Attention-Aware CNN for Fruit Image Classification

Wenjia Geng*

Department of computer science, Nottingham Trent University, Nottingham, UK

*Corresponding author: wenjia.geng2020@my.ntu.ac.uk

Abstract. Fruit image classification has significant potential value in agricultural harvesting and commercial fruit trading. However, the wide variety of fruits and their complex and diverse properties make the fruit image classification using computer vision techniques with high accuracy still challenging. This paper aims to introduce attention mechanism into the convolutional neural networks to improve the models' accuracy for the classification and explore the impact of the convolutional block attention module (CBAM) implementation before and after convolution layers on model performance. Firstly a 16-layer CNN model called BasicCNN was built. Secondly, two models named BasicCNN+CBAM_v1 and BasicCNN+CBAM_v2 were built by adding CBAM layers before the first convolution layer and after the convolution layers of the BasicCNN. Then a test set was created called Real-world, containing more diverse fruit images than another open-source dataset, Fruit360, that was also used in this research. Lastly, the three network models were trained and evaluated using two datasets, Fruits 360 and Real-world. Among the tests on the Real-world test set, the accuracy of BasicCNN(16.67%) was much lower than that of BasicCNN+CBAM_v1(48.89%) and BasicCNN+CBAM_v2(46.73%). Moreover, BasicCNN+CBAM_v1 also achieved the highest accuracy on Fruit360 datasets. The results show that the CNN model obtains a performance improvement when the CBAM is added and that implementing the attention mechanism before the convolution also allows the model to achieve better performance.

Keywords: Convolutional neural network (CNN), attention mechanism, convolutional block attention module (CBAM), Fruit image classification.

1. Introduction

Nowadays, the demand for the fruit has grown with the improvement of people's living quality. However, the problem of accurately classifying fruit in the fruit industry has still not been effectively solved. The current manual approach to classifying fruit still has many drawbacks. For example, when a customer buys fruit in a supermarket, the cashier needs to know the price after identifying its category. Instead, the price cannot be confirmed by scanning the barcode on the product, like other conventional goods [1]. Therefore, a solution for fruit classification using computer vision techniques to extract fruit features can effectively overcome these difficulties [2].

Nevertheless, the subtle variations in size, contour, colour and ambient light of the fruit in the image will directly impact the final classification result [3]. Raheel's proposed 12-layer convolutional neural network (CNN) model, called FruitNet [3], tested 99.73% accuracy on the Fruits-360 dataset [4] by implementing regularisation techniques. However, because the model has not been tested on more comprehensive datasets, the inclusion of regularisation techniques has not demonstrated that it can overcome the interference of mentioned various factors in fruit classification. Notably, the Attention mechanism, which borrows from human thinking patterns to capture more valuable information, has been incorporated into the recurrent neural network (RNN) model for image classification tasks by Mnih et al. [5], and the neural network model has achieved better classification performance. Additionally, adding the attention mechanism to the CNN can also deliver performance improvements to the model. For example, in the ABCNN [6] model for sentence matching task pairs, Yin et al. placed the attention mechanism in front of the convolutional layer and the pooling layer in the CNN models, and also the models improved their performance.

According to the research mentioned above, the attention mechanism can be applied to improve a neural network's performance. Consequently, this experiment will investigate whether the CNN with the attention mechanism can obtain a similar performance improvement when executing the fruit

image classification task and whether the attention mechanism results in a performance difference when executed before versus after the convolutional layer. Aiming at the limitations of the current fruit image classification, a 16-layer convolutional neural network called BasicCNN was designed, and two variants called BasicCNN+CBAM_v1 and BasicCNN+CBAM_v2 was produced by combining the convolutional block attention module (CBAM) into the BasicCNN. The BasicCNN+CBAM_v1 model adds the CBAM before the first convolution layer, and the BasicCNN+CBAM_v2 adds the CBAM after all convolution layers. Then the authors experimented with the three networks on the Fruits360 and real-world test sets to prove the effectiveness of the performance improvement of the CNN on the fruit image classification task by adding an attention mechanism, and to determine the difference that CBAM execution before and after the convolutional layer brings to the performance of the network.

2. Method

2.1. Dataset

This project uses the open-source Fruits 360 dataset from Kaggle. The Fruits 360 presented by Mure¸san et al. [4] is a high-quality dataset that is continuously updated and covers the most common fruits. The dataset is provided with a train and a test dataset, and it contains a total of 90,483 images with a size of 100 by 100 pixels, which are divided into 131 classes of fruit. The fruit images were obtained by filming a 20 seconds video of the corresponding fruit rotating at 3 rpm in front of a white background and then extracting the frames from the video.

In addition, the author created an external test set named real-world to check the model's performance in real-world application. The test set contains a total of 524 images of 131 classes of fruit with a size of 100 x 100 pixels. The images were obtained from a google search, and each image had a different background, lighting conditions and shooting angle.

Figure 1. Fruits 360 dataset.

Figure 2. Real-world test dataset.

2.2. Basic CNN structure

The basic model named BasicCNN used in this project is based on the HoreaCNN model proposed by Mure¸san et al. [4] with improved hierarchy, pre-processed image methods and optimiser selection. Figure 3 is a graph visualisation of the CNN structure, it reveals that the BasicCNN is comprised of four convolutional layers, three max-pooling layers, four ReLU layers, two fully connected layers, and two dropout layers. Compared to the baseline HoreaCNN model, figure 4 shows the basic model makes four key changes, which are explained below:

Adding the rescale = 1./255 parameter to the ImageDataGenerator function, the value of RGB on the pixel is divided by 255, normalising the data and effectively reducing the risk of overfitting [7].

Adding a data enhancement including the parameters of shear_range, horizontal_flip and vertical_flip in the ImageDataGenerator function, which shears, flips, and scale the image in the training dataset, effectively increasing the number of data samples for training the model and implementing image enhancement [8].

Stacking the first and second convolutional layers allows the ReLU activation function to be used only once before the pooling layer. As a result, the output feature data will have more non-linear transformations, giving the model a more robust feature extraction capability.

Adding a fully connected layer with a ReLU activation function, a non-saturating nonlinearity function solves the vanishing gradient problem of neural networks and provides a fast convergence rate. Thus, it optimises the efficiency of model feature learning.

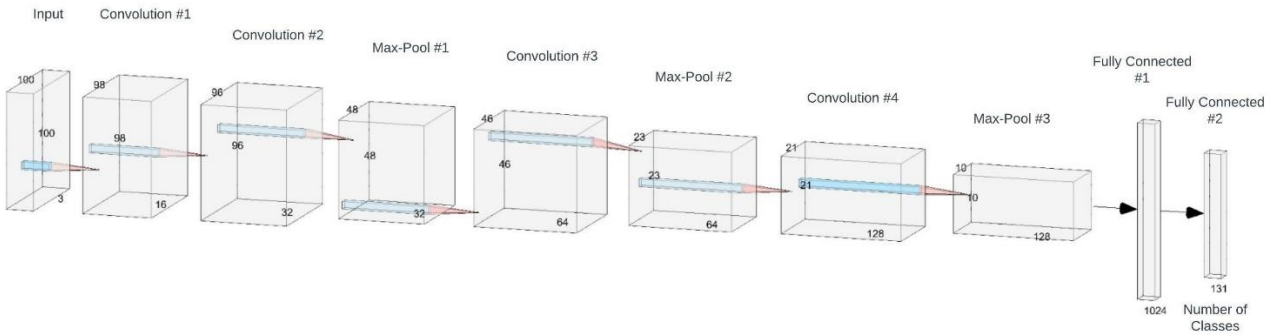

Figure 3. Basic convolution neural network structure.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (InputLayer)          [(None, 100, 100, 3)]     0

 conv1 (Conv2D)              (None, 98, 98, 16)        448

 conv1_relu (Activation)     (None, 98, 98, 16)        0

 conv2 (Conv2D)              (None, 96, 96, 32)        4640

 conv2_relu (Activation)     (None, 96, 96, 32)        0

 pool1_max (MaxPooling2D)    (None, 48, 48, 32)        0

 conv3 (Conv2D)              (None, 46, 46, 64)        18496

 conv3_relu (Activation)     (None, 46, 46, 64)        0

 pool2_max (MaxPooling2D)    (None, 23, 23, 64)        0

 conv4 (Conv2D)              (None, 21, 21, 128)       73856

 conv4_relu (Activation)     (None, 21, 21, 128)       0

 pool3_max (MaxPooling2D)    (None, 10, 10, 128)       0

 dropout1 (Dropout)          (None, 10, 10, 128)       0

 flatten1 (Flatten)          (None, 12800)             0

 dense1_relu (Dense)         (None, 1024)              13108224

 dropout2 (Dropout)          (None, 1024)              0

 dense2_softmax (Dense)      (None, 131)               134275

=================================================================
Total params: 13,339,939
Trainable params: 13,339,939
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. Basic convolution neural network details.

2.3. Attention Machanism

The attention mechanism is designed similarly to neural networks by mimicking human thinking patterns and observation behaviors to quickly find the vital information to focus on from a broad scope [9]. To make the trained model more accurate in the classification tasks, the model needs to enhance the representation ability by increasing the number of layers. However, over-increasing the parameters involves the risk of information overload, which increases the computational power required for model training and reduces the training speed. Notably, the logic of the attention mechanism, which places resources on important information, avoids wasting limited resources on unrelated information processing. Therefore, using the attention mechanism can successfully resolve both issues while improving the model's performance and accuracy of the model.

2.4. CBAM

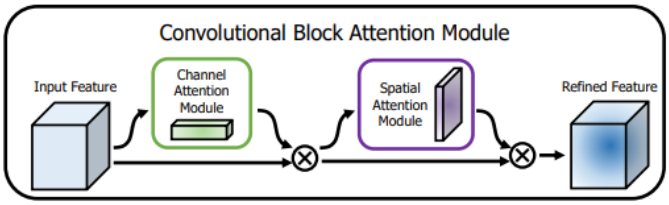

Figure 5. Convolutional block attention module structure [4].

In this experiment, the convolutional block attention module(CBAM) proposed by Sanghyun et al. [10] will be used. Figure 5 depicts the structure of the CBAM, which includes a channel attention module(CAM) followed by a spatial attention module(SAM) sequentially. The Bottleneck Attention Module (BAM) [11], also proposed by Sanghyun et al. [10], uses a design that connects CAM and

SAM in parallel rather than in series. Besides, the BAM also uses dilation convolution to increase the perceptual field of the convolutional kernel, making it more suitable for image restoration, speech synthesis, and similar tasks. However, the task of fruit classification in this experiment is more suitable to be done with CBAM, which uses regular convolution. The CBAM is highly portable and can be directly embedded into other convolutional neural networks. The CAM deals with the relationship between the assignment of feature map channels, and the SAM enables the neural network to concentrate more on the pixel-level that are decisive for classification. The sequential execution of the CAM and SAM modules enables the network to perform a more efficient extraction of features on both spatial and channel dimensions separately. The calculation procedures for the two modules can be expressed as：

$$F' = M_c(F) \otimes F, \tag{1}$$

$$F^{"} = M_s(F') \otimes F', \tag{2}$$

The final refined output $F''$of the module is obtained by multiplying the output value $M_c(F)$ of the CAM with the original input value F and then with the output value $M_s(F')$ of the SAM. The following will elaborate on the details in CAM and SAM separately：

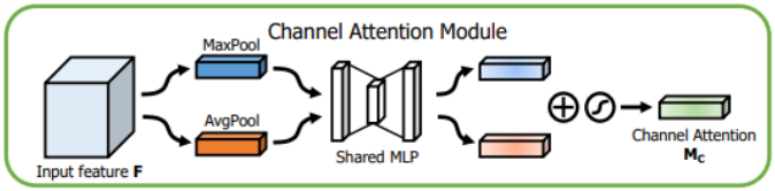

Figure 6. Channel attention module structure [4].

As depicted in Figure 6, the channel attention mechanism(CAM) feeds the feature map $F \in R^{c\times h\times w}$ through the MaxPooling and AvgPooling layers to generate two weight vectors of height and width 1, which is $c \times 1 \times 1$. The two weight vectors are then passed through the same multilayer perceptron (MLP) network, summed up, and finally passed into the sigmoid function to obtain the wanted weight coefficients $M_c(F)$. Thus, an effective compression of the channel dimension is achieved in CAM. The calculation procedure in CAM can be summarised in the following equations:

$$\mathrm{M_s(F)} = \sigma\left(\mathrm{MLP}\left(\mathrm{AvgPool(F)}\right) + \mathrm{MLP}\left(\mathrm{MaxPool(F)}\right)\right) \tag{3}$$

$$= \sigma\left(\mathrm{W_1}\left(\mathrm{W_0}\left(\mathrm{F^c_{avg}}\right)\right) + \mathrm{W_1}\left(\mathrm{W_0}\left(\mathrm{F^c_{max}}\right)\right)\right), \tag{4}$$

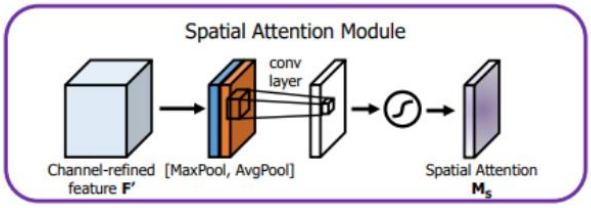

Figure 7. Spatial attention module structure [4].

The next step is to input the new feature value F', obtained by multiplying the CAM output weight $M_c(F)$ with the original input F, into the SAM. Figure 7 shows that the feature maps are first subjected to max pooling layer and average pooling layer according to the channel dimension. Then, the two output feature maps are stacked in the spatial dimension. Next, the number of channels is adjusted using a convolutional layer, and a sigmoid function normalizes the weights. Finally, the normalized weight $M_s(F')$ is multiplied by the CAM input feature value F' to produce the final refined feature F''. As a result, an effective compression of the space dimension is achieved in CAM. The equations for the SAM can then be expressed as:

$$M_s(F) = \sigma\left(f^{7x7}([AvgPool(F); MaxPool(F)])\right) \quad (5)$$

$$= \sigma\left(f^{7x7}\left(\left[F^s_{avg}; F^s_{max}\right]\right)\right), \quad (6)$$

Overall, Applying the CBAM will minimize the network model's complexity, and enhance its performance while significantly optimizing the computing power requirement.

2.5. Implementation details

In this paper, we aim to discussion the importance of visual attention and the relationship between the feature map and attention score. In specific, the author implements a total of three versions of CNN models, which are named BasicCNN, BasicCNN+CBAM_v1 and BasicCNN+CBAM_v2. The following will present the details of each network model according to the tables.

Table 1. BasicCNN network details.

Layer type	Output shape
Convolutional	98 x 98 x 16
Convolutional	96 x 96 x 32
Max pooling	-
Convolutional	46 x 46 x 64
Max pooling	-
Convolutional	21 x 21 x 128
Max pooling	-
Fully connected	1024
Softmax	131

The BasicCNN model is generated by optimizing the HoreaCNN model. Table 1 demonstrates that the BasicCNN is composed of four convolutional layers, three max-pooling layers, and one fully connected layer. The convolution layer takes the input data of size 100 x 100 x 3, uses a 3 x 3 filter to recognize the features, and then stacks the resulting feature maps to achieve perceptual field optimization. In addition, set the stride value to 1 and padding only covers the valid pixels. The max-pooling layer's kernel size is set to 2 x 2 and the stride length to 2 to reduce the feature map size after the convolution. Finally, a Softmax classifier corresponding to 131 fruit classification labels was designed behind a fully connected layer with 1024 channels.

Table 2. BasicCNN+CBAM_v1 network details.

Layer type	Output shape
CBAM	100 x 100 x 3
Convolutional	98 x 98 x 16
Convolutional	96 x 96 x 32
Max pooling	-
Convolutional	46 x 46 x 64
Max pooling	-
Convolutional	21 x 21 x 128
Max pooling	-
Fully connected	1024
Softmax	131

Based on Table 2, the BasicCNN+CBAM_v1 model is obtained by adding the CBAM layer in front of the first convolutional layer based on the BasicCNN model. On the other hand, Table 3 indicates that the BasicCNN+CBAM_v2 model is derived from the BasicCNN model by adding the CBAM layer after the third max-pooling layer.

Table 3. BasicCNN+CBAM_v2 network details.

Layer type	Output shape
Convolutional	98 x 98 x 16
Convolutional	96 x 96 x 32
Max pooling	-
Convolutional	46 x 46 x 64
Max pooling	-
Convolutional	21 x 21 x 128
Max pooling	-
CBAM	10 x 10 x 128
Fully connected	1024
Softmax	131

The operating system of the experimental computer is windows 10, the python version is 3.9.12, the TensorFlow version is 2.9.2, and the GPU is accelerated by using CUDA 11.2. The Numpy, Keras and matplotlib libraries were also invoked. The initial learning rate was given a uniform setting at 0.001 during training, the optimiser used the RMSprop algorithm, and the loss function was categorical_crossentropy. The number of epochs for the three models was also set to the same number of 10 to avoid too many disturbing factors affecting the results of the experiment.

3. Experiment

Figures 8 displays the accuracy and loss rates of the training and validation sets for each of the three network models during training. All three network models exhibit a steadily increasing trend in test and validation accuracy and a gradually decreasing trend in test and validation loss rates. After ten iterations, the three models achieved relatively high accuracy rates for the Fruits-360 test set, which were 98.01%, 99.69% and 98.98%, respectively. Noticeably, the accuracy of the BasicCNN+CBAM_v1 and BasicCNN+CBAM_v2 models reached very high levels in their fifth and sixth iterations and plateaued in the subsequent iterations. Whereas the BasicCNN model only reached the corresponding level in its eighth-time iteration. The accuracy improvement speed of the model with the CBAM joined is significantly better than that of the BasicCNN model without the CBAM. Moreover, as shown by the experimental ablation results in Table 4, the models with the CBAM incorporated achieved an accuracy of 48.98% and 46.73% for the real-world test set, which is a significant jump compared to the BasicCNN model with an accuracy of only 16.67%. Hence including the CBAM attention mechanism in the convolution neural network can give a considerable performance boost to the model.

On the other hand, the data in Table 4 also indicates that the BasicCNN+CBAM_v1 model has a 0.71% and 1.16% accuracy improvement over the BasicCNN+CBAM_v2 model on the Fruit360 test set and the Real-world test set. Secondly, by comparing the loss rate plot in both Figure 9 and Figure 10, there is a rapid reduction in the loss rate of the BasicCNN+CBAM_v1 model, which reached a very optimal value within the second iteration. In other words, performing an attention mechanism before learning features can optimize the network further in feature extraction, such as image curves, edges and contours. The experiment demonstrates that the CBAM before the convolution layer performs better classification than the CBAM after. According to the experiments, we conclude that the flexibility of visual attention is limited by the learned feature map, where the CBAM is behind the conv layers. Placing the CBAM in the initial of the network, we can obtain a better performance, which can be validated on the public dataset and real-world application.

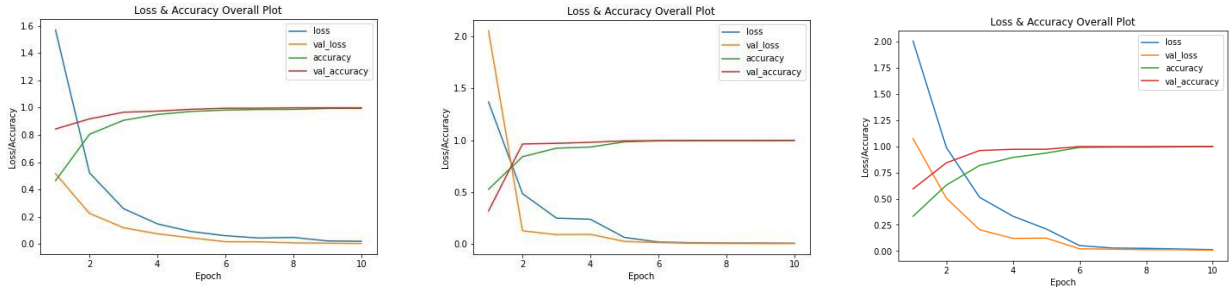

Figure 8. Loss and accuracy curve of all three models.

Table 4. Performance result comparison.

Method	Fruit360 test set	Real-world test set
BasicCNN	98.01%	16.67%
BasicCNN+CBAM_v1	99.69%	48.89%
BasicCNN+CBAM_v2	98.98%	46.73%

4. Conclusion

In this paper, a total of three CNN models are proposed for fruit image classification. The huge accuracy difference between the BasicCNN model without CBAM and the BasicCNN+CBAM_v1 and BasicCNN+CBAM_v2 models with CBAM added to the Real-world test set proves the fact that CNN was effective in enhancing the performance of the fruit image classification task by adding CBAM. Furthermore, the increase of accuracy in both test sets for the BasicCNN+CBAM_v1 model compared to the BasicCNN+CBAM_v2 model also indicates the use of attention mechanism before the convolution layer will enable the model to achieve better performance in image classification. The research outcome may be used in software used to classify fruit or vegetables at supermarket checkouts by including an attention mechanism to the neural network to optimize the accuracy of fruit classification and thus improve customer checkout satisfaction. On the other hand, the network design concepts and results from this paper can also be extended to help improve the performance of network models in areas such as object detection, object localization, and instance segmentation. In future work, the author looks to improve the lack of sample volume by including more varieties and more diverse fruit images in the Real-world dataset. For example, images of unpicked or eaten fruit could be included. These changes are used to optimize the accuracy of the network model further.

References

[1] Marimuthu, S., & Roomi, S. M. M. (2017). Particle swarm optimized fuzzy model for the classification of banana ripeness. IEEE Sensors Journal, 17(15), 4903-4915.

[2] Rocha, A., Hauagge, D. C., Wainer, J., & Goldenstein, S. (2010). Automatic fruit and vegetable classification from images. Computers and Electronics in Agriculture, 70(1), 96-104.

[3] Siddiqi, R. (2020, July). Comparative performance of various deep learning based models in fruit image classification. In Proceedings of the 11th International Conference on Advances in Information Technology (pp. 1-9).

[4] Mureşan, H., & Oltean, M. (2017). Fruit recognition from images using deep learning. arXiv preprint arXiv:1712.00580.

[5] Mnih, V., Heess, N., & Graves, A. (2014). Recurrent models of visual attention. Advances in neural information processing systems, 27.

[6] Yin, W., Schütze, H., Xiang, B., & Zhou, B. (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. Transactions of the Association for Computational Linguistics, 4, 259-272.

[7] Zhong, Z., Zheng, L., Kang, G., Li, S., & Yang, Y. (2020, April). Random erasing data augmentation. In Proceedings of the AAAI conference on artificial intelligence (Vol. 34, No. 07, pp. 13001-13008).

[8] Han, D., Liu, Q., & Fan, W. (2018). A new image classification method using CNN transfer learning and web data augmentation. Expert Systems with Applications, 95, 43-56.

[9] Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. IEEE Transactions on pattern analysis and machine intelligence, 20(11), 1254-1259.

[10] 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).

[11] Park, J., Woo, S., Lee, J. Y., & Kweon, I. S. (2018). Bam: Bottleneck attention module. arXiv preprint arXiv:1807.06514.