

# **ITRI 626 Mini-Project Submission**

**A. du Plessis**

Lecturer: Prof. Abasalom E. Ezuguw

Student number: 34202676

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Architecture of Model</b>	<b>2</b>
2.1	Input Layer . . . . .	3
2.2	Convolution Layer . . . . .	3
2.3	Batch Normalisation Layer . . . . .	3
2.4	Activation Function (Nonlinearity Layer) . . . . .	4
2.5	Pooling Layer . . . . .	4
2.6	Dropout Layer . . . . .	4
2.7	Fully Connected Layer . . . . .	4
2.8	Output Layer . . . . .	5
<b>3</b>	<b>Performance Evaluation</b>	<b>6</b>
3.1	Loss . . . . .	6
3.2	Accuracy . . . . .	7
3.3	F1-Score . . . . .	7
3.4	ROC AUC . . . . .	8
<b>4</b>	<b>Conclusion</b>	<b>9</b>

# List of Figures

2.1	Neural Network Architecture . . . . .	2
2.2	Representation of Input Batch . . . . .	3
2.3	Representation of the Layers . . . . .	5
3.1	Loss per Epoch . . . . .	6
3.2	Accuracy per Epoch . . . . .	7
3.3	F1-Score per Epoch . . . . .	8
3.4	ROC Curve with AUC Score . . . . .	8

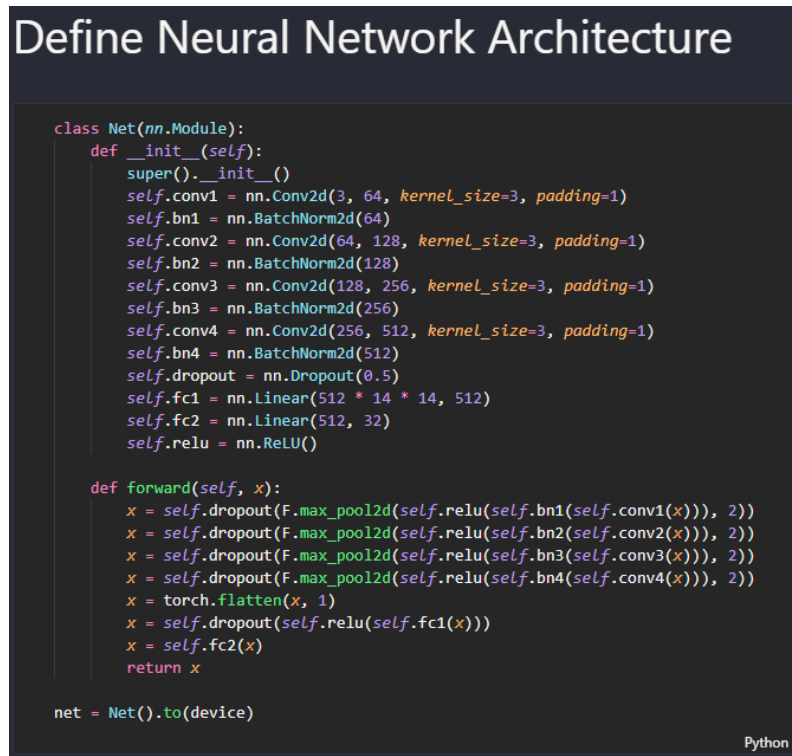
# Introduction

Convolutional Neural Networks (CNNs) are deep neural networks that are often used for image analysis. It can recognise images, classify them, detect objects, and identify faces. CNN comprises neurons with learnable weights and biases. Neurons may process several inputs by creating a point as a product and possibly following it with nonlinearity. The network still expresses a single scoring function, distinguishing between raw picture pixels and class scores. CNN provides computational efficiency through convolution, spatial integration, and parameter sharing. Thus, it enables CNN models to operate on any platform, even mobile devices (Valentino et al., 2021). Being utilised in clinical research for medical image analysis, with a strong emphasis on the clinical aspects of the field (Singha et al., 2021) along with face recognition, action recognition, image classification and natural language processing (Shamsaldin et al., 2019). Deep learning is a recent trend in machine learning and artificial intelligence research (Wang et al., 2020). Many notable advancements have occurred in this sector across the world. Such studies include deep learning that can be used as an automatic system for plant counting (Cenggoro et al., 2018) (Rahutomo et al., 2019).

# Architecture of Model

The model that was developed is a convolutional neural network (CNN). CNNs are based on neurons that are organised in layers, this enables CNNs to hierarchical representations (Kattenborn et al., 2021). The architecture of any CNN, this includes the model that was developed to write up this report, consists out of the following layers as stated by Bhatt et al. (2021):

- Input layer
- Convolution layer
- Batch normalisation layer
- Activation function (Nonlinearity layer)
- Pooling layer
- Dropout layer
- Fully connected layer
- Output layer



```
class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 64, kernel_size=3, padding=1)
        self.bn1 = nn.BatchNorm2d(64)
        self.conv2 = nn.Conv2d(64, 128, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(128)
        self.conv3 = nn.Conv2d(128, 256, kernel_size=3, padding=1)
        self.bn3 = nn.BatchNorm2d(256)
        self.conv4 = nn.Conv2d(256, 512, kernel_size=3, padding=1)
        self.bn4 = nn.BatchNorm2d(512)
        self.dropout = nn.Dropout(0.5)
        self.fc1 = nn.Linear(512 * 14 * 14, 512)
        self.fc2 = nn.Linear(512, 32)
        self.relu = nn.ReLU()

    def forward(self, x):
        x = self.dropout(F.max_pool2d(self.relu(self.bn1(self.conv1(x))), 2))
        x = self.dropout(F.max_pool2d(self.relu(self.bn2(self.conv2(x))), 2))
        x = self.dropout(F.max_pool2d(self.relu(self.bn3(self.conv3(x))), 2))
        x = self.dropout(F.max_pool2d(self.relu(self.bn4(self.conv4(x))), 2))
        x = torch.flatten(x, 1)
        x = self.dropout(self.relu(self.fc1(x)))
        x = self.fc2(x)
        return x

net = Net().to(device)
```

Python

Figure 2.1: Neural Network Architecture

## 2.1 Input Layer

The images of the fruits that the model is trained gets inputted into the model in branches of 32. The images are then resized to 224x224 pixels with the 3 colour channels, RGB (Red Green Blue). Each of the colour channels are normalised by using a mean of [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225].

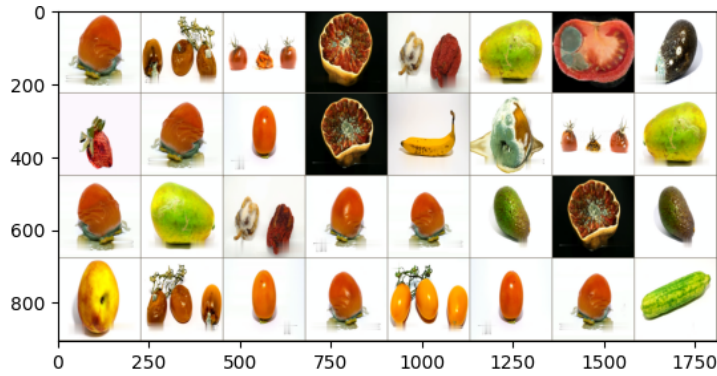


Figure 2.2: Representation of Input Batch

## 2.2 Convolution Layer

This model consists of 4 convolutional layers. Each of the convolutional layers are followed by batch normalisation, activation, pooling, and dropout.

The parameters for each convolutional layer are structured as follows. The input colour channels, the colour channels as the images are entered into the model. The output channels, the new colour channels after the images passes throw a convolutional layer this becomes for the following convolutional layer. The kernel size, the kenel size refers to the dimensions of the sliding window that moves across the input image, the kernel performs element-wise multiplication with the input data it covers, followed by a summation to produce a single output value, this helps in feature extraction, such as detecting edges, textures, or more complex patterns in deeper layers (Ding et al., 2022). The padding, the adding of extra pixels around the border of an image after is has passed throw the convolutional layer.

## 2.3 Batch Normalisation Layer

The batch normalisation helps to standardise and accelerate the training of the model, by normalising the inputs of each of the batches, the batch normalisation is located before the activation function (Kumar and Shankar Hati, 2021).

## 2.4 Activation Function (Nonlinearity Layer)

The activation function that was used in this model is the ReLU activation function. This activation function introduces non-linearity into the model, by enabling the model to learn complex patterns. ReLU demonstrated that an activation function in the hidden layers can improve the training speed of the model (Ide and Kurita, 2017).

## 2.5 Pooling Layer

The purpose of this layer is to down size the convolved feature's spatial size, this helps to reduce the computing power that is needed to process the data (Bhatt et al., 2021). For this model maximum pooling was used in the pooling layer. The input tensor is processed by the pooling layer, where a 2x2 kernel moves across the matrix, selecting the maximum value at each position to populate the output matrix (Bhatt et al., 2021).

## 2.6 Dropout Layer

The dropout layer is a regularisation technique that helps to prevent overfitting by randomly setting a fraction of input units to zero during training (Khan et al., 2019). As stated by Khan et al. (2019) a dropout rate of 0.5 is a standard dropout rate.

## 2.7 Fully Connected Layer

The fully connected layer, also known as the dense layer, is located at the end of all the hidden layers, and allows the model to perform classification. The fully connected layer takes input from the final pooling layer, which is flattened before being passed to it (Bhatt et al., 2021). Flattening transforms the 3D output from the previous layer into a single vector (Bhatt et al., 2021). The FC layer then learns nonlinear combinations of high-level features from the convolutional layer, allowing it to model complex functions in that space (Bhatt et al., 2021).

## 2.8 Output Layer

The output layer is the last layer for a CNN model. The output that the model provides can be between any of the 32 different classes that the dataset exists out of. The activation function is also applied here to determine the probability of the model accurately predicting the output. Based on this predicting the model does backpropagation and changes values to help improve the training process. Since the model is being saved, this makes it possible to apply transfer learning, by using the saved model to train on a new dataset.

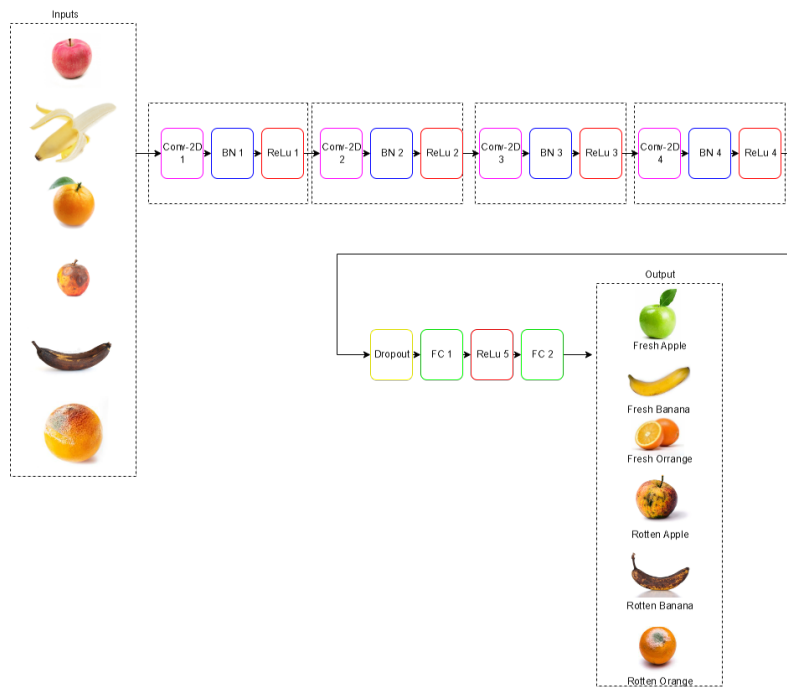


Figure 2.3: Representation of the Layers



# Performance Evaluation

The model that was developed could train for a maximum epoch of 150. A patient of 10 epochs was added to prevent overfitting the model, by stopping if no improvements to the model had been made after 10 epochs. During the training process of the model metrics such as loss per epoch, accuracy per epoch, F1-score per epoch, and the ROC and AUC were collected and saved in their own respective text files. This section will discuss each of these metrics of the model.

## 3.1 Loss

The loss value helps to improve the model during training, by avoiding overfitting. Since the classes are not the same size this can lead to the model overfitting. But the loss value removes features from classes that have more data than the classes that do not have as much data (Pham et al., 2021). During the training process the model started with a Loss of 3.8605431059928477 and at the end of the training process the model had a Loss of 0.04921753687264379. The most desirable final loss value is a loss value of 0, but to prevent the model from overfitting if the loss value for a epoch remain close to the same value for 10 epoch the model would then stopped training. The loss was calculated by dividing the running loss of the epoch by the size of the train dataset. The loss per epoch can be seen in Figure 3.1.

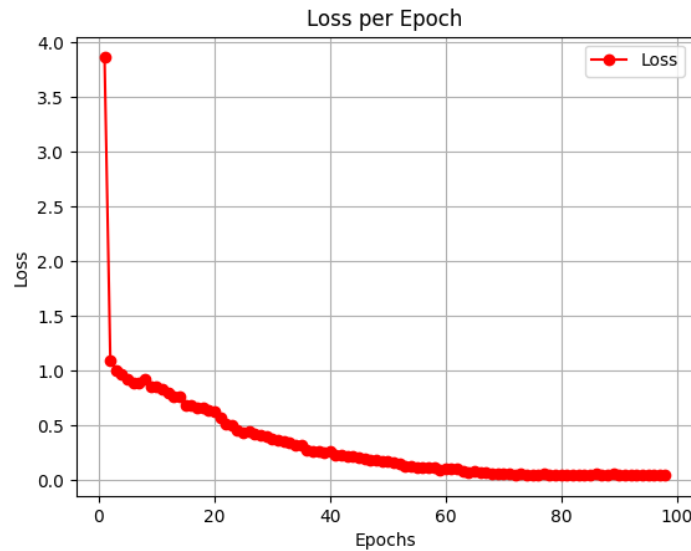


Figure 3.1: Loss per Epoch

## 3.2 Accuracy

The accuracy is a way to determine how accurate the model is during training. In order to calculate the accuracy of the model for each epoch a test dataset is needed, the test dataset for this model consists of 10 images per each class in their respective class. The accuracy that was calculated is the probability that the model correctly classifies the classes for each of the images in the test dataset. The accuracy per epoch can be seen in Figure 3.2.

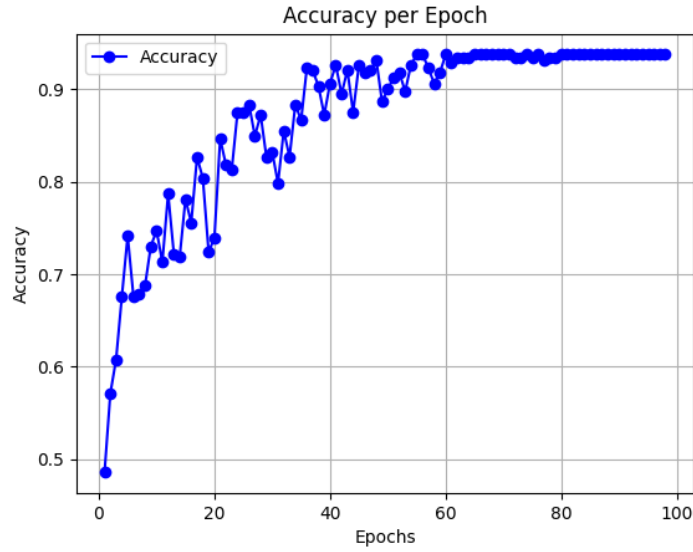


Figure 3.2: Accuracy per Epoch

## 3.3 F1-Score

The F1-score is the harmonic mean of the precision and recall, the value of the F1-score can be any value between 0 and 1 where a F1-score of 1 is seen as being the best score (Humphrey et al., 2022). Since the classes are not evenly distributed a macro averaged F1-score was calculated. The F1-score of the model is the same as the accuracy of the model, the reason for this is that the classes are not even, and since a macro averaged F1-score was calculated the two values are the same. IF the classes were evenly distributed a micro averaged F1-score could have been calculated and the results would then not be the same (Takahashi et al., 2022). The F1-score per epoch can be seen in Figure 3.3.

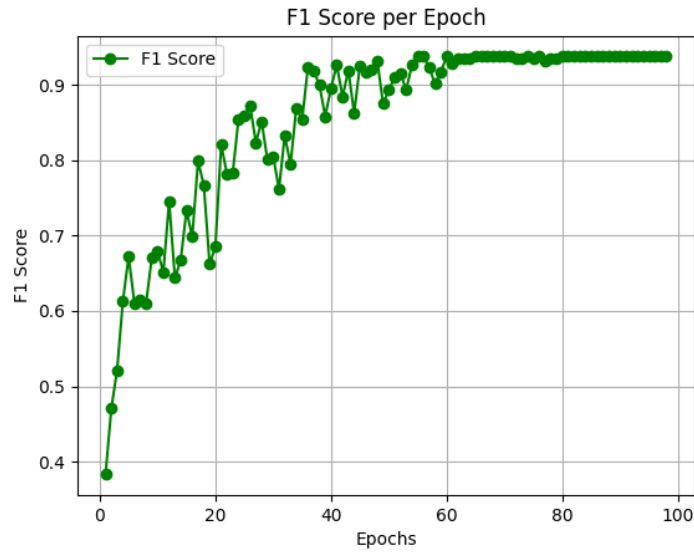


Figure 3.3: F1-Score per Epoch

### 3.4 ROC AUC

To draw a ROC curve the true positive rate and false positive rate for each epoch is needed. The area located under the ROC curve (AUC score) is the most common statistics used in scientific research to assess binary classifications, and can range from 0 (worst result) to 1 (perfect result) (Chicco and Jurman, 2023). The AUC score of this model is 0.8552, this is not a perfect result but it is close to a perfect result. The reason for this result can be related to the prevention of overfitting, if a patient's value was not set in place there could be a possibility that the AUC score would be 1. The ROC curve with AUC score can be seen in Figure 3.4

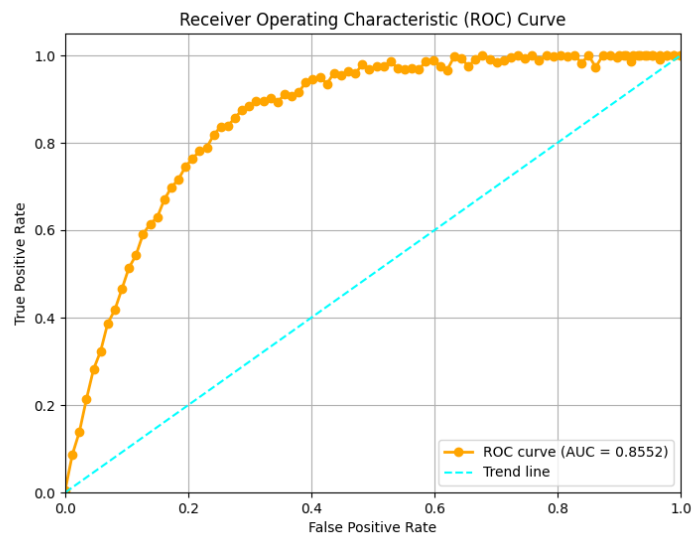


Figure 3.4: ROC Curve with AUC Score

# Conclusion

In this project, a Convolutional Neural Network (CNN) was successfully developed and trained to classify images of fruits into 32 different classes. The architecture of the model followed a standard design with multiple convolutional, batch normalisation, activation, pooling, dropout, and fully connected layers, ensuring that the model could learn complex patterns while minimising overfitting.

The performance evaluation metrics, including loss, accuracy, F1-score, and ROC-AUC, indicate that the model performed well. The loss decreased significantly throughout the training process, and the accuracy and F1-score demonstrated the model's ability to correctly classify the images. Although the AUC score was not perfect, it remained high, showcasing the model's robustness in handling the classification task.

Despite these promising results, the model's performance could be further improved by fine-tuning the architecture or experimenting with more advanced techniques like data augmentation or transfer learning. Future work could also involve deploying this model in real-world applications or adapting it to other image classification tasks.

In conclusion, this project demonstrated the effectiveness of CNNs in image classification and highlighted areas for further optimisation, making it a valuable contribution to the field of machine learning and artificial intelligence.

# Bibliography

- Bhatt, D., Patel, C., Talsania, H., Patel, J., Vaghela, R., Pandya, S., Modi, K., and Ghayvat, H. (2021). Cnn variants for computer vision: History, architecture, application, challenges and future scope. *Electronics*, 10(20):2470.
- Cenggoro, T. W., Budiarto, A., Rahutomo, R., and Pardamean, B. (2018). Information system design for deep learning based plant counting automation. In *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, pages 329–332. IEEE.
- Chicco, D. and Jurman, G. (2023). The matthews correlation coefficient (mcc) should replace the roc auc as the standard metric for assessing binary classification. *BioData Mining*, 16(1):4.
- Ding, X., Zhang, X., Han, J., and Ding, G. (2022). Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11963–11975.
- Humphrey, A., Kuberski, W., Bialek, J., Perrakis, N., Cools, W., Nuytens, N., Elakhrass, H., and Cunha, P. (2022). Machine-learning classification of astronomical sources: estimating f1-score in the absence of ground truth. *Monthly Notices of the Royal Astronomical Society: Letters*, 517(1):L116–L120.
- Ide, H. and Kurita, T. (2017). Improvement of learning for cnn with relu activation by sparse regularization. In *2017 international joint conference on neural networks (IJCNN)*, pages 2684–2691. IEEE.
- Kattenborn, T., Leitloff, J., Schiefer, F., and Hinz, S. (2021). Review on convolutional neural networks (cnn) in vegetation remote sensing. *ISPRS journal of photogrammetry and remote sensing*, 173:24–49.
- Khan, S. H., Hayat, M., and Porikli, F. (2019). Regularization of deep neural networks with spectral dropout. *Neural Networks*, 110:82–90.
- Kumar, P. and Shankar Hati, A. (2021). Convolutional neural network with batch normalisation for fault detection in squirrel cage induction motor. *IET electric power applications*, 15(1):39–50.
- Pham, T.-C., Luong, C.-M., Hoang, V.-D., and Doucet, A. (2021). Ai outperformed every dermatologist in dermoscopic melanoma diagnosis, using an optimized deep-cnn architecture with custom mini-batch logic and loss function. *Scientific Reports*, 11(1):17485.
- Rahutomo, R., Perbangsa, A. S., Lie, Y., Cenggoro, T. W., and Pardamean, B. (2019). Artificial intelligence model implementation in web-based application for pineapple object counting. In *2019 International Conference on Information Management and Technology (ICIMTech)*, volume 1, pages 525–530. IEEE.

- Shamsaldin, A. S., Fattah, P., Rashid, T. A., and Al-Salihi, N. K. (2019). A study of the convolutional neural networks applications. *UKH Journal of Science and Engineering*, 3(2):31–40.
- Singha, A., Thakur, R. S., and Patel, T. (2021). Deep learning applications in medical image analysis. *Biomedical Data Mining for Information Retrieval: Methodologies, Techniques and Applications*, pages 293–350.
- Takahashi, K., Yamamoto, K., Kuchiba, A., and Koyama, T. (2022). Confidence interval for micro-averaged f 1 and macro-averaged f 1 scores. *Applied Intelligence*, 52(5):4961–4972.
- Valentino, F., Cenggoro, T. W., and Pardamean, B. (2021). A design of deep learning experimentation for fruit freshness detection. In *IOP conference series: earth and environmental science*, volume 794, page 012110. IOP Publishing.
- Wang, X., Zhao, Y., and Pourpanah, F. (2020). Recent advances in deep learning. *International Journal of Machine Learning and Cybernetics*, 11:747–750.