**Name:- KARAN TARIYAL Roll Number:- M22AI563 Subject:- CSL7020**

Machine Learning - I

Fractal 3 Assignment

**INSTRUCTOR: Dr. Anand Mishra TOPIC: Fractal 3 Assignment**

**Problem 2: Learning to implement Neural Network**

**INTRODUCTION:**

The objective of this project is to classify Gurmukhi handwritten digits using a Neural Network. The dataset used in this project consists of images of handwritten digits from 0 to 9, written in the Gurmukhi script.

**The git link for the entire code is here =>** https://github.com/tariyalkaran/Machine\_Learning\_Fractal3\_Assignment/blob/main/2\_Learning\_to\_implement\_NN.ipynb

**METHODOLOGY:**

1. **Implementation of Neural Networks from Scratch:**

* The above problem statement was first implemented using Neural Networks Using Gradient Descent Algorithm from scratch.
* The Losses and accuracies of the model are plot and observed.
* The model is tested on test images and its predicting nature is observed.

1. **Implementation of Neural Networks Using Keras:**

* The above problem statement was first implemented using Neural Networks from Scratch Using Gradient Descent Algorithm.
* The Losses and accuracies of the model are plot and observed.
* The model is tested on test images and its predicting nature is observed.

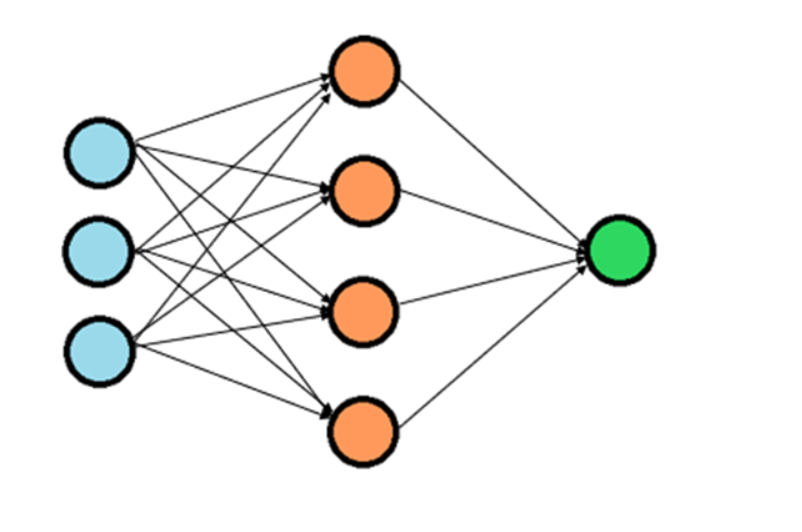
**DATA PREPARATION:**

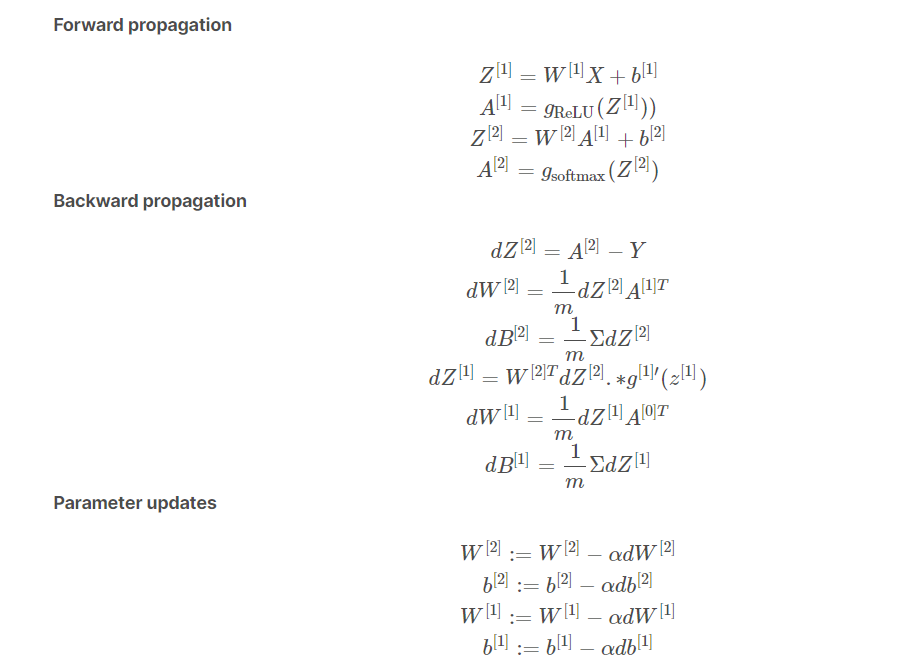
* The data were uploaded from Google drive and were fetched.
* These were stored in a train and valid set.
* To perform these operations a class named **Image\_data()** was created which has various methods which performs various functions:

1. **\_\_init\_\_(self, root\_path)**: Initializes the root path and train/test data variables.
2. **image\_to\_array(self, path)**: Reads an image file and returns its array representation, flattened.
3. **data\_shuffle(self, x, y)**: Shuffles elements in lists **x** and **y** randomly and returns a list of paired elements in the new randomized order.
4. **data\_processing(self)**: Reads and processes image data from a root directory path. It reads images from train and test folders, converts them to array format, shuffles the training data, and returns the processed data as train/test feature and target arrays.

* An instance is created for the class and the image path is passed .
* The train directory contains the training data, which is used to train the neural network, and the test directory contains the test data, which is used to evaluate the performance of the neural network.
* Each directory contains ten subdirectories, one for each class of digit. Each image in the dataset is a grayscale image of size 32x32.
* The images were read in using the PIL library and converted to a numpy array.
* The pixels of each image were then flattened into a one-dimensional array.
* This resulted in a two-dimensional array where each row represents an image and each column represents a pixel.

**NEURAL NETWORK IMPLEMENTATION USING GRADIENT DESCENT ALGORITHM FROM SCRATCH:**

****

****

Gradient descent with forward and backward propagation is an optimization algorithm used in machine learning for training artificial neural networks. The main steps involved in this algorithm are:

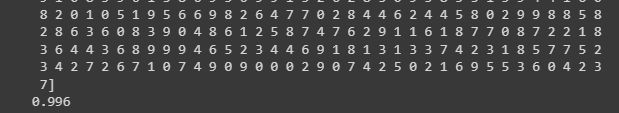
1. Initialize the parameters: First, the weights and biases of the neural network are initialized with random values.
2. Forward propagation: In the forward propagation step, the input data is passed through the neural network to generate a prediction. This involves computing the weighted sum of the input data and biases at each layer, applying the activation function (e.g. ReLU) to the result, and passing it to the next layer. This process is repeated until the output layer is reached, and the final prediction is generated using the softmax activation function.
3. Compute the loss: The difference between the predicted output and the actual output (i.e. the labels of the input data) is computed as the loss. This loss function measures the quality of the neural network's prediction.
4. Backward propagation: In the backward propagation step, the error is propagated back through the neural network in order to update the weights and biases. The partial derivative of the loss function with respect to each weight and bias is computed using the chain rule of calculus. The weights and biases are then adjusted in the opposite direction of the gradient (i.e. where the loss is decreasing) by a certain learning rate (alpha) to minimize the loss function.
5. Update the parameters: The weights and biases are updated using the computed partial derivatives and the learning rate.
6. Repeat: Steps 2-5 are repeated for multiple epochs (i.e. iterations), until the loss function converges to a minimum and the neural network produces accurate predictions.

**TRAINING :**

* The function gradient\_descent() is applied on the training data X\_train and Y\_train with a learning rate of 0.10 and 1000 iterations.
* This function initializes the weights and biases, performs forward propagation, computes the loss, performs backpropagation, updates the weights and biases using gradient descent, and returns the updated parameters and losses over iterations.
* The updated parameters W1, b1, W2, b2 and losses losses are then stored in variables.
* This step is a crucial part of the neural network training process as it adjusts the weights and biases of the network to minimize the loss function, which leads to better prediction accuracy on unseen data.

**ACCURACY OF THE NN MODEL:**

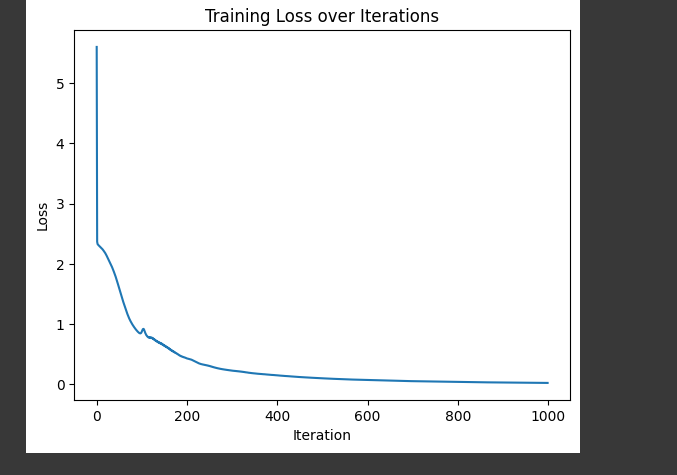
* The accuracy of the model obtained by implementing neural network from scratch is

****

**ACCURACY: 99.6%**

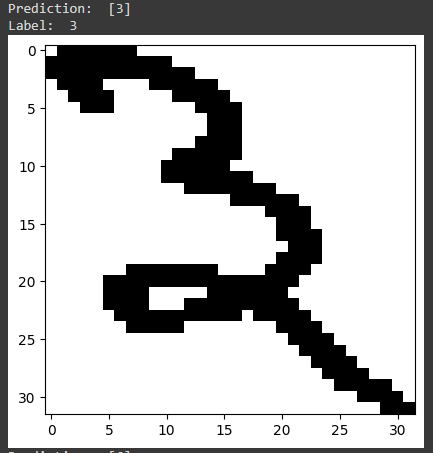
**PLOTTING TRAINING LOSS:**

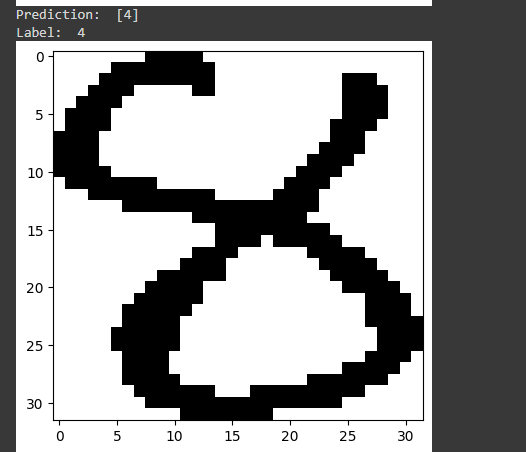
* The training losses were plotted and the loss gradually decreases accounting for good accuracy.

****

**PREDICTIONS:**

* The above developed model is made to predict the labels for some test images and the accuracy of the test dataset is calculated .





**ACCURACY OF TEST DATASET:**

**Accuracy of the Test Dataset:94%**

**RESULTS:**

* The network was able to achieve a high level of accuracy on the test data, which indicates that it was able to generalise well to unseen data.
* The predicted and actual labels are correct.

**MODEL IMPLEMENTATION USING KERAS:**

* The model was also implemented using Keras Library and obtained accuracies :
* Training set Accuracy:94.6%
* Validation Accuracy: 92.13%

**CONCLUSION:**

The project is focused on using a neural network to classify Gurmukhi handwritten digits with high accuracy. The results achieved by the neural network on the test data indicate that it can be used in practical applications. Furthermore, the project can be extended by incorporating more advanced neural network architectures and optimization algorithms to further improve the accuracy of the network. By doing so, the neural network can be utilized in a wider range of applications, such as digit recognition in real-world scenarios.

**PROBLEM 3: CLASSIFICATION-ARCHITECTURE-FOR CLASSIFYING CHART IMAGES**

**INTRODUCTION:**

* The purpose of this project is to develop a machine learning model that can accurately identify the type of chart from an image using a **convolutional neural network and pretrained model ‘Alexnet’**
* The dataset includes various types of charts such as line charts, bar charts, and scatter plots.
* The main objective is to train a model that can effectively distinguish between these different chart types and provide accurate classifications.
* This can have numerous practical applications, such as automated data analysis and report generation.
* Further advancements can be made by using more advanced neural network architectures and optimizing the model to achieve higher accuracy levels.

**Task 1: Download the dataset from drive link given below and split them into Training and Validation Sets:**

**The git link for the entire code is here =>** https://github.com/tariyalkaran/Machine\_Learning\_Fractal3\_Assignment/blob/main/3\_Chart\_Image\_Classification\_using\_CNN.ipynb

**DATA PREPARATION:**

* The dataset provided was uploaded from the Google drive unzipped and were stored in separate directories .
* The **image\_dataset\_from\_directory** function is used to create a TensorFlow dataset object from a directory containing image files. It automatically reads the images from the directory, applies any necessary preprocessing such as resizing or data augmentation, and creates batches of images that can be used to train a neural network

BATCH\_SIZE = 32  
IMAGE\_SIZE = 128  
CHANNELS=3  
EPOCHS=50

* By this function the training and testing sets are separated with its corresponding labels.

**SPLITTING INTO TRAIN AND VALIDATION SETS:**

* The input dataset is converted into training and validation sets with ratio 80% and 20% correspondingly.

**DATA AUGMENTATION:**

* Data augmentation is a technique used in machine learning and deep learning to artificially increase the size of the training dataset.
* It involves generating new training data from the existing data by applying various transformations like flipping, rotating, cropping, or changing the brightness and contrast of the images.
* This technique helps to increase the robustness and generalization of the trained model by exposing it to a more diverse set of training examples.
* Data augmentation techniques are applied to the training set and maps the testing set to an empty data augmentation pipeline.

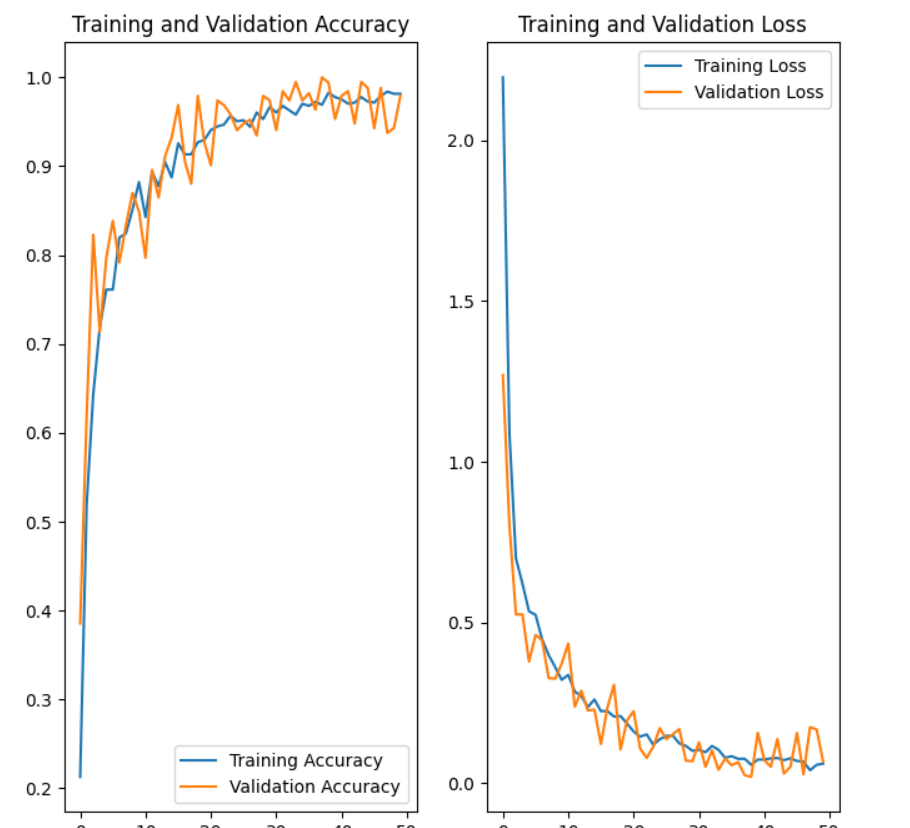
**TASK 2: IMPLEMENT A TWO-LAYER CONVOLUTIONAL NEURAL NETWORK**

**BUILDING THE CNN MODEL:**

* A CNN model with two convolutional layers, a fully connected layer, and an output layer with a softmax activation function was built.
* The model is compiled using the adam optimizer, sparse categorical cross-entropy loss, and accuracy as a metric.
* Finally, the model is trained on the training dataset with batch size, validation dataset, epochs, and verbose set.

**Task 2.1 Calculate Accuracy and loss and plot the obtained accuracy and loss.**

* The losses and the accuracies were plotted for the training and validation sets.



**From the above plot the ACCURACIES increases and the losses decreases as model is complied.**

**Model Evaluation:**

The trained model was evaluated on the validation set, which was not used during the training process. The evaluation was done using the evaluate method of the Keras API, which calculated the loss and accuracy of the model on the validation set

* The model built using CNN performs well.
* The overall accuracy of model is :

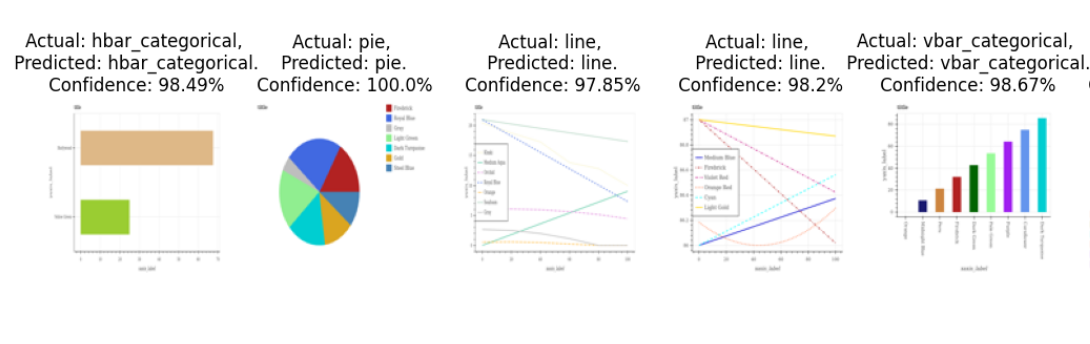
**Accuracy= 0.9814 i.e 98.14%**

* The Validation accuracy of model is :

**Accuracy= 0.9792 i.e 97.12%**

**Model Prediction:**

* The trained model was used to predict the types of charts in the test set, which consisted of 50 images.
* The images were loaded using the PIL library, and the prediction was made using the predict method of the Keras API.
* The predicted class label and the actual label was printed and had good result.



**CONCLUSION:**

* The actual and predicted Values are matching for test images

The model can be further improved by increasing the number of epochs, using a larger dataset, or using a more complex model architecture.

**Task 3: Finetune a pretrained network (e.g., AlexNet) for this task and report the results:**

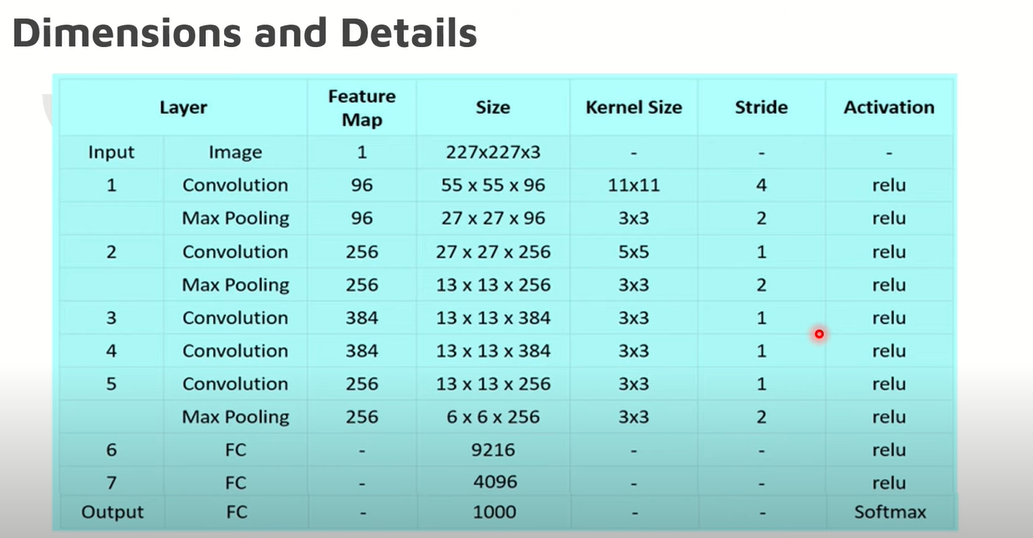
**INTRODUCTION:**

The following problem was expanded and an pre-trained model ‘Alexnet’ is applied for the training and validation set to note the results.

**The git link for the entire code is here =>** https://github.com/tariyalkaran/Machine\_Learning\_Fractal3\_Assignment/blob/main/3\_Task\_3\_Tune\_Pretrained\_Model\_Alexnet.ipynb

**ALEXNET MODEL BUILDING:**

AlexNet is a convolutional neural network architecture designed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton for the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC). It consists of 8 layers: 5 convolutional layers, 2 fully connected layers, and 1 softmax output layer. The architecture was one of the first to use rectified linear units (ReLU) as activation functions and dropout regularization to prevent overfitting. AlexNet achieved state-of-the-art performance on the ILSVRC 2012 competition, and its success popularized the use of deep learning and convolutional neural networks in computer vision.

****

**DATA PREPARATION:**

The training and validation datasets wrere iterated to get batches of preprocessed images and their respective labels.

The labels are then converted into one-hot encoded vectors using the to\_categorical function from Keras.

**IMPLEMENTATION:**

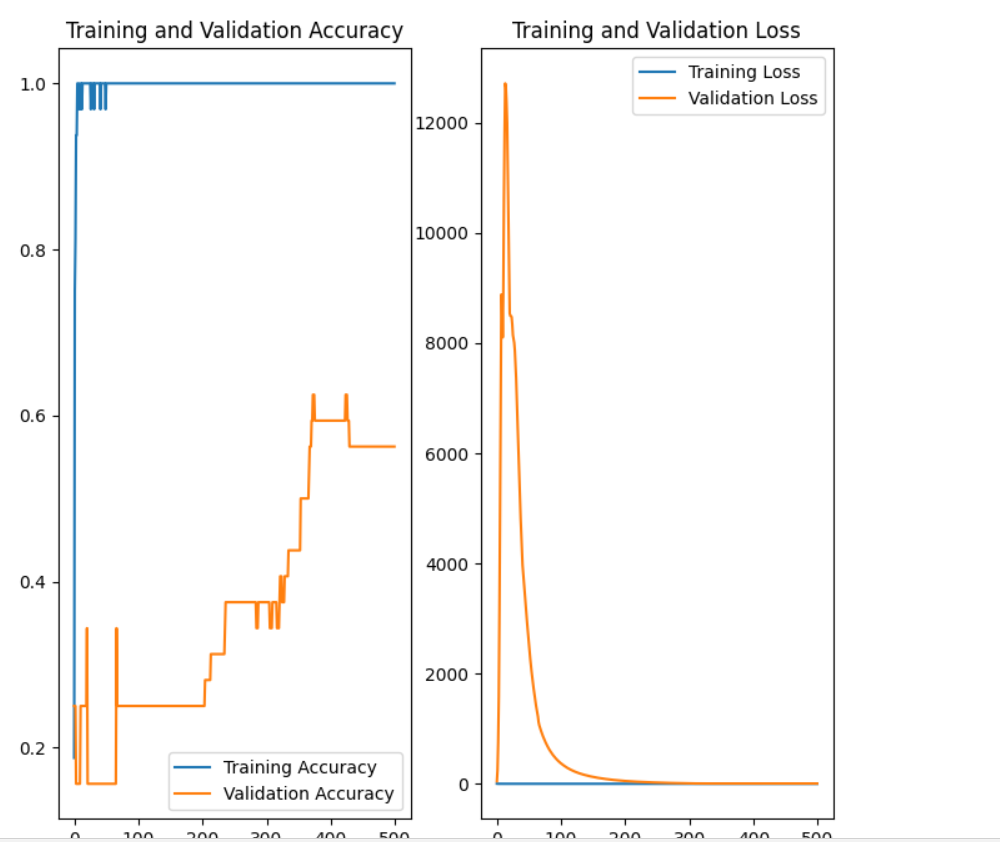
* The following alexnet architecture were applied to the above processed train and valid sets
* Implementation: First, you need to import the necessary libraries and datasets for training and testing the model.
* Compiling: Next, you'll define the model architecture using the Keras API and compile the model with an optimizer and loss function.
* Fitting: Then, you'll fit the model to the training data, specifying the batch size, number of epochs, and validation split.
* Evaluation: After training, you can evaluate the model's performance on the test data and calculate metrics such as accuracy and loss.
* Fine-tuning: Finally, you can fine-tune the model by adjusting the hyperparameters and retraining on the data to improve performance.

**ACCURACY OF THE MODEL:**

The training set accuracy : 100%

Validation set accuracy:71%

**Calculate Accuracy and loss and plot the obtained accuracy and loss.**

****

**CONCLUSION:**

The model can be further improved by increasing the number of epochs, using a larger dataset, or using a more complex model architecture.

The dataset worked pretty well for CNN with greater accuracy rather than Alexnet .