CARNEGIE MELLON UNIVERSITY

(CMU AFRICA, KIGALI)

Name: Pierre Ntakirutimana
Course: Applied Computer Vision

Assignment: 4
Date: March 12, 2024

Abstract

This submision serves to explain and answer some questions on assignment 4 of applied computer vision. The assignment has two main parts, first part is classifing images in 10 different classes using Cifar-10 dataset and multiplayer perceprorn(MLP) and basic convolutional neural network. Among these two models, convolutional neural network performed better that MLP model. The second part is to build a model that classify water level by using water bootle images. The model that was build was generalising better than of a well known AlexNets.

1. Image Classification: Cifar-10 Dataset

1.1. Multi-Layer Perceptron Model

The first task for this project was to build the model using the provided mlp architecture. The provided model architecture presents a basic Multi-Layer Perceptron (MLP) configuration tailored for image classification tasks. Illustrated in Figure 1, the model begins with a flattened layer that preprocesses input images, followed by a dense layer comprising 200 neurons.

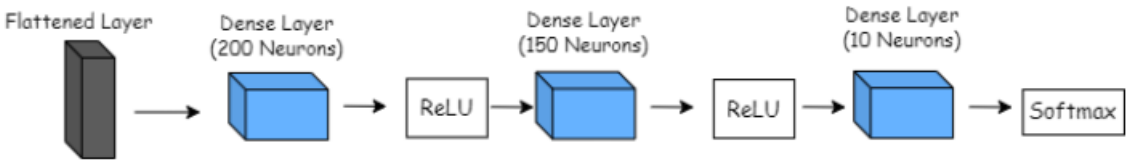

Figure 1: MLP Model Architecture

The outputs from these neurons undergo activation through Rectified Linear Units (ReLU), this is to enhance the ability of the model to capture nonlinear relationships within the data.

Subsequently, the network incorporates another hidden layer consisting of 150 neurons, each connected to a ReLU activation function. The final layer of the MLP has 10 neurons, which is corresponding to the 10 different image classes, and used a softmax activation function to output class probabilities.

The task was to get the data and split them into training and test datasets. The model was trained and we have achieved an accuracy of 46.13% which was in range of required accuracy to achieve.

1.2. Basic CNN Architecture

We also build a network in the architecture in figure 2. Same padding was calculated and added to the architucture.

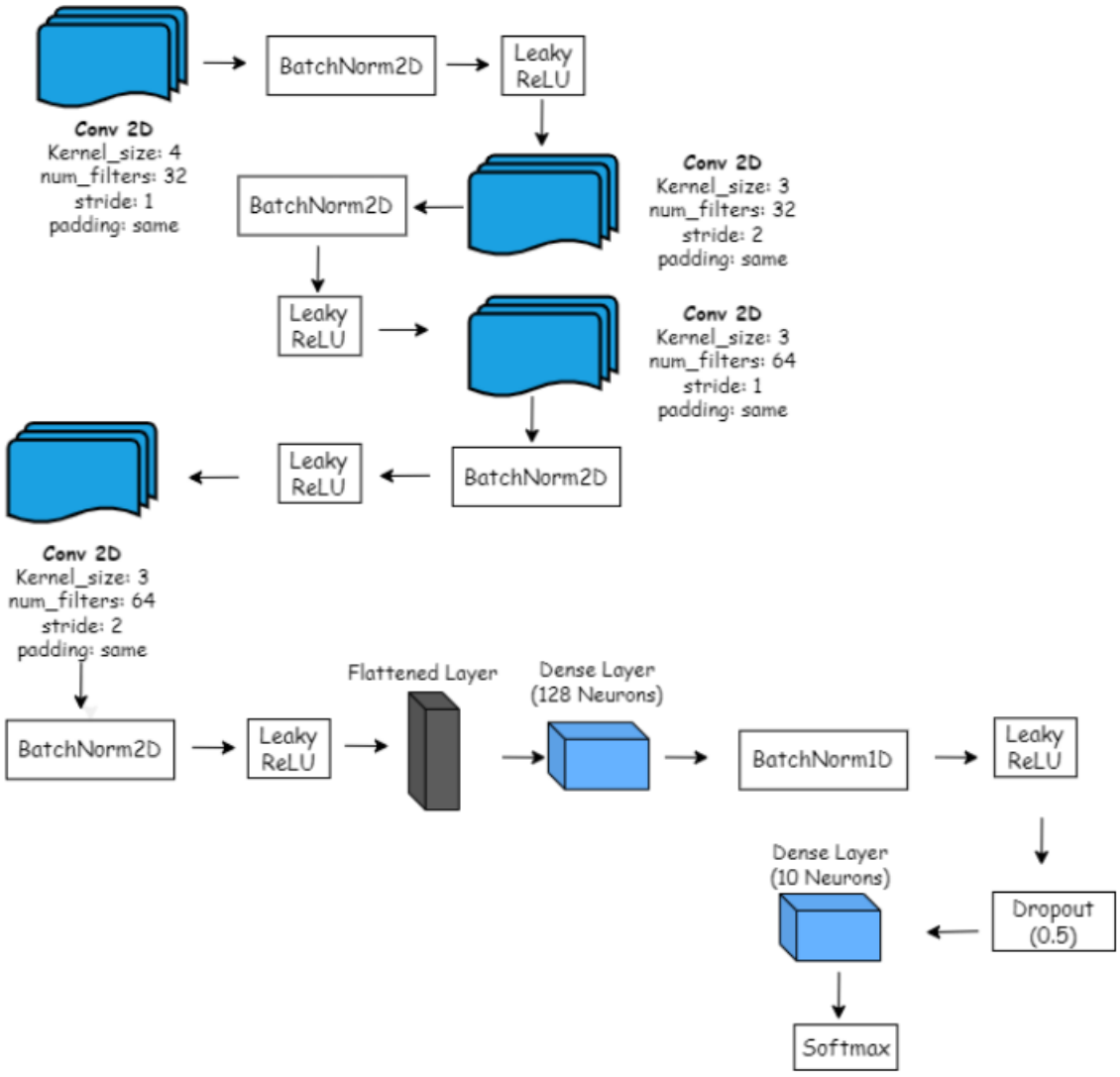

Figure 2: Convolutional Neural Network

We have trained the model on the same dataset with the same splitting and characteristics so that we can compare the results. After training, we have a test accuracy of 72.82% which above

the anticipated minimum.

1.3. CNN architecture perform better than the MLP architecture

CNN architecture performs better than MLP architecture because CNN architecture does not begin with a flattened layer like MLP. The fact that MLP starts by flattening the dataset, makes it consider all the pixels, whether they are important or not. The model fails to capture the spatial information on the images and is not invariant to the rotation, this says that the model struggle to see if images are similar when they are translated. It treats all the numbers as similar, and some of the numbers are not really important in the training of the model. This also makes the dataset look big, and that makes it hard for the model to distinguish between more important and less important features. On the other hand, CNN is better for image classification problems because it has capabilities to understand spatial information on images. It also help in capturing the similarity in the images when they are transformed (flipping or rotation), this reason made CNN model easier to generalise on unseen images as both models we build shows.

1.4. LeakyReLU activation function is preferable to a ReLU activation function

LeakyReLU activation function is preferable to a ReLU activation function as it considers both negative and positive inputs. It is good as it considers negative inputs but penalizes those values by applying 0.1 to them. This approach does not consider all negative inputs as similar and can provide much more information than the ReLu activation function, which considers all values below zero as similar. Using ReLu would have resulted in the fact that some neurons deactivated when during training due to making these neurons zero. LeakyReLu will make sure that all the neurons are working and making impact.

1.5. Batch Normalization in a CNN architecture

Batch normalization is a normalization technique done between the layers of a Neural Network instead of in the raw data1. It is done by training the model on minibatches instead of the full data set1. Batch normalization essentially sets the pixels in all feature maps in a convolution layer to a new mean and a new standard deviation and it serves to speed up training and use higher learning rates, making learning easier. Batch Layer Normalization estimates normalization statistics from the summed inputs to the neurons within a hidden layer across mini-batch and features. Batch normalization can be done to prevent overfitting and make the model generalise better on unseen data by stabilizing training, acting as a regularizer, and allowing for higher learning rates.

1.6. Dropout of 0.5

Dropout helps prevent overfitting by encouraging robustness and diversity among neurons. It is a valuable tool for improving the generalization performance of neural networks. A dropout rate of 0.5 means that, on each training iteration, half of the input units will be randomly dropped out. This helps in preventing the model from relying too heavily on any specific subset of neurons, forcing it to learn more robust and generalize features.

2. Water Level Classification in Water Bottles

In this part, we focused on developing a convolutional neural network (CNN) model for the classification of water levels in water bottles and comparing that model to a well-known architecture. Firstly, we created a CNN model capable of classifying images of water bottles into three different classes: full, half, and overflowing water levels. The comparison aims to evaluate the performance of the newly developed model against a known CNN architecture, providing valuable insights into the effectiveness and efficiency of different models in handling image classification tasks. The dataset used for this study comprises images categorized into the aforementioned classes, representing various states of water levels inside the bottles. These images serve as the foundation for training and testing the CNN models, enabling the classification of water bottle images based on their respective water levels.

2.1. Data Preparation and Exploration

The image datasets were downloaded from canvas as they were provided for the assignment. The dataset was organized into sub-folders according to their labels. The first task was to create a TensorFlow object to organize the dataset and its labels. We have created a tensorflow object with with batchsize of 32 and resized each image to be 255x255 with 3 image chanels. We set shufle to be true so that the images in this object are mixed. The object autoencoded the images by giving them numbers labels instead of alphabetical text labels. ['Full Water level', 'Half water level', 'Overflowing'] were 0, 1, and 2 respectively. Figure **??** is a sample image from the dataset.

Figure 3: Sample Image: Full water

We splited the dataset in the tensorflow object by taking 75% (12 batches) of the data as train-

ing set, 12.5% (2 batches) were used for testing and other 12.5% (2 batches) of the dataset were used for validation purposes. The 2 portions were further processed to change and prefetch as way to improve the efficiency and effectiveness of the training process, ultimately leading to better model performance and training throughput.

We also defined a data augmentation pipeline using a sequential model from TensorFlow. The pipeline begins with resizing and rescaling operations using the 'Resizing' and 'Rescaling' layers, respectively, which standardize the input image dimensions and pixel values. Subsequently, the 'RandomFlip', 'RandomRotation', 'RandomZoom', and 'RandomContrast' layers are sequentially applied to introduce various transformations to the images, such as horizontal and vertical flips, random rotations, zooming, and contrast adjustments. These augmentation techniques enhance the diversity and robustness of the training data, enabling the model to generalize better to unseen data and improve its overall performance during training. We passed our image through the pipeline to augment of images.

2.2. Model Development

The custom CNN model developed for the water level classification task incorporates a series of convolutional layers followed by Leaky ReLU activation functions to capture intricate features from input images effectively. Utilizing the 'padding'same'' argument ensures that the spatial dimensions of the feature maps remain consistent post-convolution, preserving valuable information. Subsequent max-pooling layers with a pool size of (2, 2) downsample the feature maps, facilitating computational efficiency while retaining essential features. Following convolutional and max-pooling layers, a Flatten layer transforms the output into a one-dimensional vector for input into fully connected layers. Two dense layers are then employed for classification, with the first comprising 128 neurons and Leaky ReLU activation, enabling nonlinear transformations of extracted features. The final dense layer consists of 3 neurons with softmax activation, producing output probabilities for the three water level classes: full, half, and overflowing. This model architecture facilitates hierarchical feature extraction and accurate water level predictions.
The model is compiled using the Adam optimizer, which is well-suited for training deep neural networks due to its adaptive learning rate and momentum features. Sparse categorical crossentropy is chosen as the loss function, suitable for multi-class classification tasks where the labels are integers. This loss function calculates the cross-entropy between the probability distribution predicted by the model and the true distribution of the labels. Additionally, accuracy is selected as the metric to monitor during training, providing insight into the model's performance by measuring the proportion of correctly classified samples. By configuring the model with these settings, it is primed for effective training and evaluation on the water level classification task.

The model was trained on a training dataset. The training dataset involves the original provided images and augmented images. At each state, there was training and validation. The model was trained over 100 epochs. The training history is shown in figures 4 and 5.

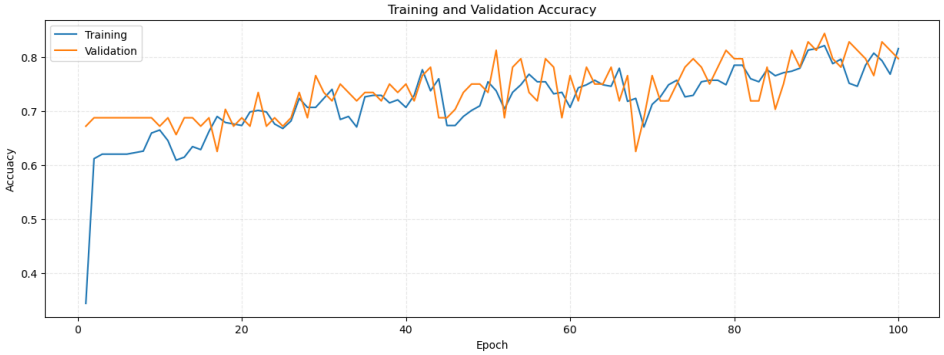

Figure 4: Training and Validation Accuracy

Figure 5: Training and Validation Loss

Looking at the figure 4 and 5, we see that training of the model went well as the validation and training loss as well as accuracy were very close. This helped us to avoid overfitting. The model was tested on test dataset and the accuracy was found to be 87.50% which above the lower threshold of 80%.

We also created an AlexNet model to compare with the one we developed. We trained the model on the same dataset and the same number of epochs, the training of this model took more time as the architecture is much deeper. The traing and validation history is shown in figures 6 and 7 shown below,

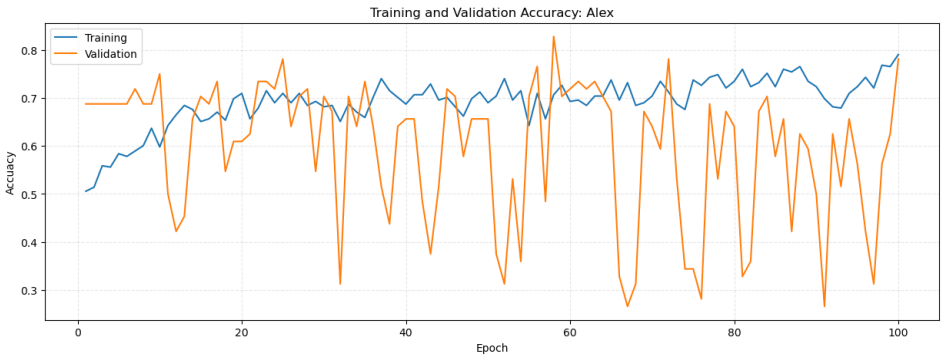

Figure 6: Training and Validation Accuracy:AlexNet

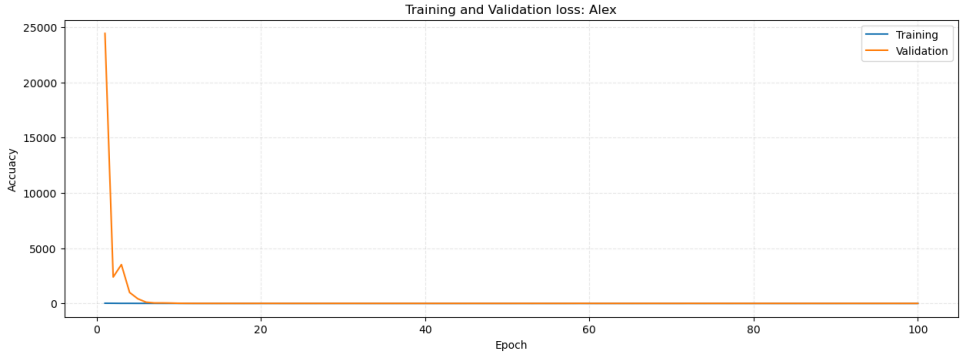

Figure 7: Training and Validation Loss: AlexNet

Training this model shows overfitting at some steps, as shown by figure 6. The training accuracy and validation accuracy are much different. This gives us the intuition to favour our model over AlexNets. The evaluation of the model on test data showed an accuracy of 79.6875% which is below the anticipated minimum accuracy.

We have tested the model on the images from our test dataset to show the application of this model. Figure 8 shows how the model worked on different images and their confidence levels.

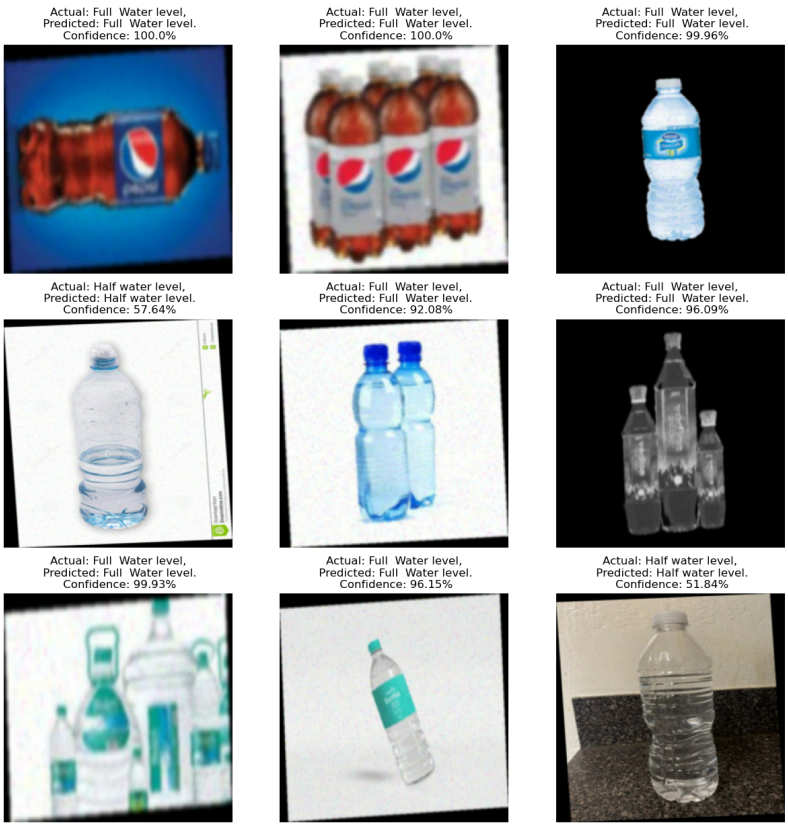

Figure 8: Model Testing

Figure 8 shows that the modal predicted all images correctly with great confidence. The confidence level for 4 images among the 6 images is more than 90% which means that the model did its best to generalise and capture complex relationships in the images.

2.3. Comparison

The comparison of the two models we have built in the above subsection shows that our own model is better than AlexNet, AlexNet took longer to train, around two times the time taken by our own model. The modal also showed overfitting during the training because the network is deep and did not struggle to memorize the training dataset.

The model of ourselves showed a better performance of 87.5% while AlexNet showed 79.6875%. This means that the model we have built also generalizes the water bottle much better than AlexNets.

2.4. Model Application

This model would be helpful in different sectors of life:

- **Water Packaging Plants**: Implementing this model in water-packaging plants can revolutionize the quality control process by accurately measuring water levels in bottles before they are sent to the market. This ensures that customers receive products with precise and consistent water levels, enhancing customer satisfaction and brand reputation.
- **Medicine Delivery**: In telemedicine applications, especially for remote patient monitoring and medication management, the model can play a crucial role in measuring the amount of water needed for patients' medication. By accurately gauging water levels, healthcare providers can ensure precise dosage administration, improving treatment efficacy and patient outcomes.
- **Factory Automated Manufacturing**: Adopting the model in factory automated manufacturing processes can optimize water usage efficiency. By accurately monitoring and controlling water levels in various stages of production, manufacturers can minimize water wastage, reduce operational costs, and contribute to sustainable manufacturing practices. This not only benefits the environment but also enhances operational efficiency and profitability for the manufacturing facilities.