What is AlexNet:
AlexNet is a convolutional neural network (CNN) architecture that played a pivotal role in advancing the field of deep learning, particularly in the domain of computer vision and image recognition.

Data collection code:

```
clc
clear all
close all
warning off

% initializes the web camera
c=webcam;

% coordinates and size of the bounding box
x=0;
y=0;
height=200;
width=200;

% creates bounding box with the defined coordinates and size
bboxes=[x y height width];

% initializes a counter variable
temp=0;

% loop that will continuously run as long as temp is less than or equal 300
while temp<=300
    % snapshot an image from web camera
    e=c.snapshot;

    % adds annotation to the image
    IHand = insertObjectAnnotation(e,'rectangle',bboxes,'Processing Area');

    % displays the annotated image in the figure window
    imshow(IHand);

    % generates a filename for the image using the value of temp
    filename=strcat(num2str(temp),'.bmp');

    % crops and stores the image
    es=imcrop(e,bboxes);

    % resizing the cropped image
    es=imresize(es,[227 227]);

    % saves the cropped image
    imwrite(es,filename);

    % counter increment
    temp=temp+1;

    drawnow;
end
clear c;
```

Figure 1: Data Collection

This code is solely responsible for capturing the hand gesture to train the Alexnet to recognise it. The code will capture 300 snapshots from the webcam, crop each snapshot to a specified region, resize it, annotate it with a rectangle, and save it as a BMP file with a unique filename. [1]: The code starts by clearing the Command Window “clc”, closing all figure windows “close all”, clearing all variables from the workspace “clear all”, and turning off MATLAB warnings “warning off”. [7]: Followed by initialising a connection to the webcam using the webcam function. [10-13]: After initialising, it creates a bounding box with a

defined coordinate and size that was defined. The variable “temp” is initialised as a counter variable and is set to 0.

[22] It then enters a loop that will continuously run as long as the counter variable “temp” is less than or equal to 300. Inside the loop, using “**e=c.snapshot**” it will capture an image frame from the webcam and annotate the captured frame with a rectangle to highlight the processing area defined by the bounding box using “**IHand = insertObjectAnnotation(e,'rectangle',bboxes,'Processing Area')**”. The function of the code “**imshow(IHand)**” is to display the annotated image in the figure window. In generating a filename for the image, the filename is created by converting temp to a string **“(num2str(temp))”** and concatenating it with the string **'.bmp'** using **strcat()**. This results in filenames such as "**0.bmp**", "**1.bmp**", and so on, up to "**300.bmp**."

As for the code “**es=imcrop(e,bboxes)**” it crops and stores the captured image to the region of interest specified by the bounding box and resizes the cropped image to the standard size of 227x227 pixels by using “**es=imresize(es,[227 227])**”. The resized and cropped image is saved with the generated filename in BMP format using “**imwrite(es,filename)**”. There is also an increment of 1 in the counter variable temp at the end of each iteration of the loop. The “**drawnow**” function is to command MATLAB to update figures and process any pending callbacks or events in the event queue.

Testing code:

```
clc;
close all;
clear all
warning off

% initializes the web camera
c=webcam;

% loads a pretrained neural network
load myNet1;

% coordinates and size of the bounding box
x=0;
y=0;
height=200;
width=200;

% creates bounding box with the defined coordinates and size
bboxes=[x y height width];

while true
    % snapshot an image from web camera
    e=c.snapshot;

    % annotates the captured frame
    IHand = insertObjectAnnotation(e,'rectangle',bboxes,'Processing Area');

    % crops and resizes the image
    es=imcrop(e,bboxes);
    es=imresize(es,[227 227]);

    % classifies the resized image using the pretrained neural network
    label=classify(myNet1,es);

    % displays the annotated frame
    imshow(IHand);

    % title of the figure window to the classification label
    title(char(label));

    drawnow;
end

```

Figure 2: Testing (Data Collection)

This MATLAB code tests whether AlexNet will recognise the hand gesture that was stored and saved in the previous code. The code continuously captures frames from the laptop camera and annotates them to highlight a specified processing area. It also crops and resizes the frames to a standard size. Additionally, it classifies the cropped region using a pre-trained neural network and displays the annotated frames with classification labels in real time.

Similar to the data collection code, the code starts by clearing the Command Window, closing all figure windows, clearing all variables from the workspace, turning off MATLAB warnings, and initialising the webcam. The code loads a pre-trained neural network model named myNet1 from a file where the hand dataset is stored.

Moreover, it defines a bounding box (‘**bboxes’**) with coordinates (**x, y**) at the top-left corner and dimensions (**height, width**). This bounding box is used to specify a region of interest in the webcam feed. In this case, it is set to (0, 0) with a size of 200x200 pixels.

The code then enters an infinite loop "while true" to continuously capture and process frames from the webcam. Inside the loop, it captures a frame from the webcam, annotates, crops, and resizes the captured image. The "**label = classify(myNet1, es)"** classifies the resized image using the pretrained neural network. **"(imshow(IHand))"** is responsible for displaying the annotated frame with the processing area rectangle. As for the line of code **"(title(char(label)))"**, it sets the title of the displayed frame to the classification label obtained from the neural network, and "**drawnow**" updates the figure window to show the latest frame.

TRAINING:

```
clc
clear all
close all
warning off
g=alexnet;
layers=g.Layers;
layers(23)=fullyConnectedLayer(5);
layers(25)=classificationLayer;
allImages=imageDatastore('hand dataset','IncludeSubfolders',true, 'LabelSource','foldernames');
opts=trainingOptions('sgdm','InitialLearnRate',0.001,'MaxEpochs',20,'MiniBatchSize',64);
myNet1=trainNetwork(allImages,layers,opts);
save myNet1;

```

Figure 3: Training

This code snippet employs transfer learning to fine-tune a pre-trained convolutional neural network (CNN), specifically the AlexNet architecture, for a custom image classification task. Initially, it clears the MATLAB workspace, closes any open figures, and suppresses warning messages to ensure a clean execution environment. The pre-trained AlexNet model is loaded, and its architecture is modified by replacing the last fully connected layer and classification layer to suit the new classification task, which involves five output classes. Subsequently, image data for training is prepared using an **imageDatastore**, which collects images from a specified directory, organises them into classes based on subfolders, and assigns labels accordingly. Training options for the stochastic gradient descent with momentum (SGDM) optimiser are set, specifying parameters such as initial learning rate, maximum epochs, and mini-batch size. The network is then trained using the provided image data and training options, adjusting its parameters to minimise classification error. Finally, the trained network

is saved to a MAT file for future use. This code illustrates the process of leveraging transfer learning with a pre-trained CNN model to adapt it to a new image classification task efficiently, benefiting from the learned features of the base model while customising the output layers to suit the application's specific requirements.

Code 4:

```
% main Code
clc;
close all;
clear all;
warning off;

% Initialize Raspberry Pi connection and camera module
rpi = raspi();
cao = cameraboard(rpi, 'Resolution', '640x480'); % Adjust resolution as needed
load('myNet1.mat'); % Load trained network

gestures = {'MR1', 'ML1', 'HR1', 'HL1'}; % List of gestures in your dataset

while true
    % Capture an image from the webcam
    e = cao.snapshot;

    % Resize the image for classification
    es = imresize(e, [227 227]);

    % Classify the resized image using the trained model
    label = classify(myNet1, es);

    % Display the image with gesture recognition result
    imshow(e);
    title(char(label));
    drawnow;

    % Check if the recognized gesture is in the dataset
    if ismember(label, gestures)
        % Take a snapshot
        snapshot = e;

        % Apply a filter to the captured image (e.g., convert to grayscale)
        grey_frame = rgb2gray(snapshot);

        % Apply bilateral filtering to the grayscale image
```

```
        filtered_image = imbilatfilt(grey_frame);

        % Display the filtered image
        figure;
        subplot(1, 2, 1);
        imshow(snapshot);
        title('Original Snapshot');

        subplot(1, 2, 2);
        imshow(filtered_image);
        title('Filtered Image');
        drawnow;

        % Pause to prevent continuous snapshots for the same gesture
        pause(2);
    end
end
```