# Experiment-2.4

**Aim of the Experiment :** Handwritten text recognition using Artificial Neural Networks.

**Problem Description :**

Handwritten text recognition using Artificial Neural Networks (ANNs) involves training a neural network to interpret and convert handwritten text into machine-readable text. This process typically involves several steps:

1) Data Collection: Gather a dataset of handwritten text samples. This dataset should include a variety of handwriting styles, sizes, and orientations to ensure the model's robustness.

2) Preprocessing: Preprocess the handwritten images to standardize them and make them suitable for input into the neural network. This may involve resizing, normalization, and noise reduction.

3) Feature Extraction: Extract features from the preprocessed images that are relevant for recognizing characters or words. This might involve techniques like edge detection, contour analysis, or feature point extraction.

4) Training: Train the neural network using the preprocessed images and their corresponding labels (i.e., the text they represent). This involves feeding the images through the network, adjusting the network's weights and biases through backpropagation, and optimizing the network's performance using a loss function.

5) Testing: Test the trained model on a separate test dataset to evaluate its performance metrics such as accuracy, precision, recall, and F1 score.

There are different architectures that can be used for handwritten text recognition using ANNs. Some common architectures include:

1) Convolutional Neural Networks (CNNs): CNNs are commonly used for image recognition tasks due to their ability to effectively learn hierarchical features from image data. They have been successfully applied to handwritten text recognition by treating handwritten images as two-dimensional input.

2) Recurrent Neural Networks (RNNs): RNNs are well-suited for sequential data and are commonly used for tasks like speech recognition and language modeling. They can be used for handwritten text recognition by processing sequences of image patches or by treating the handwriting strokes as a sequence of input.

3) Connectionist Temporal Classification (CTC): CTC is a technique used in conjunction with RNNs for sequence labeling tasks where the alignment between the input and output sequences is not known beforehand. CTC has been successfully applied to handwritten text recognition by allowing the model to learn to align the input image with the corresponding text label during training.

4) Transformer-based models: Transformer-based models, such as the Transformer architecture and its variants (e.g., BERT, GPT), have shown remarkable performance in various natural language processing tasks. They can also be adapted for handwritten text recognition by treating the handwriting images as sequences of tokens and applying self-attention mechanisms to capture long-range dependencies.

The choice of architecture depends on factors such as the nature of the handwriting data, the available computational resources, and the specific requirements of the application.

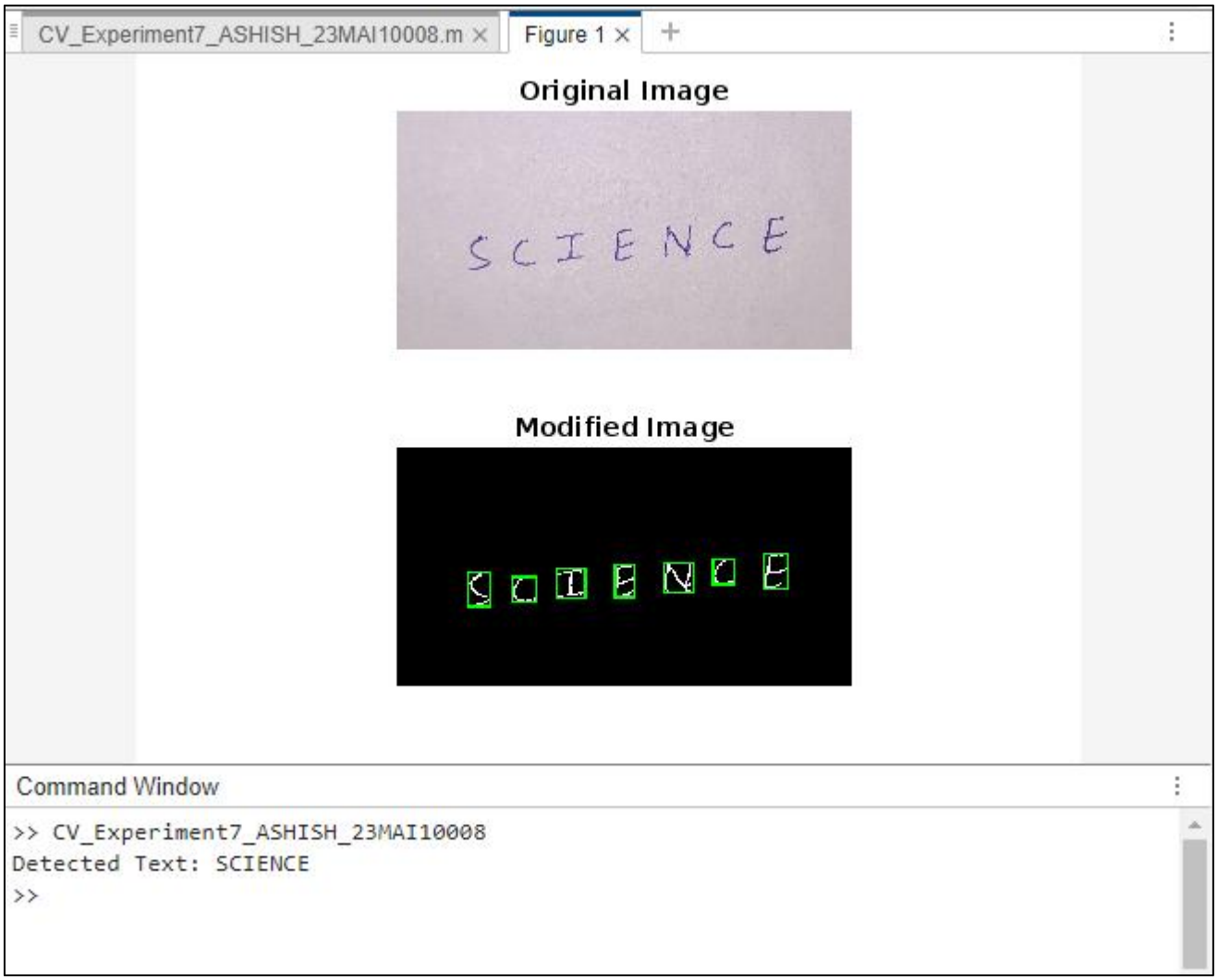
**Code :**

```
%% Handwritten Character Recognition Using Neural Networks

% Load the pretrained network
load("/MATLAB Drive/Computer Vision/Experiment 7/network.mat");

% Input Image
imagePath = fullfile('testImages','Image7.jpg');
originalImage = imread(imagePath);
subplot(2,1,1), imshow(originalImage)
title('Original Image');

% Convert image to grayScale image
grayImage = rgb2gray(originalImage);
% Convert into binary image
threshold = graythresh(grayImage);
binaryImage = ~im2bw(grayImage,threshold);

% Remove all object containing fewer than 200 pixels
moddedImage = bwareaopen(binaryImage,200);
pause(1)
subplot(2,1,2), imshow(moddedImage);
title('Modified Image');

% Labelling connected components
[L,Ne] = bwlabel(moddedImage);

% Measure properties of image regions
propied = regionprops(L,'BoundingBox');
```

```
hold on
% Plot Bounding Box
for n=1:size(propied,1)
	rectangle('Position',propied(n).BoundingBox,'EdgeColor','g','LineWidth',1)
end
hold off
pause (1)

%% Image Segmentation
for n=1:Ne
	[r,c] = find(L==n);
	n1 = moddedImage(min(r):max(r),min(c):max(c));
	n1 = imresize(n1,[128 128]);
	n1 = imgaussfilt(double(n1),1);
	n1 = padarray(imresize(n1,[20 20],'bicubic'),[4 4],0,'both');
	fullFileName = fullfile('segmentedImages', sprintf('image%d.png', n));
	imwrite(n1, fullFileName);
	pause(1)
end

%% Feeding to Neural Network and Detecting Text
for i=1:Ne
	segImage=reshape(double(imread(fullfile('segmentedImages', sprintf('image%d.png',
		i)))) , 784, 1);
	outputMatrix=net(segImage);
	% returns the row number which has highest probability
	row=find(ismember(outputMatrix, max(outputMatrix(:))));
	character = double(imread(fullfile('segmentedImages', sprintf('image%d.png', i))));
	detectedWord(1,i)=imageLabeler(row);
End

%% Displaying Detected Text
fprintf('Detected Text: %s\n',detectedWord)



%% Function for Image Labelling
function detectedWord = imageLabeler(label)
	% This Function returns the character coressponding to label number
	% It is necessary to keep this function in the same directory with the main code
	caps = ['A' 'B' 'C' 'D' 'E' 'F' 'G' 'H' 'I' 'J' 'K' 'L' 'M' 'N' 'O' 'P' 'Q' 'R' 'S'
		'T' 'U' 'V' 'W' 'X' 'Y' 'Z'];
	smalls = ['a' 'b' 'd' 'e' 'f' 'g' 'h' 'n' 'q' 'r' 't'];
	if 0<=label&&label<=9
		detectedWord=label;
	elseif 10<=label&&label<=35
		detectedWord=caps(label-9);
	elseif 36<=label&&label<=46
		detectedWord=smalls(label-35);
	end
end
```

Course Name: Master of Engineering - AIML Course Code: AI-301

**Output :**




**Learning outcomes :**

1. Learnt about the concept of Handwritten Digit Recognition.
2. Learnt about the concept of ANN and its working.
3. Learnt about the architecture of ANN.
4. Learnt about how to visualize text images.
5. Learnt about different layers present in ANN architecture.