Fundamentals of Image Processing with MATLAB – Assessment 2

Task 1 - Image deblurring by the Wiener filter.

The first task introduces a greyscale image that requires the implementation of the Weiner filter restoration scheme to deblur the image. This is the Mean Squared Error (MSE) optimal stationary linear filter for images degraded by additive noise and blurring. Through Fourier transformations, an image can be broken down into their sine and cosine components, this is utilised in a Weiner filter to restore the greyscale image.

In MATLAB, Weiner filter was implemented with code that loaded the image to corrupt with noise and image motion blur. Therefore, to generate the random noise, a linear operator is created that is convolved with the original image to first blur and then add noise to the blurred image. Subsequently, the Wiener filter equation was emulated into an algorithm in MATLAB analyse the non-iterative process. through emulation of the followed the equation in code form.

There are three parts to produce restored image…

Firstly, each variable was assigned to construct the equation as the Fourier transform of the linear operator. The constant as K has already been initialised where this is found to be estimated noise to signal ratio.

Text

Description automatically generated

Figure Task 1 Initialisation

Secondly, the equation incorporates the numerator with the complex conjugate of H(u, v) and the denominator with the square absolute value of H(u, v) plus the constant, K. As the calculated value is the Fourier transform value of the original image, the inverse transform produces the same image but restored. To demonstrate the process, the procedure would output the plotted figures: original image, motion image and restored image.

Text

Description automatically generated with medium confidence

Figure Weiner Filter Implementation

Based on the results, the algorithm successfully presented the three images with their intended altercation for the image. For example, the motion images had strong noise and blur, whereas the restored images had an exceptional improvement with less noise. It is evident that each figure incorporated a different type of blur and strength of noise.

A collage of a person

Description automatically generated with low confidence

Figure Task 1 Image Outputs

Task 2 - Image deblurring by ISRA

For this task, the MATLAB script deblur.m contains the deblurring scheme with Landweber and Richard Lucy method. To complete the script, the image space reconstruction algorithm (ISRA) method should be implemented. Figure 4 demonstrates the equation for the ISRA was added to the script for the iterations to calculate from 1 to maxiter (1000). This is intended to present semiology graph to display the reading for each deblurring scheme.

Graphical user interface, text, application

Description automatically generated

Figure ISRA implementation

These calculations were output in the command terminal, shown in Figure 5.

Table

Description automatically generated

Figure Task 3 Command Terminal Output

This to presents the original image, corrupt image and Wiener deblur in on figure, along with the Landweber, Richardson-Lucy and ISRA. All seen in Figure 6.

Graphical user interface, Word

Description automatically generated

Figure Task 2 Image Outputs

Finally, a semiology graph is plotted that displayed for each deblurring scheme in Figure 7.

Chart

Description automatically generated with low confidence

Figure Task 2 Semiology Graph

Task 3 - Image filtering in frequency domain.

MATLAB script provides Artificial Neural Network (ANN) is for handwritten digital recognition. This creates the task is to modify hidden layers of the network to generate an accuracy higher than 93%. Convolutional neural networks (CNN) layers are a specific type of neural networks that corporates the data such as images, sound and data that would not still make sense by shifting information such as swapping column. Furthermore, the networks analyse patterns in images using smaller pattern as Kernal. Therefore, the network uses a database of handwritten digits the computer should train itself with existing images and the known answer to help increase accuracy over time.

The assignment has implemented MATLAB code to initiate the number of neurons in each hidden later. In addition, the three layers must not have the requirement sum of each neuron exceed 100. Through testing, the neuron distribution produced the highest accuracy with the most neurons in the hidden layer with 58, whereas the remaining two layers had 44. For the next layer, the neuron number must be half of the previous layer to execute optimally that meant the layers were 28 and 14. It is important to state that the two remaining neurons were not added to any of the layers because this avoided overfitting.

Text

Description automatically generated

Figure MATLAB hidden layers from handwritten\_digit\_recognition\_simple.m

Through the script, the procedure ran neuron distribution with 58-28-14 split on the arbitrary test Kernels for their respected test, displayed in Figure 9. This presents the training progress for the specified architecture where the validation accuracy would be compared to the expected accuracy of 93%.

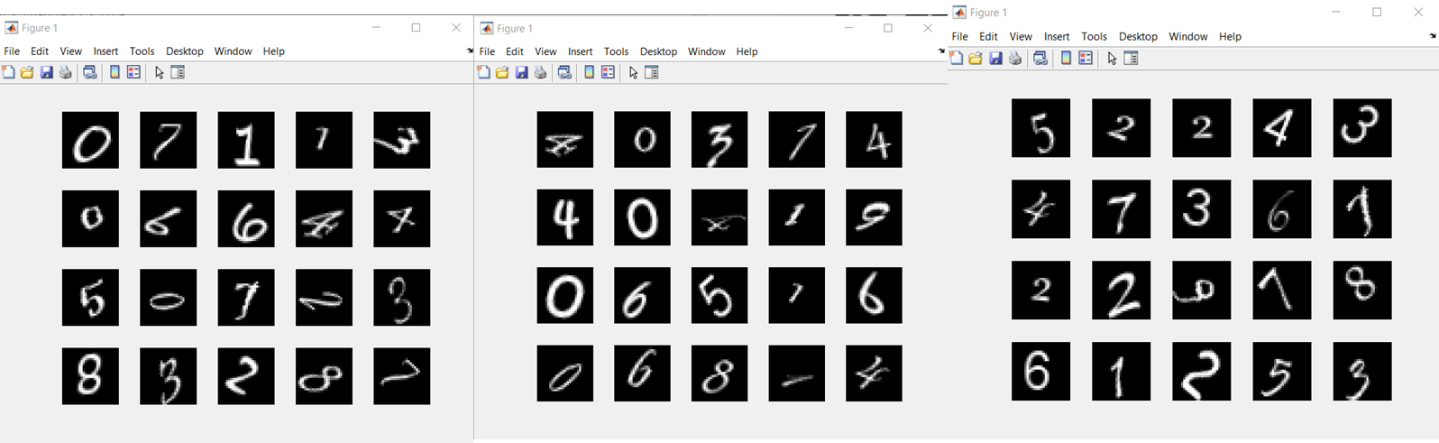


Figure Three test Kernels for the three tests

Test 1

First, the test began with being unsuccessful, in Figure 10, being due to the lower 91.88% validation reading that is lower than the required 93% accuracy.

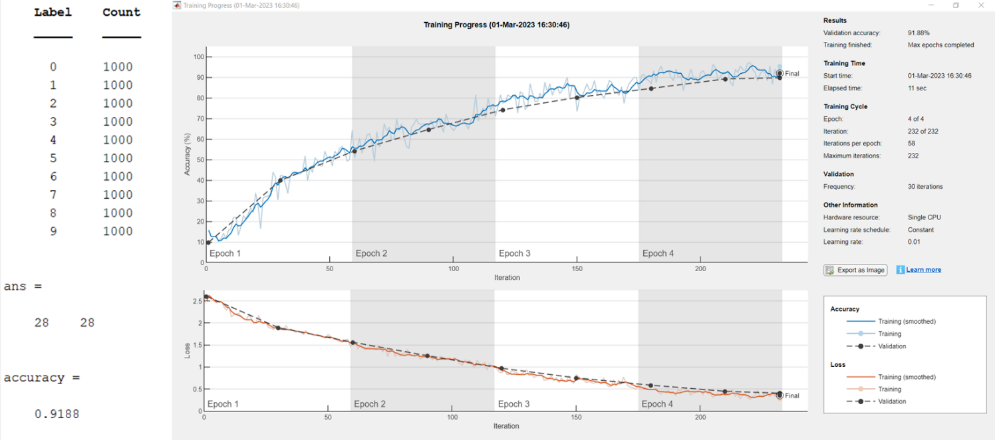


Figure Test 1 Results

Test 2

Figure 10 has a validation accuracy of 92.54% that had a better percentage than the previous task. Even though, this was closer to 93%, this was still an unsuccessful result because this was lower 93% accuracy.

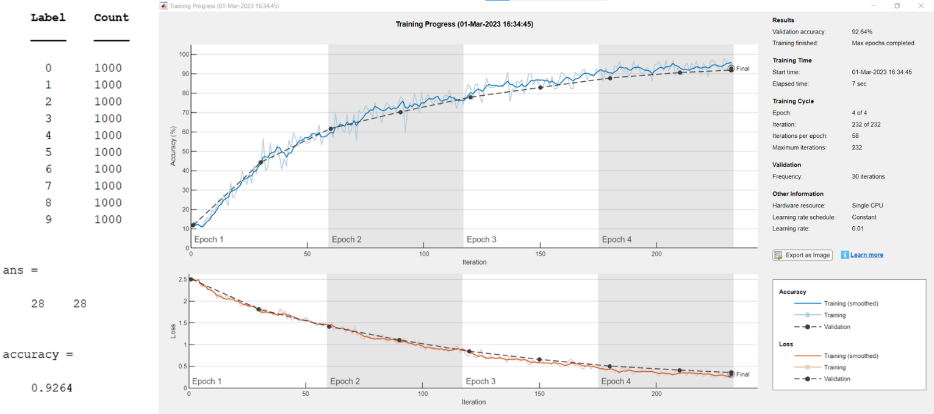


Figure Test 2 Results

Test 3

The final test, in Figure 11, generated the validation accuracy of 93.72% that shows a great improvement because this exceeds the expected value of 93%. Therefore, the hidden layers as 58-28-14 has potential to produce a training progress sufficient in its accuracy.

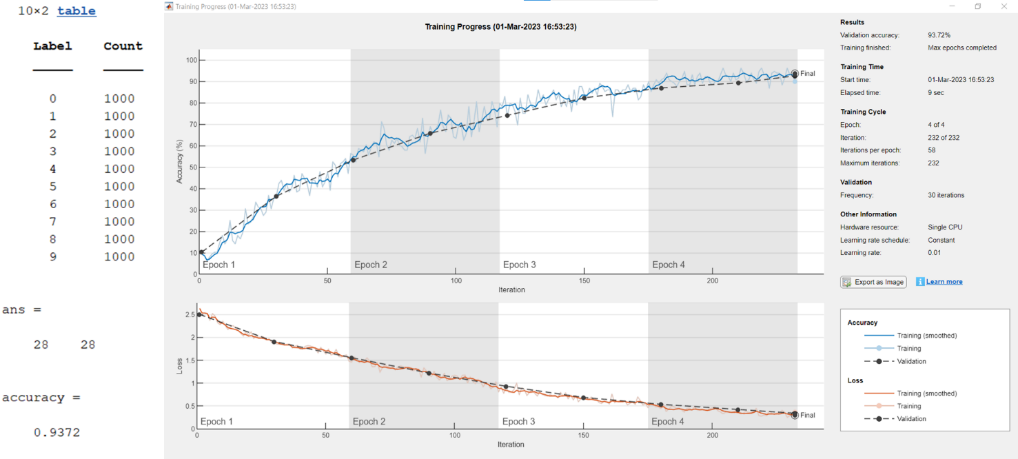


Figure Test 3 Results