



University of Calgary

# Vegetable Classifying Convolutional Neural Network

ENEL 525 Final Project Report

ENEL 525 – Machine Learning for Engineers

Professor Henry Leung & Professor Mohamed Sahnoun

December 15<sup>th</sup>, 2022

Sami Abdelhalem

UCID : 30046859

# Introduction

In this project, a convolutional neural network (CNN) was developed to classify images of vegetables into 15 different classes. The dataset used for this project was sourced from Ahmed et al. (2021)[1] and consists of 21,000 images, with each class containing 1,400 images. The vegetable classes in the dataset include bean, bitter gourd, bottle gourd, brinjal, broccoli, cabbage, capsicum, carrot, cauliflower, cucumber, papaya, potato, pumpkin, radish, and tomato.

Due to limitations in processing power, only 1/5 of the dataset was used for training, validation, and testing. A simple CNN architecture consisting of a single convolutional layer and a single fully connected layer was employed. The model was trained using a subset of the dataset, and the remainder was used for testing and validation. The hyperparameters, such as the learning rate and the optimizer, were experimented with to find the best combination for the problem.

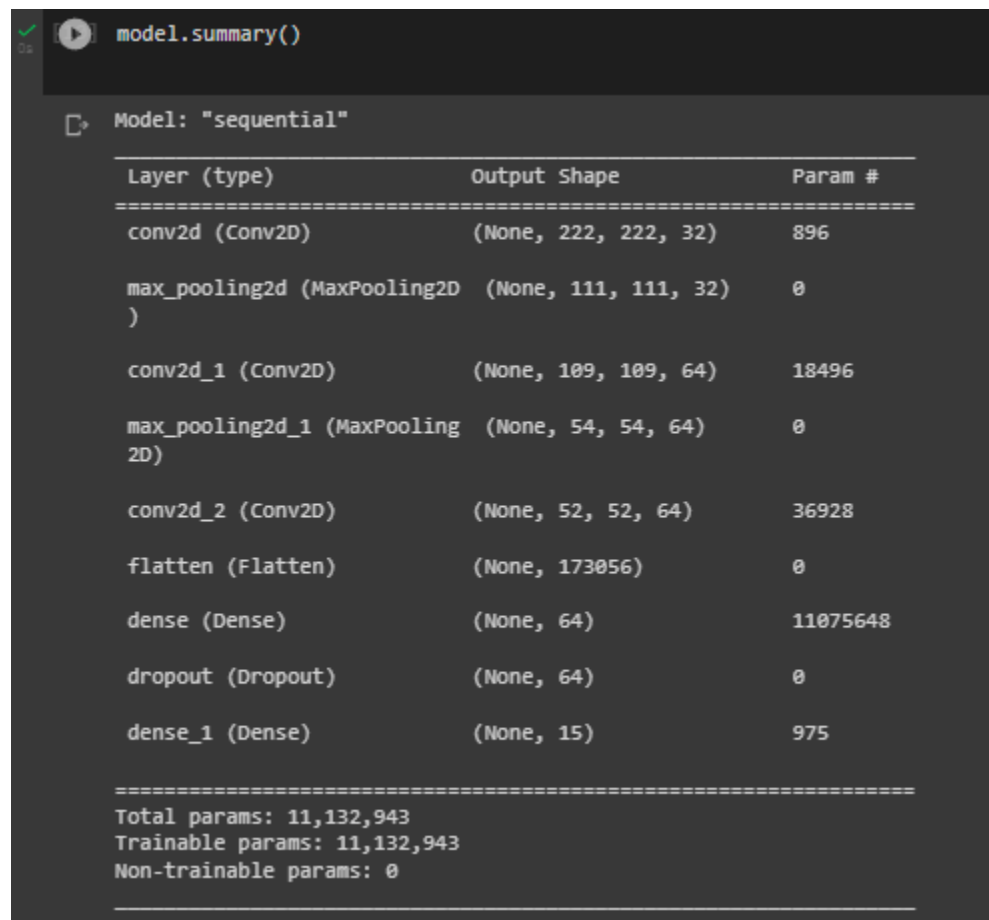
The model was applied to a set of 10 test images, some taken with a camera and others sourced online with creative commons licenses. The results were analyzed, and suggestions were made for improving the model's performance.

# Methodology

In this project, the goal was to create a vegetable classifier using a convolutional neural network (CNN). To accomplish this task, a dataset of vegetable images was used, which was split into training, validation, and test sets.

The network design included three convolutional layers with 32, 64, and 64 filters, respectively, followed by max pooling layers for the first two. The activation function used for all layers was ReLU, and a dropout layer was included in the fully connected part of the network to help prevent overfitting. These design choices allowed the model to learn multiple levels of abstraction from the images, reduce computational complexity, introduce non-linearity, and regularize the model.

Network overview:

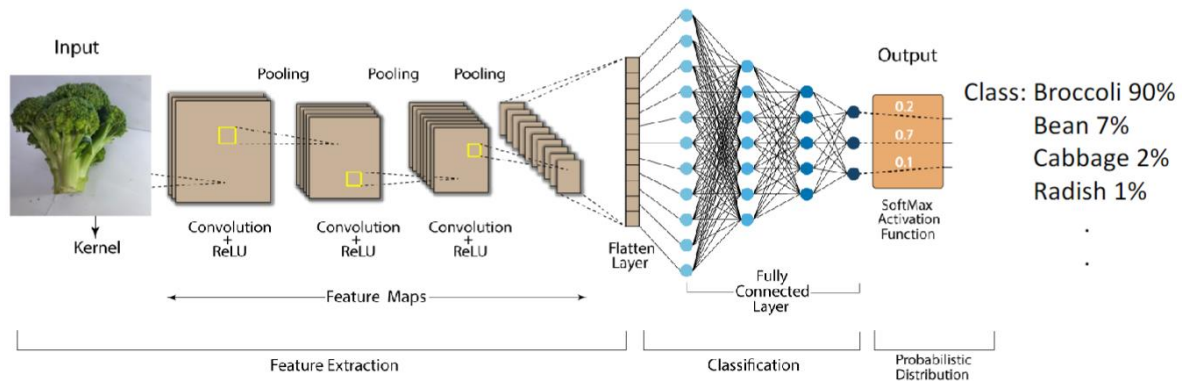


```
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_2 (Conv2D)	(None, 52, 52, 64)	36928
flatten (Flatten)	(None, 173056)	0
dense (Dense)	(None, 64)	11075648
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 15)	975

=====  
Total params: 11,132,943  
Trainable params: 11,132,943  
Non-trainable params: 0  
=====

Network diagram:



For the hyperparameter choices and reasoning, the learning rate was set to 0.0005, which determines the step size at which the optimizer updates the model's weights. A lower learning rate may result in slower convergence but may also reduce the risk of overshooting the minimum of the loss function. The Adam optimizer was used, which is a popular choice due to its ability to adaptively tune the learning rate for each parameter based on the historical gradient information. The number of epochs was set to 20, which refers to the number of times the model sees the entire training set. A higher number of epochs may allow the model to better optimize its weights, but it may also increase the risk of overfitting. These hyperparameter choices helped control the behavior of the model during training and contributed to its overall performance.

To evaluate the performance of the model, the categorical cross-entropy loss function was used, which is defined as:

$$\text{Loss} = -\sum(y * \log(\hat{y}))$$

where

$y$  is the true label

and

$\hat{y}$  is the predicted label.

The model was trained on a fraction (1/5) of the training set and validated on a fraction (1/5) of the validation set. The model's performance was then evaluated with a fraction (1/5) of the test set. A confusion matrix was also produced, showing the model's performance on each class which will be discussed in the results.

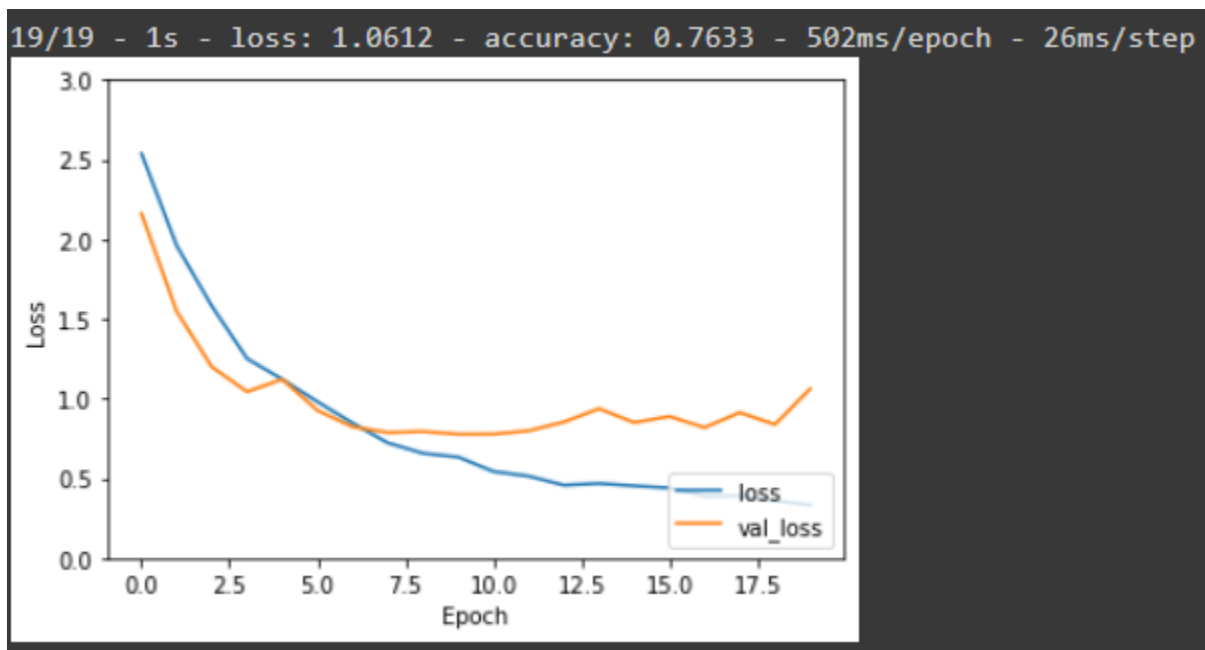
Backpropagation is a common method used in CNNs to optimize the weights of the network and improve the model's performance. It involves calculating the gradient of the loss

function with respect to the weights of the network and updating the weights in the opposite direction of the gradient to minimize the loss. This process is repeated for multiple epochs until the loss reaches a minimum.

## Results and Discussion

The training curve for the model is shown in the chart below, which plots the loss and validation loss over 20 epochs. As can be seen, the loss decreases steadily while the validation loss fluctuates slightly but generally decreases over the training period. This suggests that the model can improve its performance on the training data while also generalizing well to the validation data. The relevant training parameters are summarized in the table below, which shows that a learning rate of 0.0005 and Adam optimizer were used, and the categorical cross-entropy loss function was applied.

Training curve for the model, showing the loss and validation loss over 20 epochs:



Training parameters for the model:

Parameter	Value
Learning rate	0.0005
Optimizer	Adam
Loss function	Categorical cross-entropy
Number of epochs	20

The model used in this project was a convolutional neural network with three convolutional layers, each with a different number of filters to detect different features in the images. Max pooling was used after each convolutional layer to reduce the dimensionality of the feature maps and decrease the computation required. The model also included a dense layer with 64 units and a dropout layer to prevent overfitting. The final dense layer had 15 units, corresponding to the number of vegetable classes in the dataset. The model used the

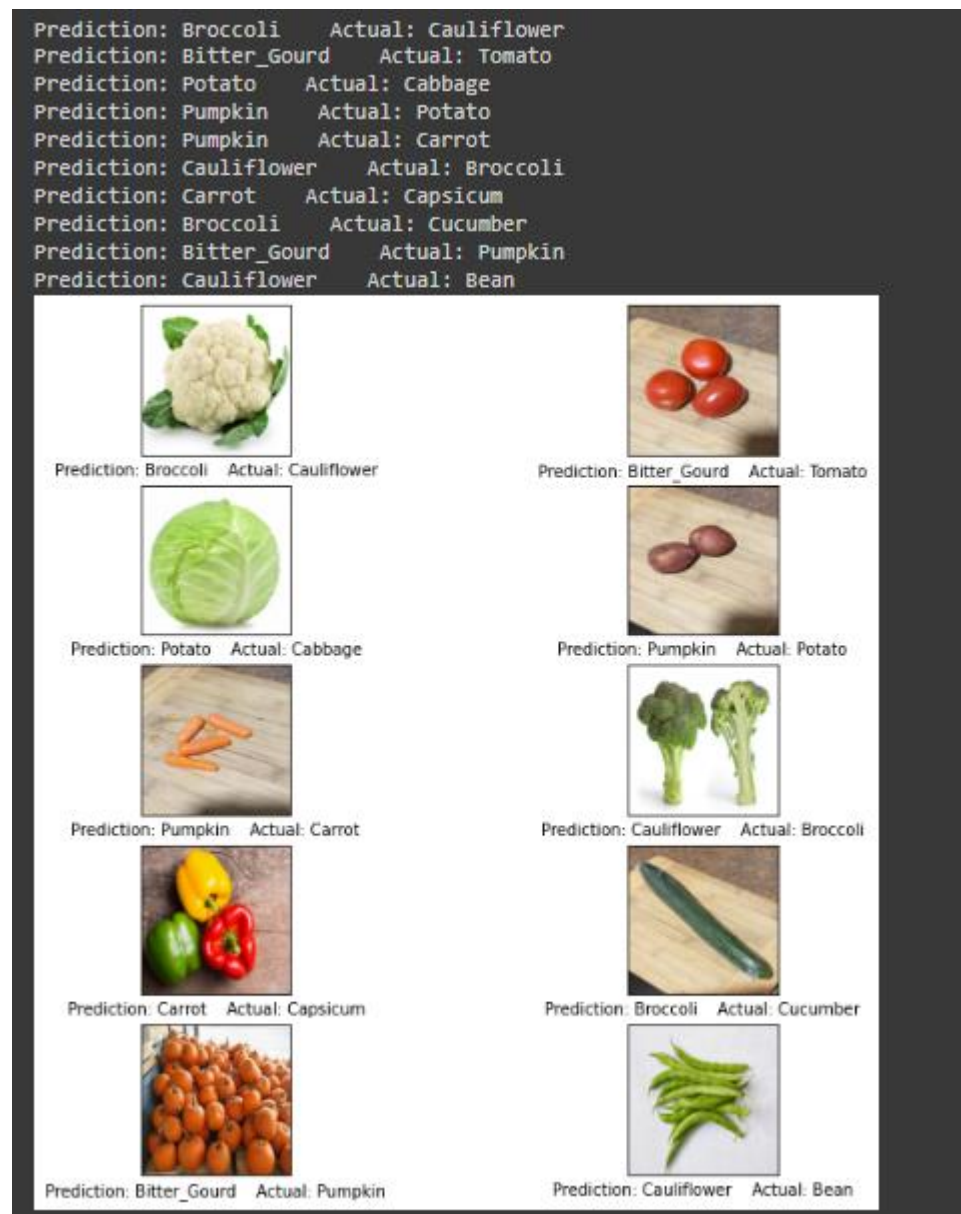
Adam optimizer and a categorical cross-entropy loss function during training. The number of epochs was 20 and the batch size was 32. These hyperparameters were chosen based on their effectiveness in improving model performance in similar image classification tasks.

After training the model, it was tested on the test set and various evaluation metrics were computed. The model achieved an accuracy of 79.5% on the test set, as shown in the confusion matrix below. The confusion matrix visualizes the performance of the model on each class, with the rows representing the true labels and the columns representing the predicted labels.

```
19/19 [=====] - 1s 24ms/step
Confusion matrix:
[[33.  4.  0.  0.  0.  0.  0.  0.  0.  0.  0.  2.  0.  0.  1.]
 [ 4. 34.  0.  0.  0.  0.  1.  0.  1.  0.  0.  0.  0.  0.  0.]
 [ 1.  1. 27.  2.  0.  1.  1.  0.  0.  1.  6.  0.  0.  0.  0.]
 [ 0.  0.  1. 26.  0.  4.  0.  0.  0.  4.  3.  0.  0.  0.  2.]
 [ 2.  8.  0.  0. 26.  0.  0.  0.  1.  0.  0.  0.  2.  0.  1.]
 [ 0.  8.  0.  0.  0. 25.  0.  0.  2.  2.  0.  0.  2.  1.  0.]
 [ 1.  0.  1.  1.  0.  0. 35.  0.  0.  0.  0.  0.  1.  0.  1.]
 [ 0.  0.  0.  0.  0.  0.  0. 40.  0.  0.  0.  0.  0.  0.  0.]
 [ 2.  1.  0.  0.  0.  1.  0.  0. 34.  0.  0.  0.  1.  0.  1.]
 [ 3.  2.  1.  2.  0.  1.  1.  0.  0. 28.  2.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  0.  0. 39.  0.  0.  0.  0.]
 [ 1.  0.  0.  0.  0.  0.  0.  0.  3.  1.  0. 32.  0.  2.  1.]
 [ 0.  0.  0.  0.  2.  2.  0.  1.  0.  0.  0.  0. 34.  1.  0.]
 [ 0.  0.  0.  0.  0.  0.  0.  0.  5.  0.  0.  0.  0. 34.  1.]
 [ 3.  0.  0.  0.  1.  2.  0.  0.  1.  0.  1.  1.  1.  0. 30.]]
Accuracy: 79.5%
```

Overall, the performance of the network was satisfactory, with an accuracy of 79.5% on the test set. The model consisted of 3 convolutional layers, with max pooling applied after each layer to reduce the size of the feature maps. The activation function used in each layer was ReLU, which is known to improve the performance of deep learning models. The model also included a dropout layer to prevent overfitting, as well as a dense layer before the final output layer.

Below is the results of classifying images outside of the dataset provided, the images were a mixture of photos taken personally and some taken from the internet.



There are a few potential reasons for the relatively low accuracy of the model. One issue could be the relatively small size of the dataset, as a larger dataset may have allowed the model to learn more robust features. Additionally, the use of only a fraction of the images in each subdirectory for training and testing may have resulted in a less diverse dataset, potentially limiting the model's ability to generalize to new images. However, the use of image augmentation during training may have partially compensated for this issue. Overall, it is likely that the model would not generalize well to images outside the dataset, as the features learned by the model may not be applicable to new images. It would be beneficial to further evaluate the model's performance better hardware access, as Google Colab has limitations to RAM and



runtime usage. Also, a larger and more diverse dataset would likely better its generalization ability more accurately.

## Conclusion

In conclusion, the task at hand was to classify 15 different types of vegetables using a convolutional neural network. The approach taken was to first preprocess the images by resizing them and converting them to grayscale. The dataset was then split into training, validation, and testing sets. A CNN was designed and implemented using the Tensorflow and various other libraries in Python. The model had 3 convolutional layers, 2 max pooling layers, and 2 dense layers. The model was trained for 20 epochs using an Adam optimizer and a categorical cross-entropy loss function.

In terms of results, the model achieved an accuracy of 79.5% on the testing set, as shown in the confusion matrix. While the model performed well in general, certain vegetables such as tomatoes and broccoli were misclassified more frequently. This may be due to the small size of the dataset and the fact that only a fraction of it was used due to hardware access limitations. With a larger and more diverse dataset and access to better hardware, it is likely that the model would perform even better. However when it came to classifying images that were outside of the dataset it appeared to have struggled significantly. This can be due to many factors, including image lighting, angle, and limited training due to hardware limitations.

Overall, the convolutional neural network was effective in classifying the different types of vegetables. In the future, it would be interesting to explore the use of other network architectures or optimization techniques to further improve the performance of the model. A large portion of this project would not have been completed without the slides and tutorials provided by Dr. Sahnoun and the explanations of the concepts by Dr. Leung.

## References:

1. M. I. Ahmed, "Vegetable image dataset," *Kaggle*, 24-Dec-2021. [Online]. Available: <https://www.kaggle.com/datasets/misrakahmed/vegetable-image-dataset>. [Accessed: 15-Dec-2022].
2. Mohammad Sahnoon. "ENEL 525 \_Project\_Overview," [Online]. Available: <https://d2l.ucalgary.ca/d2l/le/content/471312/viewContent/5626457/View> [Accessed December 15, 2022].