**Deep Learning - Lab 3 IT20657864\_A.D.S.Dodanthenna**

1. Open Google colab. Upload the 1D\_Convolution.ipynb to colab. Run all cells. Based on the result, explain how 1D convolution can be used to identify the edges in an image.

In the context of image processing, including convolutional neural networks (CNNs), 1D convolution can be used to identify edges in an image by applying filters/kernels that are designed to respond strongly to intensity changes. When a filter with coefficients like [-1, 0, 1] is convolved with a 1D signal (row of pixel values), it emphasizes regions where there is a transition from dark to light or light to dark, which corresponds to an edge in the image. The positive and negative coefficients of the filter effectively compute the gradient of the signal and highlight edges where there are sharp changes in intensity.

1. Upload the Image\_Filtering\_(Convolution).ipynb file to colab. Change the filters and see if you can obtain different kinds of edges from the image. Download the modified ipynb file.

**Note:** You may have to copy the lenna.png image to the google drive path mentioned in the file in the notebook file. When you run the notebook for the first time, you may have to click the authorization link and enter the authorization code to the text box displayed.

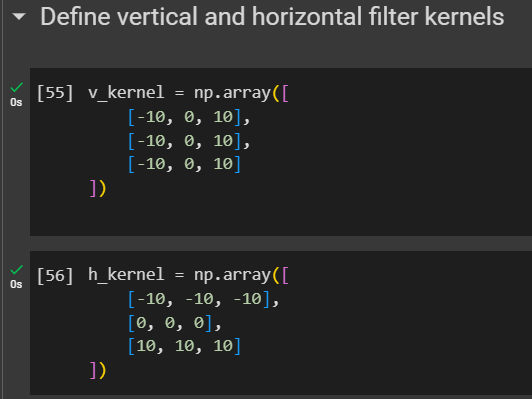
A screenshot of a computer program

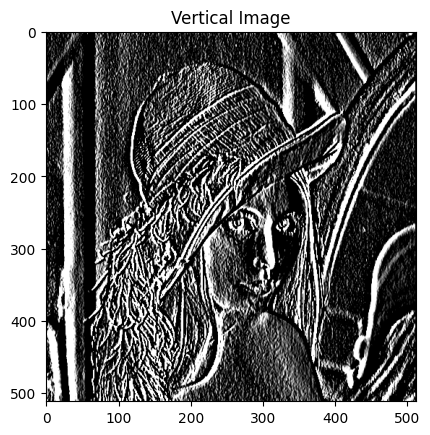
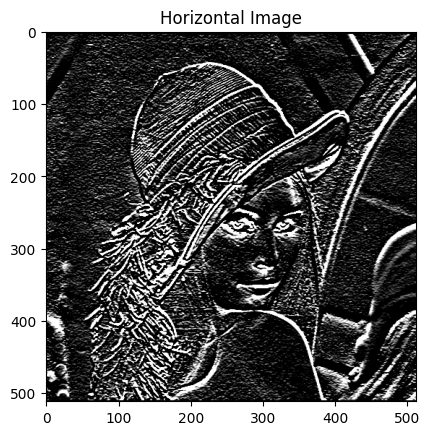
Description automatically generated

A graph of a person

Description automatically generated A close-up of a person wearing a hat

Description automatically generated



A screenshot of a computer program

Description automatically generated

A black and white image of a person

Description automatically generated A close-up of a person wearing a hat

Description automatically generated

1. Upload the CNN\_with\_keras3.ipynb file to colab. Increase the number of epochs to 50.

Why does the validation error increase when the number of epochs are increased? Explain how you can modify the training process to stop that from happening.

Increasing the number of epochs in training can sometimes lead to overfitting. Overfitting occurs when the model learns to fit the training data too closely, including the noise, and fails to generalize well to new, unseen data. This can result in an increase in validation error as the model becomes less robust.

To prevent this, you can employ techniques such as early stopping or regularization. Early stopping involves monitoring the validation error during training and stopping when the error starts to increase. Regularization techniques, like L1 or L2 regularization, add penalties to the loss function based on the magnitude of the model's parameters. These penalties prevent the model from fitting noise and encourage it to learn more general features.

Explain how the mini batch SGD (Stochastic Gradient Descent) algorithm can converge faster than the batch Gradient Descent algorithm.

Stochastic Gradient Descent (SGD) and Batch Gradient Descent (BGD) are optimization algorithms used for training machine learning models.

Batch Gradient Descent (BGD): In BGD, you compute the gradient of the loss function using the entire training dataset and then update the model's parameters. This can be computationally expensive, especially for large datasets, but it provides a more stable estimate of the gradient direction. Mini Batch SGD: Mini Batch SGD lies between BGD and pure Stochastic Gradient Descent. It involves dividing the training dataset into small batches and computing the gradient and updating the parameters based on each batch. This approach strikes a balance between the stability of BGD and the faster convergence of SGD. It can make use of hardware optimizations for matrix operations and can converge faster since updates are made more frequently. Remember, we will need to perform the practical tasks in Google Colab yourself, and I'm here to help answer any questions or provide guidance along the way.