OBJECT COLOR DETECTION USING CNN

The objective of this project is to use Convolutional Neural Networks (CNN) for object color detection. The project uses the CIFAR-10 dataset, which is a collection of 60,000 32x32 color images in 10 different classes, such as airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

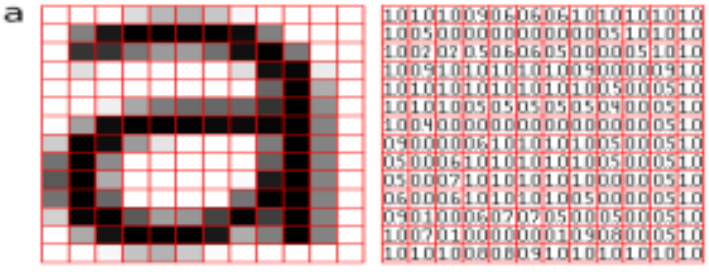
Abstract:

The algorithm utilized in this case is a Convolutional Neural Network, or ConvNet, which features a sophisticated section called Max-pooling that manages the details. It has a self-learning neuron that processes a variety of inputs and sends out appropriate outputs. Multiple layers of perceptron make up CNN. Each layer has several neurons that are intended to prevent data overfitting. It requires less pre-processing than other algorithms because here data is self-learned and taught, as opposed to other algorithms where data must be manually saved and synthesized. We can see that SimpleNet offers the finest outcomes and a platform for this self-learning visual mechanism by using these concepts. Changing the matrix containing the epochs will yield the best results. By utilizing CNN, issues like overfitting are solved and time complexity is decreased. It can accept input in the form of images with the appropriate grayscale size, which is followed by the images' fragmentation into smaller sizes for improved machine comprehension.

There are 10 unique classes in Cifar-10, all of which are mutually exclusive. There are 60,000 total photos in it, 50,000 of which are used for testing and the remaining 20,000 for training. Testing can be done both with and without data augmentation. We can get better results by modifying the inputs, the epochs, and the input matrix.

Introduction:

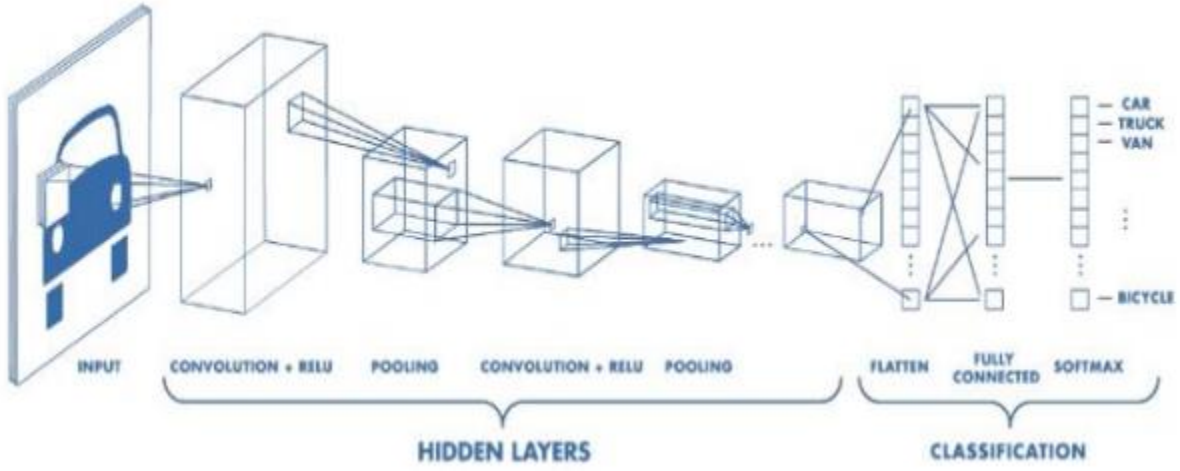
CNN, or Convolutional Neural Network ConvNet is another name for CNN, a unique technique for turning an image into a grid-like structure. As a result, each pixel is distinct and serves to highlight the color scheme and level of brightness in each section of the image.



CNN Architecture: It is a three-layered system namely.

- Convolutional
- Pooling
- Fully Connected

Convolution is the first layer. There are two matrices formed, one with learnable parameters (kernels) and the other in the constrained receptive field. Its depth (which might be up to 3 units) will be considerably smaller than its height and width. Operating a pool. Pooling layer: By removing some visual field data, this layer makes the algorithm smaller and less time-consuming while still producing output and preventing overfitting. Max pooling, in which output is produced using neighbor values, is the method that is most frequently utilized. Utilizing Cifar-10 for object detection with the help of Cifar10, which is a data set used to train our machine using SimpleNet technology, we first understand the data before converting it to a computer programme and feeding it for recognition. Cifar-10The 60,000 photos in the Cifar-10 data set are split into 10,000 test images and 50,000 training images. Out of the 10,000 pictures, 1000 pictures are randomly chosen from each category. Diagram of model 1.4 The classes in the dataset don't cross over with one another. A matrix is created for each entered value and is then searched. Java-based SimpleNet is a framework that can send and receive data requests.



Dataset Preparation:

The CIFAR-10 dataset is preloaded in many machines learning libraries, including Keras. The dataset contains 60,000 32x32 color images, divided into 10 classes, with 6,000 images per class. The dataset is split into training and testing sets, with 50,000 and 10,000 images, respectively.

We first loads the CIFAR-10 dataset using the **keras.datasets.cifar10.load_data()** function and assigns the training and test data to separate variables **train_images, train_labels, test_images**, and test_labels.

Next, it selects only the airplane images from the training and test datasets by creating a Boolean mask that checks if the label equals 0, which is the category for airplanes. The **np.squeeze()** function is used to remove any extra dimensions created by the Boolean mask.

Then, we normalize the pixel values of the images by dividing them by 255.0, which scales the pixel values to a range between 0 and 1.

Building the CNN Model:

The CNN model is built using the Keras library in Python. The model comprises of multiple convolutional and pooling layers, followed by dense layers. The input to the model is a 32x32x3 image, where 3 represents the color channels (RGB). The output of the model is a vector of size 10, representing the probabilities of the image belonging to each of the 10 classes.

We create a convolutional neural network (CNN) model using the Keras API with the following architecture:

The first layer is a Conv2D layer with 32 filters, each of size 3x3, with a ReLU activation function, and the input shape of (32, 32, 3), which represents the shape of the input image. The second layer is a MaxPooling2D layer with a pool size of 2x2. The third layer is another Conv2D layer with 64 filters, each of size 3x3, with a ReLU activation function. The fourth layer is another MaxPooling2D layer with a pool size of 2x2. The fifth layer is a Flatten layer that flattens the output of the previous layer into a 1D vector. The sixth layer is a fully connected Dense layer with 64 units and a ReLU activation function. The seventh and final layer is another Dense layer with 3 units and a SoftMax activation function, which produces the probabilities of each of the 3 classes.

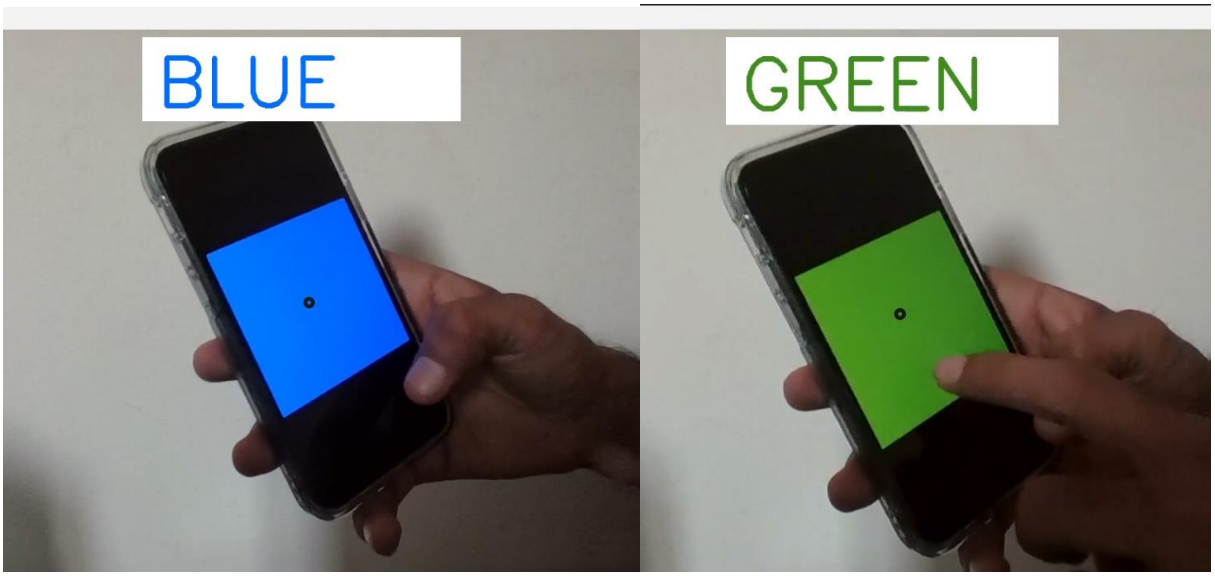
Training the Model:

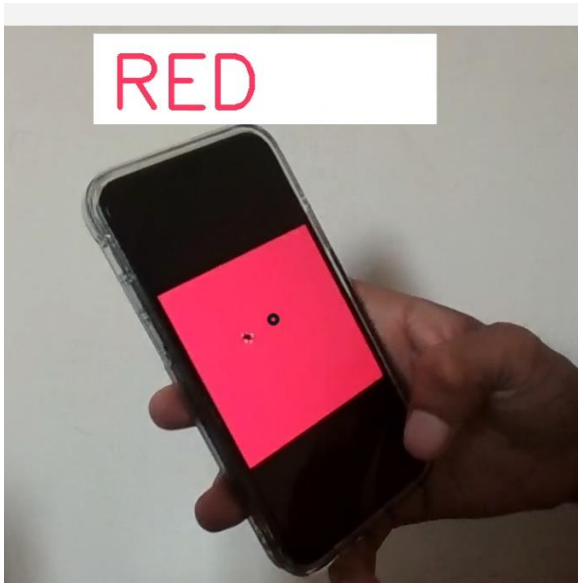
The model is trained using the training set of the CIFAR-10 dataset. The training is done using stochastic gradient descent with a learning rate of 0.001 and a batch size of 32. The model is trained for 10 epochs, with early stopping to prevent overfitting.

```
Epoch 1/10
157/157 [==============================] - 9s 47ms/step - loss: 0.0125 - accuracy: 0.9936 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 2/10
157/157 [==============================] - 8s 53ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 3/10
157/157 [==============================] - 7s 45ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 4/10
157/157 [==============================] - 8s 53ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 5/10
157/157 [==============================] - 8s 49ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 6/10
157/157 [==============================] - 7s 47ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 7/10
157/157 [==============================] - 8s 52ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 8/10
157/157 [==============================] - 7s 43ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 9/10
157/157 [==============================] - 9s 55ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
Epoch 10/10
157/157 [==============================] - 7s 43ms/step - loss: 3.0994e-10 - accuracy: 1.0000 - val_loss: 0.0000e+00 - val_accuracy: 1.0000
```

Real-time Object Color Detection:

Once the model is trained, it is used for real-time object color detection. The camera feed is captured using OpenCV, and the images are passed through the CNN model to detect the color of the object in the frame. The output of the model is the class with the highest probability, which represents the color of the object. The results are displayed in real-time on the screen.





Conclusion:

This project demonstrates the use of CNNs for object color detection using the CIFAR-10 dataset. The model developed in this project can be further improved and used in various computer vision applications.

References:

Some references that may be helpful for learning more about object color detection and the CIFAR-10 dataset:

1. Bilen, H., & Vedaldi, A. (2017). Universal representations: The missing link between faces, text, planktons, and cat breeds. arXiv preprint arXiv:1701.07275.
2. Krizhevsky, A., & Hinton, G. E. (2010). Convolutional deep belief networks on CIFAR-10. Unpublished manuscript, 1-18.
3. CIFAR-10 dataset: https://www.cs.toronto.edu/~kriz/cifar.html
4. Liao, S., & Jain, A. K. (2013). Color object recognition using histograms of color gradient orientations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(8), 1914-1924.
5. Shan, C., Gong, S., & McOwan, P. W. (2005). Facial expression recognition based on local binary patterns: A comprehensive study. Image and Vision Computing, 27(6), 803-816.