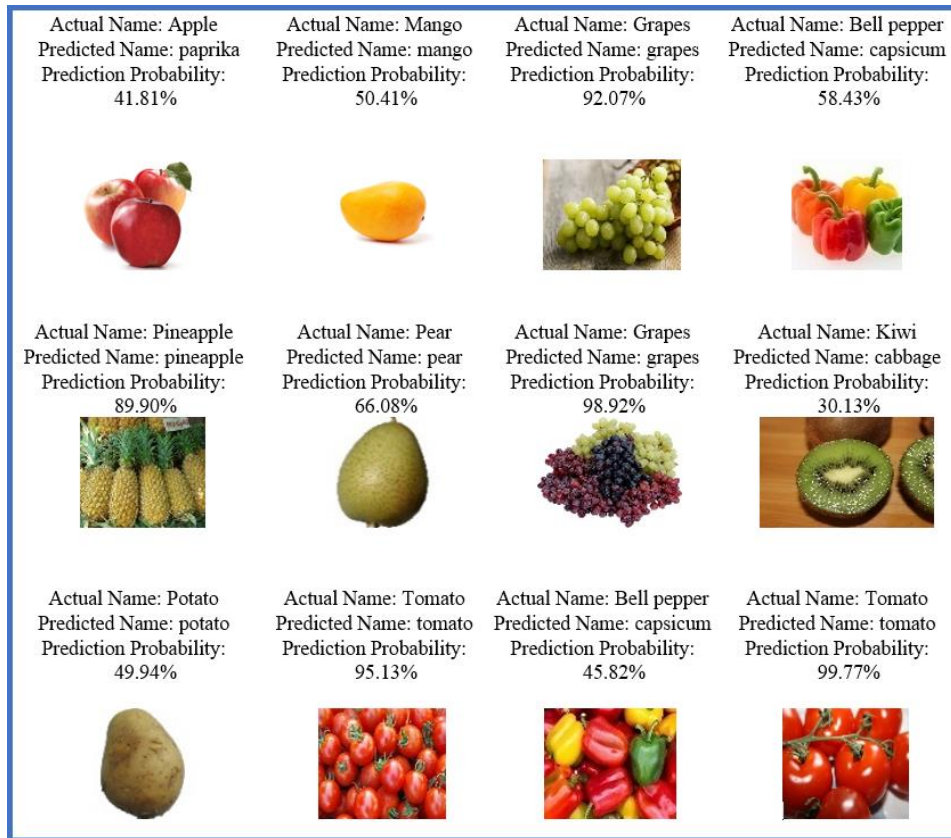
(a) GoogLeNet performance for dataset 1 test images

| | | | |
|--|--|--|---|
| Actual Name: Apple Predicted Name: nectarine Prediction Probability: 10.53% | Actual Name: Pear Red Predicted Name: pear red Prediction Probability: 86.00% | Actual Name: Onion Red Predicted Name: onion red Prediction Probability: 82.21% | Actual Name: Banana Lady Finger Predicted Name: banana lady finger Prediction Probability: 48.14% |
|  |  |  |  |
| Actual Name: Cherry 1 Predicted Name: cherry 1 Prediction Probability: 47.69% | Actual Name: Cherry Rainer Predicted Name: pear red Prediction Probability: 32.20% | Actual Name: Watermelon Predicted Name: watermelon Prediction Probability: 16.20% | Actual Name: Corn Predicted Name: corn Prediction Probability: 99.32% |
|  |  |  |  |
| Actual Name: Tomato 2 Predicted Name: tomato 2 Prediction Probability: 28.37% | Actual Name: Tomato Cherry Red Predicted Name: tomato 2 Prediction Probability: 35.15% | Actual Name: Lemon Meyer Predicted Name: lemon Meyer Prediction Probability: 93.93% | Actual Name: Huckleberry Predicted Name: Blueberry Prediction Probability: 38.45% |
|  |  |  |  |

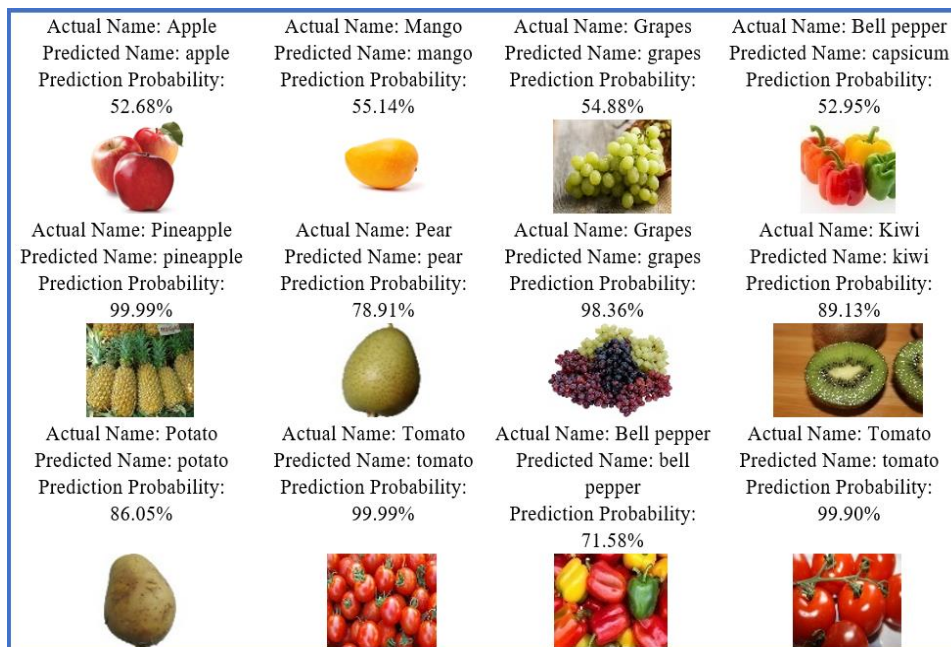
(b) GoogLeNet performance for single fruit google search (GS) images

| | | | |
|--|--|---|---|
| Actual Name: Banana Predicted Name: banana Prediction Probability: 83.20% | Actual Name: Bell Pepper Predicted Name: bell pepper Prediction Probability: 51.68% | Actual Name: Cabbage Predicted Name: cabbage Prediction Probability: 80.74% | Actual Name: Carrot Predicted Name: carrot Prediction Probability: 80.18% |
|  |  |  |  |
| Actual Name: Sweet Corn Predicted Name: corn Prediction Probability: 84.25% | Actual Name: Grapes Predicted Name: grapes Prediction Probability: 97.03% | Actual Name: Kiwi Predicted Name: cucumber Prediction Probability: 28.61% | Actual Name: Mango Predicted Name: mango Prediction Probability: 82.28% |
|  |  |  |  |
| Actual Name: Pear Predicted Name: pear Prediction Probability: 96.58% | Actual Name: Pineapple Predicted Name: pineapple Prediction Probability:99.82% | Actual Name: Pomegranate Predicted Name: pomegranate Prediction Probability: 95.65% | Actual Name: Tomato Predicted Name: tomato Prediction Probability: 98.17% |
|  |  |  |  |

(c) GoogLeNet performance for multiple fruit image from ds 2-50 epochs



(d) GoogLeNet performance for multiple fruit images from google search



(e) Performance of GoogLeNet model-after 75 epochs

Figure 13. Performance of SANet

Table 10. GMAC, GFLOP, top-1 error and top-5 error analysis

| | GMAC | GFLOPs | Parameters | Top-1 Error | Top-5 Error |
|--------------------|--------|--------|------------|-------------|-------------|
| LeNet | 7.8 | 15.7 | 0.99M | 91.67% | 2.29 |
| vgg16 | 15.52 | 31.04 | 14.7M | 75.0% | 1.29 |
| GoogLeNet | 1.51 | 3.02 | 7.5M | 33.33% | 5.17 |
| mobilenet_v2 | 320.3 | 0.640 | 4.2M | 24.0% | 1.16 |
| squeezenet1_0 | 836.82 | 1.673 | 1.25M | 38.27% | 2.15 |
| shufflenet_v2_x2_0 | 44.77 | 0.895 | 1.47M | 30.75% | 0.88 |
| LwSANet | 188.65 | 0.431 | 0.25M | 8.33% | 0.02 |

Table 11. Computational time

| Model | Computational Time |
|-------------|--------------------|
| LeNet | 28 ms |
| VGG-16 | 46 ms |
| GoogLeNet | 45 ms |
| MobileNet | 145 ms |
| SqueezeNet | 35 ms |
| Shuffle Net | 30 ms |
| LwSANet | 20 ms |

4.8 Ablation experiment and analysis

In the ablation experiment, the LwSANet model tested with different dataset images, different number of classes, changes in the performance by increasing the number of CBA Block, MFLOPs count and also the performance metrics compared with state-of-the-art deep neural networks and resource constraint networks.

The dataset1 images used with 100x100 resolution. Even though the resolution of the image is low, training with 131 number of classes images, increases the MFLOPs. When we compare with state-of-the-art deep neural network with ≥ 1 GFLOPs models, proposed model's floating point operations count limited with MFLOPs. Proposed model's MFLOPs also compared with state-of-the-art resource constrained networks like MobileNet, SqueezeNet, ShuffleNet. Proposed model's FLOP count is 0.431 GFLOPs, recorded in Table 10.

Introduction of CBA and FBA blocks decreases the number of parameters by 12% MFLOPs by 16.3%. Also, the proposed model performance analyzed with different number of CBA and FBA Blocks. Increasing a greater number of such blocks doesn't make much impact on the performance of the model. Increasing a greater number of time training also freezes the accuracy.

4.9 Computational time

Table 11 presents a comparison of the calculation efficiency of several detectors.

On a 640x480 image using a GTX-1080Ti, the LeNet weights size and computation time are 7.4M and 28ms, respectively. VGG-16 (small) yields comparable outcomes; the magnitude of the weights and computation time of VGG-16 (small) measures 35.4M and 30ms in turn. In contrast to better detection performance is achieved using LeNet and VGG-16 (small). LedNet's IoU and F1 scores when combined with LW-net are 0.826 and 86.3%, which are, respectively, 4.4% and 3.9% greater than the LeNet (small). The experimental findings show that LeNet with resnet-101 achieves comparable computing efficiency to the VGG-16. Two stages make up the FasterRCNN detector: an RPN and a classification stage network. Consequently, the computation time is given in the Table 11.

4.10 Effect of batch size

The number of input samples is applied to the network's layers to minimize memory usage. The batch size significantly influences the experiment's outcome. If the batch size is too small, there may be an underfitting risk, and if the value is excessive, there may potentially be an overfitting risk be wary of overfitting.

4.11 Practical applications and deployment scenarios of LwSANet

The LwSANet framework designed for crop recognition from snapshots has beneficial applications in various fields wherein green and accurate photograph recognition is needed. It can be utilized in automatic classification systems to categorize and classify harvests primarily based on function popularity and can be used to robotically identify the fruit, decreasing the want for manual sorting and speeding up processes. Its characteristics a lightweight also makes it appropriate for mobile embedded devices, along with actual-time fruit detection on hand-held scanners for purchasers and small farmers enables cellular apps and different programs. The performance and accuracy of the version make it a valuable device in useful resource-constrained environments wherein excessive computing electricity isn't to be had.

4.12 Discussion

The novelty of LwSANet lies in its strategic use of a lightweight self-attention mechanism, tailor-made specifically for the undertaking of fruit popularity. Unlike conventional fashions like LeNet or VGG-sixteen, which depend closely on deep and huge convolutional layers, LwSANet integrates self-attention in a compact form to awareness of relevant capabilities in the photograph while minimizing computational overhead. This technique permits LwSANet to capture diffused variations in fruit textures and shapes, which can be critical for accurate type, especially in scenarios wherein fruits have similar appearances. The version's architecture is designed to stability performance and accuracy, making it appropriate for deployment on resource-limited gadgets along with cell telephones or edge computing structures. In terms of results, LwSANet outperforms several modern-day models no longer just in accuracy however additionally in performance, as evidenced by means of decrease parameter counts and quicker inference times. The particular strategies employed, including the lightweight self-interest mechanism and optimized function extraction layers, contribute to this performance enhance.

5. CONCLUSION

The increase in population day by day increases the need of nutritional foods like fruits and vegetables. Fruits and vegetables are recommended good nutrition food for all age people. Improper maintenance and categorization of these foods in supermarkets or during transportations, affects the health of one another. Rotten fruits make other fruits rotten. The problem of labour shortage to work in field have tremendous impact on quality maintenance of fruits and vegetables. Because of the technology advancements, we can address these problems through automated robots or machines. In this work we propose a deep learning based Light weight Self-Attention Network to detect and classify different fruits and vegetables from images. The proposed model is compared with State-of-the-art deep neural network models LeNet, VGG-16, GoogLeNet and Tiny network models MobileNet, ShuffleNet, SqueezeNet. Two data sets were used to evaluate the performance of the proposed model. The model demonstrated that, the way of organizing the different layers, normalization, dropout, multiply and permute improves the

performance of simple CNN models. Incorporation of Self-Attention Block enables channel wise feature recalibration, proper usage of Batch normalization and Dropouts helps to remove unwanted pixel data and to bring mattered pixel data to the next layer. By using the feature extracted with pruning, the model will produce compressed smaller and faster model to be deployed in resource constraint devices.

REFERENCES

- [1] Huynh, T.T.M., Le, T.M., That, L.T., Van Tran, L., Dao, S.V.T. (2022). A two-stage feature selection approach for fruit recognition using camera images with various machine learning classifiers. *IEEE Access*, 10: 132260-132270. <https://doi.org/10.1109/ACCESS.2022.3227712>
- [2] Gill, H.S., Khalaf, O.I., Alotaibi, Y., Alghamdi, S., Alassery, F. (2022). Multi-Model CNN-RNN-LSTM based fruit recognition and classification. *Intelligent Automation & Soft Computing*, 33(1). <http://doi.org/10.32604/iasc.2022.022589>
- [3] Moon, J., Lim, S., Lee, H., Yu, S., Lee, K.B. (2022). Smart count system based on object detection using deep learning. *Remote Sensing*, 14(15): 3761. <https://doi.org/10.3390/rs14153761>
- [4] Agarap, A.F. (2018). Deep learning using rectified linear units (relu). *arXiv Preprint arXiv: 1803.08375*. <https://arXiv.org/abs/1803.08375>
- [5] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of The IEEE*, 86(11): 2278-2324. <https://doi.org/10.1109/5.726791>
- [6] Campos-Leal, J.A., Yee-Rendón, A., Vega-López, I.F. (2022). Simplifying vgg-16 for plant species identification. *IEEE Latin America Transactions*, 20(11): 2330-2338. <https://doi.org/10.1109/TLA.2022.9904757>
- [7] Hossain, M.S., Al-Hammadi, M., Muhammad, G. (2018). Automatic fruit classification using deep learning for industrial applications. *IEEE Transactions on Industrial Informatics*, 15(2): 1027-1034. <https://doi.org/10.1109/TII.2018.2875149>
- [8] Fu, Y.S., Song, J., Xie, F.X., Bai, Y., Zheng, X., Gao, P., Wang, Z.T., Xie, S.Q. (2021). Circular fruit and vegetable classification based on optimized GoogLeNet. *IEEE Access*, 9: 113599-113611. <https://doi.org/10.1109/ACCESS.2021.3105112>
- [9] Lai, J.W., Ramli, H.R., Ismail, L.I., Hasan, W.Z.W. (2022). Real-time detection of ripe oil palm fresh fruit bunch based on YOLOv4. *IEEE Access*, 10: 95763-95770. <https://doi.org/10.1109/ACCESS.2022.3204762>
- [10] Xuan, G., Gao, C., Shao, Y., Zhang, M., Wang, Y., Zhong, J., Li, Q., Peng, H. (2020). Apple detection in natural environment using deep learning algorithms. *IEEE Access*, 8: 216772-216780. <https://doi.org/10.1109/ACCESS.2020.3040423>
- [11] An, Q., Wang, K., Li, Z., Song, C., Tang, X., Song, J. (2022). Real-time monitoring method of strawberry fruit growth state based on YOLO improved model. *IEEE Access*, 10: 124363-124372. <https://doi.org/10.1109/ACCESS.2022.3220234>
- [12] Hassam, M., Khan, M.A., Armghan, A., Althubiti, S.A., Alhaisoni, M., Alqahtani, A., Kadry, S., Kim, Y. (2022). A single stream modified mobilenet v2 and whale controlled entropy based optimization framework for citrus fruit diseases recognition. *IEEE Access*, 10: 91828-91839. <https://doi.org/10.1109/ACCESS.2022.3201338>
- [13] Zhong, S., Xu, W., Zhang, T., Chen, H. (2022). Identification and depth localization of clustered pod pepper based on improved Faster R-CNN. *IEEE Access*, 10: 93615-93625. <https://doi.org/10.1109/ACCESS.2022.3203106>
- [14] Zhang, L., Gui, G., Khattak, A.M., Wang, M., Gao, W., Jia, J. (2019). Multi-task cascaded convolutional networks based intelligent fruit detection for designing automated robot. *IEEE Access*, 7: 56028-56038. <https://doi.org/10.1109/ACCESS.2019.2899940>
- [15] Muresan, H., Oltean, M. (2018). Fruit recognition from images using deep learning. *Acta Universitatis Sapientiae, Informatica*, 10(1): 26-42. <https://doi.org/10.2478/ausi-2018-0002>
- [16] Fruits and Vegetables Image Recognition Dataset taken from, <https://www.kaggle.com/datasets/kritikseth/fruit-and-vegetable-image-recognition>, accessed on Aug. 2024.
- [17] Woo, S., Park, J., Lee, J.Y., Kweon, I.S. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3-19.
- [18] Hu, J., Shen, L., Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132-7141.
- [19] Li, P., Gai, S. (2023). Single image deraining using multi-scales context information and attention network. *Journal of Visual Communication and Image Representation*, 90: 103695. <https://doi.org/10.1016/j.jvcir.2022.103695>
- [20] Su, Z., Liu, R., Feng, Y., Zhou, F. (2023). Attention-adaptive multi-scale feature aggregation dehazing network. *Journal of Visual Communication and Image Representation*, 90: 103706. <https://doi.org/10.1016/j.jvcir.2022.103706>
- [21] Xu, Y., Xie, L., Xie, C., Dai, W., Mei, J., Qiao, S., Shen, W., Xiong, H., Yuille, A. (2023). Bnet: Batch normalization with enhanced linear transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7): 9225-9232. <https://doi.org/10.1109/TPAMI.2023.3235369>
- [22] Li, X., Chen, S., Hu, X., Yang, J. (2019). Understanding the disharmony between dropout and batch normalization by variance shift. In *Proceedings of The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2682-2690.
- [23] Ilyas, T., Khan, A., Umraiz, M., Jeong, Y., Kim, H. (2021). Multi-scale context aggregation for strawberry fruit recognition and disease phenotyping. *IEEE Access*, 9: 124491-124504. <https://doi.org/10.1109/ACCESS.2021.3110978>
- [24] Zhao, Z., Hao, K., Liu, X., Zheng, T., Xu, J., Cui, S., He, C., Zhou, J., Zhao, G. (2023). MCANet: Hierarchical cross-fusion lightweight transformer based on multi-ConvHead attention for object detection. *Image and Vision Computing*, 136: 104715. <https://doi.org/10.1016/j.imavis.2023.104715>
- [25] Tamayo-Monsalve, M.A., Mercado-Ruiz, E., Villa-Pulgarin, J.P., Bravo-Ortiz, M.A., Arteaga-Arteaga, H.B.,

- Mora-Rubio, A., Alzate-Grisales, J.A., Arias-Garzon, D., Romero-Cano, V., Orozco-Arias, S., Gustavo-Osorio, G., Tabares-Soto, R. (2022). Coffee maturity classification using convolutional neural networks and transfer learning. *IEEE Access*, 10: 42971-42982. <https://doi.org/10.1109/ACCESS.2022.3166515>
- [26] Dubey, S.R., Singh, S.K., Chaudhuri, B.B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503: 92-108. <https://doi.org/10.1016/j.neucom.2022.06.111>
- [27] Elfwing, S., Uchibe, E., Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107: 3-11. <https://doi.org/10.1016/j.neunet.2017.12.012>
- [28] Kingma, D.P., Ba, J., (2017). Adam: A method for stochastic optimization. Preprint at arXiv. <https://arXiv.org/abs/1412.6980>
- [29] Wang, Z., Shi, S., Zhai, Z., Wu, Y., Yang, R. (2022). ArCo: Attention-reinforced transformer with contrastive learning for image captioning. *Image and Vision Computing*, 128: 104570. <https://doi.org/10.1016/j.imavis.2022.104570>
- [30] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580. <https://doi.org/10.48550/arXiv.1207.0580>
- [31] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014). A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929-1958.