20L-1116 Faiqa Adnan
20L-2156 Mariyam Ali
20L-0921 Aisha M Nawaz
AI PROJECT BSCS 6E

Fruit Recognition From Images Using Deep Learning - Report

Problem Solved

We picked the problem of fruit classification because it is relatively difficult to solve as there are a great variety of intra class forms, colors & textures of fruits available. Due to this complexity there is a shortage of Automated Fruit Classification System for multiple systems. More complex information system of fruit automated detection & classification may be useful in picking the right fruit with correct nutrition. By solving this problem we aim to help in supporting children & people with visual impairment and in development of self-checking supermarkets.

NN Model Used

We used Convolutional Neural Network- CNN, a network architecture for deep learning that assigns weights & biases to different features in an image that helps to differentiate images. The model was chosen as it comparatively requires least amount of preprocessing on data & prevents overfitting of data. Moreover, CNN can learn abstract representations of input & can effectively learn complicated patterns in data so is at the end of complexity & connectivity spectrum.

Our model was created using Keras framework's Sequential API. At the input layer, input was of RGB images with size 100 by 100 and 3 channels. Lambda layer was used that was passed a custom function called convet_to_hsv_and_grayscale. This layer was used to preprocess the input images. On the third layer , we used Conv2D with 16 filters , each of size 5 by 5 , stride of 1,1 and padding was "same" which means the output feature maps will have the same spatial dimensions as the input layer. An activation layer was used with Rectified Linear Unit (ReLU) as the activation function which is applied element-wise to the output of the previous layer. It introduces non-linearity to the model. A Max Pooling layer was used that performs max pooling operation with a pool size and stride of 2x2. It reduces the spatial dimensions of the previous layer's output by selecting the maximum value within each pooling window.

Fourth Layer onwards a common pattern with a stack of Conv2D, MaxPooling2D, activation layers were used. Each convolutional layer increases the number of filters, which helps the model learn more complex patterns in the data. Flatten Layer was used which flattens the output of the last pooling layer into a 1D vector, which is important to connect it to the fully connected layers.

Moreover, Fully connected layer was used with 1024 Neurons and applied the ReLU activation function. This layer learns the higher-level representations of the input features. Dropout layer was used as well. It applies dropout regularization, randomly setting 20% of the input units to 0 at each update during training. It helps prevent overfitting. We continued to add more connected & dropout layers with decreasing number of neurons and dropout layers. At last we used an output layer, Dense. Dense layers are used for classification. There were 6 classes and the

Softmax activation function was used at the last layer.

Our model was compiled using an optimizer ‘Adadelta’ & loss function ‘SparseCategoricalCrossEntropy’. The model’s configurations were changed several times to check impact on accuracy.

Dataset Size & Preprocessing Details

Total 4,264 images were used. 1,068 images (Test Data) & 3,196 images (Train Data). An already preprocessed dataset was provided with the research paper that we used. Different preprocessing, already done on dataset included: convert input RGB to grayscale, keep in RGB colorspace, convert to HSV colorspace, convert to HSV colorspace & grayscale and merge them, apply random hue & saturation changes & random horizontal & vertical flip on images then convert to HSV & grayscale & at last merge them.

Training & Testing Details

75% of the dataset used was split into Training dataset and the remaining 25% into Testing dataset. We further split 25% of the Training data into Validation dataset, in order to combat overfitting. We had 6 classes & shuffling of the dataset was disabled. Learning rate of 0.1 was used. Training progressed over 25 epochs & Batch size 50 was used. Our model was trained using the ‘fit’ method on training & validation datasets. For testing, ‘predict’ method was used to predict class probabilities. These predictions were converted to softmax probabilities using tf.nn. Softmax. Our model’s performance was evaluated on a test dataset & results (testloss & accuracy) were printed.

Comparative Analysis with the Paper

In order to determine the most accurate network configuration for classifying images in our dataset, we tested our model on different configurations provided in the research paper to obtain accuracy. Same dataset & size were used in all. The following table summarizes & compares our results to the research paper:

Configurations #	**Details**	**Findings from Paper**	**Findings from Our Model**
Configuration 1	Convolutional 16 5x5 Convolutional 32 5x5 Convolutional 64 5x5 Convolutional 128 5x5 Fully Connected 1024 Fully Connected 256	Training Accuracy: 100% Testing Accuracy: 98.66%	Training Accuracy: 100% Testing Accuracy: 95.13%
Configuration 2	Convolutional 16 5x5 Convolutional 32 5x5 Convolutional 64 5x5 Convolutional 128 5x5 Fully Connected 1024	Training Accuracy: 100% Testing Accuracy: 98.34%	Training Accuracy: 100% Testing Accuracy: 95.22%

	Fully Connected 256		
Configuration 3	Convolutional 32 5x5 Convolutional 32 5x5 Convolutional 64 5x5 Convolutional 128 5x5 Fully Connected 1024 Fully Connected 256	Training Accuracy: 100% Testing Accuracy: 98.41%	Training Accuracy: 100% Testing Accuracy: 99.53%
Configuration 6 **(MOST ACCURATE)**	Convolutional 16 5x5 Convolutional 32 5x5 Convolutional 32 5x5 Convolutional 128 5x5 Fully Connected 1024 Fully Connected 256	Training Accuracy: 100% Testing Accuracy: 97.92%	Training Accuracy: 100% Testing Accuracy: 99.72%
Configuration 10	Convolutional 16 5x5 Convolutional 32 5x5 Convolutional 64 5x5 Convolutional 128 5x5 Fully Connected 1024 Fully Connected 512	Training Accuracy: 100% Testing Accuracy: 98.25%	Training Accuracy: 100% Testing Accuracy: 99.44%

As summarized by the table above, accuracy achieved by each configuration was very close to accuracy achieved by the model in paper. Results show that our model was successful in classifying fruits in their respective classes. All the tested configurations achieved 100% accuracy on the training dataset whereas accuracy on testing dataset varied slightly but still above 95%. Furthermore the training & validation loss during training was observed, training rapidly improved over the first 5-8 epochs with only minute changes after that.

NEW IMAGE OF LEMON WAS TESTED THAT WAS CLASSIFIED IN CORRECT CLASS WITH 35.21% CONFIDENCE

Key Findings

As we began testing our model on different configurations, we realized some configurations took longer time than others to complete. We also noticed that by decreasing the number of input images we provided the network, our accuracy began to increase. Our model's accuracy was also affected by the amount of variation in classes selected. More accuracy was achieved when classes contained images of fruit that looked very different from one another but when we used classes containing similar looking fruit, our accuracy decreased. From the configurations tested the most accuracy was achieved by configuration 6. This suggests that with some unique configurations, less amount of data, more variation, the model trains better and faster.

References

- **Research Paper:** https://arxiv.org/pdf/1712.00580.pdf
- **Code:**https://github.com/Horea94/Fruit-Images-Dataset/blob/master/src/image_classification/Fruits-360%20CNN.ipynb
- **DataSet:** https://www.kaggle.com/datasets/moltean/fruits