MACHINE LEARNING

(COURSEWORK 2 & 3)

Submitted By: Ammar Kafeel

Date of Submission: 15/03/23

University of Roehampton London

Table of Contents

1. Explanation of the training process:

I used Keras, a high-level neural network API, to build a convolutional neural network (CNN). The goal was to categorize images of apples and tomatoes.

A CNN is a form of neural network that excels at image-processing tasks. It is made up of several layers, including convolutional layers, pooling layers, and fully linked layers. Convolutional layers use filters to extract characteristics such as edges, corners, and forms from an input picture. These filters are trained to recognize meaningful patterns in the input photos. The max pooling layer minimizes the spatial dimensionality of the feature maps, reducing the computation required for the succeeding layers.

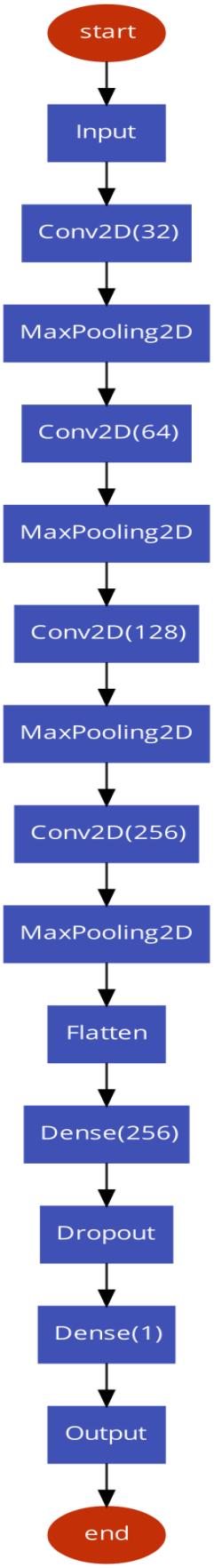
The dense layers, like those seen in a normal neural network, are fully connected layers. They use the previous layers' flattened output and generate a prediction for each class. Dropout is a regularisation strategy that randomly removes certain neurons during training, therefore lowering the network's reliance on any particular input characteristic and preventing overfitting.

The binary cross-entropy loss, which is often employed for binary classification issues, is utilized to assemble the model. The Adam optimizer is employed during training to update the weights, which helps the model converge quicker than standard optimization approaches. I have used accuracy to assess the statistic to determine how successfully my model classifies the images of apples and tomatoes.

I've also implemented an early stopping feature, which is utilized to prevent overfitting in my model by monitoring the validation loss during training. If the validation loss does not improve after a given number of epochs, the training process is terminated to avoid the model from overfitting to the training data.

Lastly, after training, the model I constructed is assessed on the testing dataset to provide an estimate of its performance on new and unknown data. The test accuracy of roughly 90% implies that my model is capable of accurately differentiating between the two classes of apples and tomatoes images.

2. Model Structure:



This flowchart shows the architecture of a convolutional neural network (CNN) with four convolutional layers and two fully connected (dense) layers. Here's a step-by-step analysis of the model's network architecture:

- **Input**: The network is provided a 3-channel image with a size of 224 by 224 pixels.
- **Convolutional layer 1:** The first convolutional layer includes 32 filters, each 3 × 3 pixels in size, and employs the ReLU activation function. The spatial dimensions of the output feature maps are the same as the input, but with 32 channels.
- **Max pooling 1:** The output of the first convolutional layer is subjected to a max pooling operation with a pooling window size of 2 x 2 pixels.
- **Convolutional layer 2:** The second convolutional layer includes 64 filters, each 3 × 3 pixels in size, and employs the ReLU activation function. With 64 channels, the output feature maps have half the spatial dimensions of the preceding layer.
- **Max pooling 2:** The output of the second convolutional layer is subjected to a max pooling operation with a pooling window size of 2 x 2 pixels.
- **Convolutional layer 3:** The third convolutional layer comprises 128 filters with 3 × 3-pixel sizes and uses the ReLU activation function. With 128 channels, the output feature maps have half the spatial dimensions of the preceding layer.
- **Max pooling 3:** The output of the third convolutional layer is subjected to a max pooling operation with a pooling window size of 2 x 2 pixels.
- **Convolutional layer 4:** The fourth convolutional layer comprises 256 filters with 3 x 3-pixel sizes and uses the ReLU activation function. With 256 channels, the output feature maps have half the spatial dimensions of the preceding layer.
- **Max pooling 4:** The output of the fourth convolutional layer is subjected to a max pooling operation with a pooling window size of 2 x 2 pixels.
- **Flatten:** The fourth convolutional layer's output is flattened into a 1D vector, which is then fed into the first fully connected layer.
- **Dense layer 1:** The first dense layer is made up of 256 units and employs the ReLU activation function.
- **Dropout:** To prevent overfitting, a dropout layer is introduced after the initial dense layer. The dropout rate is set to 0.5, implying that 50% of the units are dropped at random during training.

- **Dense layer 2:** The second dense layer is composed of a single unit and employs the sigmoid activation function, which produces a probability score between 0 and 1.
- **Output:** The network's output is a single scalar number that represents the predicted probability that the input image belongs to the positive class (e.g., if the task is binary classification).