OBJECT RECOGNITION USING COIL-100 DATASET

LAKSHANA TEKULA

ABSTRACT:

In this work, we used a dataset of 100 distinct object classes to construct and assess a Deep Convolutional Neural Network (DCNN) model for object recognition. The model design included batch normalization convolutional layers, max-pooling layers, and a fully connected layer, followed by a Softmax activation function for classification. ReLU activation functions, the ADAM optimization approach, and the Categorical Cross-Entropy loss function were all used. The model was trained for 50 epochs with batch sizes of 16, 32, and 64, and its performance was evaluated on an independent test dataset. The accuracy, F1-score, ROC, and precision-recall curve were used to evaluate the findings, and the computing times for both the training and testing phases were recorded. This report provides in-depth information about the model's training process and performance in object recognition tasks. We diligently created and analyzed a dataset containing 100 distinct item classes in this extensive analysis, shrinking the photos to 64x64 pixels. The dataset was deliberately partitioned into training and testing sets, with each class receiving an 80:20 split. A meticulously designed Deep Convolutional Neural Network (DCNN) was used for classification, which included a series of convolutional layers with batch normalization and max pooling, ending in a fully connected layer with a Softmax activation. The training regimen covered 50 epochs, with batch sizes ranging from 16 to 64 to investigate their impact. An independent test dataset was used for evaluation, with thorough analysis of performance indicators like accuracy, F1-score, ROC analysis, and precision-recall curves. The training and testing phases' computing times were meticulously documented to provide a comprehensive knowledge of the model's efficiency and performance in real-world object identification settings.

KEYWORDS: Layers, ADAM, Categorical Cross-Entropy loss, Batch size, ROC, F1-score.

INTRODUCTION:

A) PROBLEM DEFINITION:

The problem to be addressed is the creation of an effective Deep Convolutional Neural Network (DCNN) model for object recognition in a dataset with 100 discrete object classes. The determination of an optimal DCNN architecture capable of effectively capturing discriminative features from 64x64 pixel images, the investigation of the impact of key hyperparameters such as batch size and training epochs on the model's convergence and generalization abilities, and the selection of appropriate performance metrics such as accuracy, F1-score, ROC analysis, and precision-recall curves to comprehensively evaluate the model's recognition abilities are all fundamental questions. These theoretical considerations are critical for developing a robust DCNN model and determining its applicability for real-world object identification issues.

Aim of this project:

To implement a classification model for object recognition

- Deep Convolution neural network (DCNN) model using ReLU activation function.

  Model Configuration:

  64x64(inputs) →2xConv(3x3x16) +BN→MXP(2x2) → 2xConv(3x3x32) +BN →
  MXP(2x2) → 2xConv(3x3x64))+BN → MXP(2x2) → 2xConv(3x3x128))+BN → →
  MXP(2x2) → 2xConv(3x3x256))+BN → FCL (Dense layer_512) →SoftMax(100-class)

  (a) Activation functions: ReLU
  (b) Use a SoftMax activation function for classification.
  (c) Optimization method: ADAM
  (d) Loss: Categorical Cross-Entropy loss
  (e) Number of epochs: 50 and
  (f) Batch-size is 16 or 32 or 64.

B) BACKGROUND STUDY:

Object identification has significance in a variety of real-world applications, including driverless vehicles, industrial automation, and augmented reality. Machines' ability to effectively detect and classify items in their visual field has far-reaching ramifications for industries and technologies. With their hierarchical feature extraction and pattern recognition capabilities, Deep Convolutional Neural Networks (DCNNs) have emerged as the main approach in this attempt. Real-world scenarios frequently involve complicated and diverse object classes, necessitating models that can effectively generalize across a large range of objects. Furthermore, computing efficiency is essential for real-time deployment in applications such as self-driving automobiles. Because of the versatility of DCNNs, as well as the problems given by real-world diversity and computational constraints, the development and evaluation of these models is an important area of research. The selection of datasets, model architectures, hyperparameters, and performance measures is critical to the practical success of object recognition systems, emphasizing the need of conducting a thorough examination into these elements in order to satisfy the demands of real-world scenarios.

RELATED WORKS:

1. **COMPARISON OF DATASET BIAS IN OBJECT RECOGNITION BENCHMARKS by Ian Model, Lior Shamir.**

   This research in the field of autonomous visual object identification relies on benchmark datasets to evaluate the performance of new algorithms. Such datasets might be based on standardized datasets acquired deliberately in a controlled environment (for example, COIL-20), as well as benchmarks produced by gathering photos from diverse sources, typically via the World Wide Web (for example, Caltech 101). We evaluate bias in benchmark datasets by removing a small area from each image that seems blank but is too small to allow manual recognition of the object.

2. **OBJECT RECOGNITION USING DISCRIMINATIVE PARTS by Ying Ho Liu, Anthony J.T. Lee, Fu Chang.**

   The authors proposed a unique object identification method based on object discriminative parts. The proposed approach is divided into three stages. First, they obtain local portions from each model object using sliding windows and extract a feature vector for each local part. Using these feature vectors, they then create prototypes for the model items. Finally, they compute the similarity between a test item and each model object using the correspondence between the local components of the test object and the model object.

3. **DEEP VERSUS WIDE CONVOLUTIONAL NEURAL NETWORKS FOR OBJECT RECOGNITION ON NEUROMORPHIC SYSTEM by Md Zahangir Alom, Theodore Josue, Md Nayim Rahman, Will Mitchell, Chris Yakopcic.**

   We empirically examined the performance of various DCNN architectures implemented within the Eedn framework in this research. The purpose of this effort was to find the most efficient approach to use the TrueNorth system to create DCNN models for object categorization tasks. We ran our trials with benchmark data sets like MNIST, COIL-20, and COIL-100. On IBM's NSle Neurosynaptic system, the experimental findings demonstrate highly promising classification accuracies with exceptionally low power consumption. The results reveal that for datasets with a high number of classes, wider networks outperform deep networks with almost the same core complexity as IBM's TrueNorth system.

C) OBJECTIVES AND CONTRIBUTION:

- **Data preprocessing:**

  Data preprocessing is used to convert data into clean data sets which are used for analysis. It consists of several steps like data cleansing, data reduction, data enrichment organization and transformation. So, it will be easy for machine learning models to read data and learn from data.

- **Descriptive statistics:**

  Input feature properties, pixel value statistics (minimum, maximum, mean, median, standard deviation), sample resolution, and batch sizes, as well as details of any data augmentation techniques utilized, are all included in descriptive statistics. This data serves as the foundation for understanding the dataset's structure, distribution, and diversity, which are all important for effective model training and performance evaluation in object recognition.

- **Model Training:**

  We create a DCNN model based on the configurations provided, utilizing ReLU activation functions. SoftMax activation for classification, Categorical Cross-Entropy loss is used to build these models. Training takes place over a predetermined number of epochs with a fixed batch size, and model performance is evaluated on the validation set at the conclusion of each epoch.

- **Model Evaluation:**

  Model Evaluation is done by calculating different performance evaluation metrics like Accuracy, Precision, Recall, F1-score, ROC Curve, Precision-Recall curve, Computational times. This can be done using test dataset.

METHODOLOGY:

- Loading of dataset: In this step, we will load dataset from Kaggle using API.

- Converting images zip file into a dataset.

- EDA: In this step we will visualize the images and labels to describe features of the image. We will understand the shape of the train and test dataset. Whether the dataset is balanced or not can be known by this analysis.

- Splitting of data: In this step we will divide the entire dataset into two groups, training set and testing set. Again, we will divide the training set into train set and validation set. Training set is used to train the model and learn from it. Validation set is used to validate data. Testing data is used to evaluate model performance.

- Data preprocessing: In data preprocessing we will do pixel normalization, one hot encoding of both test and train images.

- Model creation: In this project we will create different models like Deep convolutional neural network using ReLU activation function. We used categorical cross-entropy loss function. These are used for object recognition.

- Model training: Model training is done by using training images. The model will get the relationship between the images and labels, and it will give us the predicted labels.

- Model Evaluation: Model evaluation is done using testing images. Using these images, we will calculate different performance evaluating features like Accuracy, Precision, Recall, F1-Score, ROC-Curve, Precision-Recall Curve, computational times.

- Analysis of results: The performance evaluating features give us whether the model we created is effective or not.

- Conclusion from results: We will conclude the performance metrics of model

MODEL DESCRIPTION:

- **Deep Convolutional Neural Network (DCNN) model using ReLU activation function:**

  The DCNN-ReLU model, short for a Convolutional Neural Network with Rectified Linear Unit activation functions, is a deep learning architecture designed for computer vision tasks. This model consists of layers that apply learnable filters to extract features from input data, such as images, with the activation functions utilizing Rectified Linear Units (ReLU) to introduce non-linearity. The model typically includes convolutional layers, pooling layers for dimensionality reduction, fully connected layers for final classification or regression, and an output layer for predictions. Through training with backpropagation and gradient descent, it adjusts its parameters to minimize a specified loss function, making it a powerful tool for tasks like image classification, object detection, and image segmentation, known for its ability to capture complex visual patterns in data.

  The proposed Deep Convolutional Neural Network (DCNN) model for object recognition in real-world scenarios is tailored to manage the inherent complexities of diverse object classes and computational constraints. This model comprises a series of meticulously designed layers, commencing with two sets of convolutional layers, each incorporating 3x3 filters with 16, 32, 64, and 128 filters, accompanied by batch normalization to enhance training stability. Max-pooling layers (2x2) are strategically inserted after each pair of convolutional layers, facilitating spatial dimension reduction.

The final convolutional layer features 256 filters to capture intricate details. The model culminates with a fully connected layer housing 512 units, where a Softmax activation function is applied for classification. The use of Rectified Linear Units (ReLU) throughout the model ensures efficient feature extraction, mitigating the gradient vanishing problem. Training spans 50 epochs, with batch sizes varying from 16 to 64, allowing for a comprehensive assessment of performance under different conditions. This model's architecture and configuration have been meticulously designed to address the intricate demands of real-world object recognition, providing adaptability to diverse object classes and computational efficiency to meet the needs of practical applications.

EXPERIMENT AND RESULTS:

A) DATABASE:

COIL-100 Database:

- Input image: 128x128 pixels
- No. of samples: 7200
- No. of classes: 100 (Objects)
- Batch size selected: 64

COIL-100 Kaggle dataset link: https://www.kaggle.com/datasets/jessicali9530/coil100/

We imported COIL-100 Database from Kaggle

```
!kaggle datasets download -d jessicali9530/coil100
```

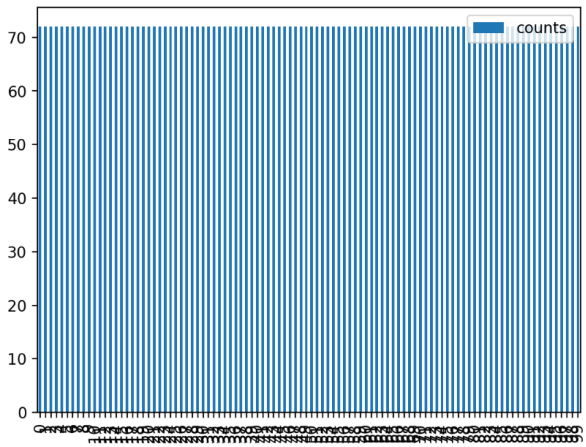
```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /content/kaggle.json'
Downloading coil100.zip to /content
100% 127M/127M [00:03<00:00, 41.7MB/s]
100% 127M/127M [00:03<00:00, 36.0MB/s]
```

Converting images zip file into a dataset

```
df.head(10)
```

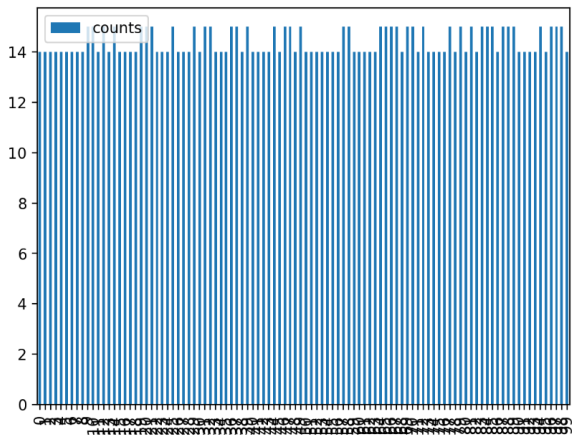
	path	label
0	coil-100/coil-100/obj21__0.png	obj21
1	coil-100/coil-100/obj73__110.png	obj73
2	coil-100/coil-100/obj92__145.png	obj92
3	coil-100/coil-100/obj58__5.png	obj58
4	coil-100/coil-100/obj39__145.png	obj39
5	coil-100/coil-100/obj93__235.png	obj93
6	coil-100/coil-100/obj5__205.png	obj5
7	coil-100/coil-100/obj43__60.png	obj43
8	coil-100/coil-100/obj51__20.png	obj51
9	coil-100/coil-100/obj64__45.png	obj64

Exploratory data analysis (EDA):

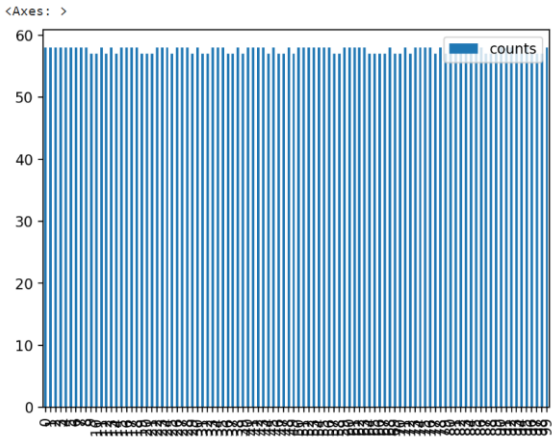


We can see that dataset is very well balanced with 72 images per each object

```
training images =  (5760,) training labels =  (5760,)
testing images =  (1440,) testing labels =  (1440,)
```

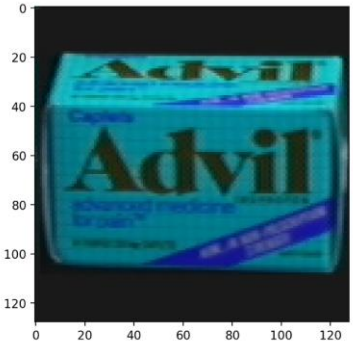


Train dataset



Test dataset

Visualization of images:



B) TRAINING AND TESTING LOGS:

- **Model: Deep Convolutional Neural Network (DCNN) using ReLU activation function.**

Model Summary:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 16)      448

 batch_normalization (Batch  (None, 126, 126, 16)      64
 Normalization)

 max_pooling2d (MaxPooling2  (None, 63, 63, 16)        0
 D)

 conv2d_1 (Conv2D)           (None, 61, 61, 32)        4640

 batch_normalization_1 (Bat  (None, 61, 61, 32)        128
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 30, 30, 32)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 28, 28, 64)        18496

 batch_normalization_2 (Bat  (None, 28, 28, 64)        256
 chNormalization)

 max_pooling2d_2 (MaxPoolin  (None, 14, 14, 64)        0
 g2D)

 conv2d_3 (Conv2D)           (None, 12, 12, 128)       73856

 batch_normalization_3 (Bat  (None, 12, 12, 128)       512
 chNormalization)

 max_pooling2d_3 (MaxPoolin  (None, 6, 6, 128)         0
 g2D)

 conv2d_4 (Conv2D)           (None, 4, 4, 256)         295168

 batch_normalization_4 (Bat  (None, 4, 4, 256)         1024
 chNormalization)

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 512)               2097664

 dense_1 (Dense)             (None, 100)               51300

=================================================================
Total params: 2543556 (9.70 MB)
Trainable params: 2542564 (9.70 MB)
Non-trainable params: 992 (3.88 KB)
_________________________________________________________________
```

Batch size :64, No of epochs: 50

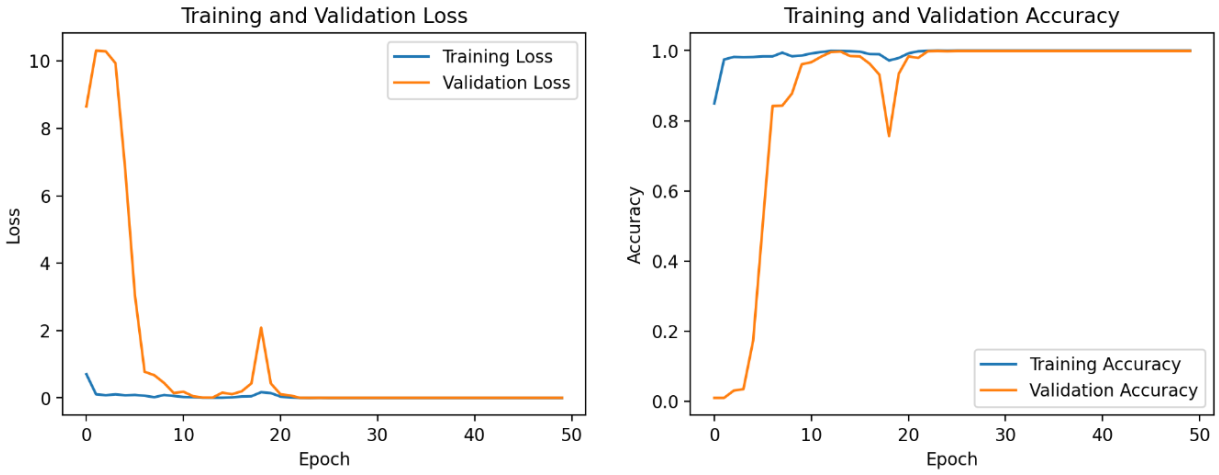
```
training images =  (4608, 128, 128, 3) training labels categorical =  (4608, 100)
validation images=  (1152, 128, 128, 3) validation labels categorical =  (1152, 100)
```

Computational Time:

```
Epoch 1/50
72/72 [==============================] - 24s 42ms/step - loss: 0.7040 - accuracy: 0.8490 - val_loss: 8.6555 - val_accuracy: 0.0095
Epoch 2/50
72/72 [==============================] - 2s 32ms/step - loss: 0.1074 - accuracy: 0.9746 - val_loss: 10.3087 - val_accuracy: 0.0095
Epoch 3/50
72/72 [==============================] - 3s 37ms/step - loss: 0.0806 - accuracy: 0.9820 - val_loss: 10.2873 - val_accuracy: 0.0304
Epoch 4/50
72/72 [==============================] - 2s 32ms/step - loss: 0.1053 - accuracy: 0.9813 - val_loss: 9.9332 - val_accuracy: 0.0347
Epoch 5/50
72/72 [==============================] - 2s 31ms/step - loss: 0.0787 - accuracy: 0.9818 - val_loss: 6.7799 - val_accuracy: 0.1727
Epoch 6/50
72/72 [==============================] - 2s 30ms/step - loss: 0.0890 - accuracy: 0.9835 - val_loss: 3.0520 - val_accuracy: 0.5087
Epoch 7/50
72/72 [==============================] - 2s 31ms/step - loss: 0.0648 - accuracy: 0.9835 - val_loss: 0.7763 - val_accuracy: 0.8420
Epoch 8/50
72/72 [==============================] - 2s 34ms/step - loss: 0.0219 - accuracy: 0.9941 - val_loss: 0.6732 - val_accuracy: 0.8429
Epoch 9/50
72/72 [==============================] - 2s 32ms/step - loss: 0.0874 - accuracy: 0.9837 - val_loss: 0.4455 - val_accuracy: 0.8776
Epoch 10/50
72/72 [==============================] - 2s 30ms/step - loss: 0.0579 - accuracy: 0.9859 - val_loss: 0.1453 - val_accuracy: 0.9609
Epoch 11/50
72/72 [==============================] - 2s 31ms/step - loss: 0.0287 - accuracy: 0.9920 - val_loss: 0.1837 - val_accuracy: 0.9670
Epoch 12/50
72/72 [==============================] - 2s 31ms/step - loss: 0.0201 - accuracy: 0.9959 - val_loss: 0.0496 - val_accuracy: 0.9826
Epoch 13/50
72/72 [==============================] - 2s 33ms/step - loss: 0.0050 - accuracy: 0.9991 - val_loss: 0.0106 - val_accuracy: 0.9957
Epoch 14/50
72/72 [==============================] - 2s 34ms/step - loss: 0.0041 - accuracy: 0.9989 - val_loss: 0.0098 - val_accuracy: 0.9974
Epoch 15/50
72/72 [==============================] - 2s 31ms/step - loss: 0.0052 - accuracy: 0.9983 - val_loss: 0.1567 - val_accuracy: 0.9844
Epoch 16/50
72/72 [==============================] - 2s 31ms/step - loss: 0.0158 - accuracy: 0.9970 - val_loss: 0.1109 - val_accuracy: 0.9835
Epoch 17/50
72/72 [==============================] - 2s 33ms/step - loss: 0.0432 - accuracy: 0.9900 - val_loss: 0.1965 - val_accuracy: 0.9627
Epoch 18/50
72/72 [==============================] - 2s 31ms/step - loss: 0.0497 - accuracy: 0.9896 - val_loss: 0.4318 - val_accuracy: 0.9314
Epoch 19/50
72/72 [==============================] - 2s 34ms/step - loss: 0.1703 - accuracy: 0.9718 - val_loss: 2.0853 - val_accuracy: 0.7561
Epoch 20/50
72/72 [==============================] - 3s 37ms/step - loss: 0.1435 - accuracy: 0.9789 - val_loss: 0.4296 - val_accuracy: 0.9340
Epoch 21/50
72/72 [==============================] - 2s 33ms/step - loss: 0.0392 - accuracy: 0.9922 - val_loss: 0.1077 - val_accuracy: 0.9835
Epoch 22/50
72/72 [==============================] - 2s 32ms/step - loss: 0.0100 - accuracy: 0.9980 - val_loss: 0.0724 - val_accuracy: 0.9792
Epoch 23/50
72/72 [==============================] - 2s 32ms/step - loss: 0.0017 - accuracy: 0.9993 - val_loss: 0.0035 - val_accuracy: 0.9983
Epoch 24/50
72/72 [==============================] - 2s 34ms/step - loss: 1.8581e-04 - accuracy: 1.0000 - val_loss: 0.0015 - val_accuracy: 0.9991
Epoch 25/50
72/72 [==============================] - 3s 37ms/step - loss: 0.0031 - accuracy: 0.9996 - val_loss: 0.0025 - val_accuracy: 0.9983
Epoch 26/50
72/72 [==============================] - 2s 32ms/step - loss: 2.5852e-05 - accuracy: 1.0000 - val_loss: 0.0015 - val_accuracy: 0.9991
Epoch 27/50
72/72 [==============================] - 2s 32ms/step - loss: 2.8710e-05 - accuracy: 1.0000 - val_loss: 0.0017 - val_accuracy: 0.9991
Epoch 28/50
72/72 [==============================] - 2s 33ms/step - loss: 1.0017e-05 - accuracy: 1.0000 - val_loss: 0.0018 - val_accuracy: 0.9991
Epoch 29/50
72/72 [==============================] - 2s 33ms/step - loss: 7.6209e-06 - accuracy: 1.0000 - val_loss: 0.0019 - val_accuracy: 0.9991
Epoch 30/50
72/72 [==============================] - 2s 32ms/step - loss: 4.1610e-05 - accuracy: 1.0000 - val_loss: 0.0021 - val_accuracy: 0.9991
Epoch 31/50
72/72 [==============================] - 2s 32ms/step - loss: 4.8269e-06 - accuracy: 1.0000 - val_loss: 0.0021 - val_accuracy: 0.9991
Epoch 32/50
72/72 [==============================] - 2s 32ms/step - loss: 6.1806e-06 - accuracy: 1.0000 - val_loss: 0.0021 - val_accuracy: 0.9991
```

```
Epoch 33/50
72/72 [==============================] - 2s 33ms/step - loss: 4.1729e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 34/50
72/72 [==============================] - 2s 31ms/step - loss: 7.7881e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 35/50
72/72 [==============================] - 3s 35ms/step - loss: 4.2937e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 36/50
72/72 [==============================] - 2s 32ms/step - loss: 6.0884e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 37/50
72/72 [==============================] - 2s 31ms/step - loss: 4.4603e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 38/50
72/72 [==============================] - 2s 32ms/step - loss: 4.2305e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 39/50
72/72 [==============================] - 2s 30ms/step - loss: 4.0270e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 40/50
72/72 [==============================] - 2s 34ms/step - loss: 2.8356e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 41/50
72/72 [==============================] - 2s 33ms/step - loss: 5.1047e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 42/50
72/72 [==============================] - 2s 32ms/step - loss: 3.9349e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 43/50
72/72 [==============================] - 2s 31ms/step - loss: 2.9062e-06 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 0.9991
Epoch 44/50
72/72 [==============================] - 2s 34ms/step - loss: 1.7143e-05 - accuracy: 1.0000 - val_loss: 0.0018 - val_accuracy: 0.9991
Epoch 45/50
72/72 [==============================] - 2s 32ms/step - loss: 2.1629e-05 - accuracy: 1.0000 - val_loss: 0.0021 - val_accuracy: 0.9991
Epoch 46/50
72/72 [==============================] - 2s 33ms/step - loss: 4.8806e-06 - accuracy: 1.0000 - val_loss: 0.0019 - val_accuracy: 0.9991
Epoch 47/50
72/72 [==============================] - 2s 31ms/step - loss: 3.5729e-06 - accuracy: 1.0000 - val_loss: 0.0018 - val_accuracy: 0.9991
Epoch 48/50
72/72 [==============================] - 2s 32ms/step - loss: 2.0283e-06 - accuracy: 1.0000 - val_loss: 0.0018 - val_accuracy: 0.9991
Epoch 49/50
72/72 [==============================] - 2s 32ms/step - loss: 1.9251e-06 - accuracy: 1.0000 - val_loss: 0.0018 - val_accuracy: 0.9991
Epoch 50/50
72/72 [==============================] - 2s 32ms/step - loss: 2.9445e-06 - accuracy: 1.0000 - val_loss: 0.0017 - val_accuracy: 0.9991
```

Training and Validation errors:



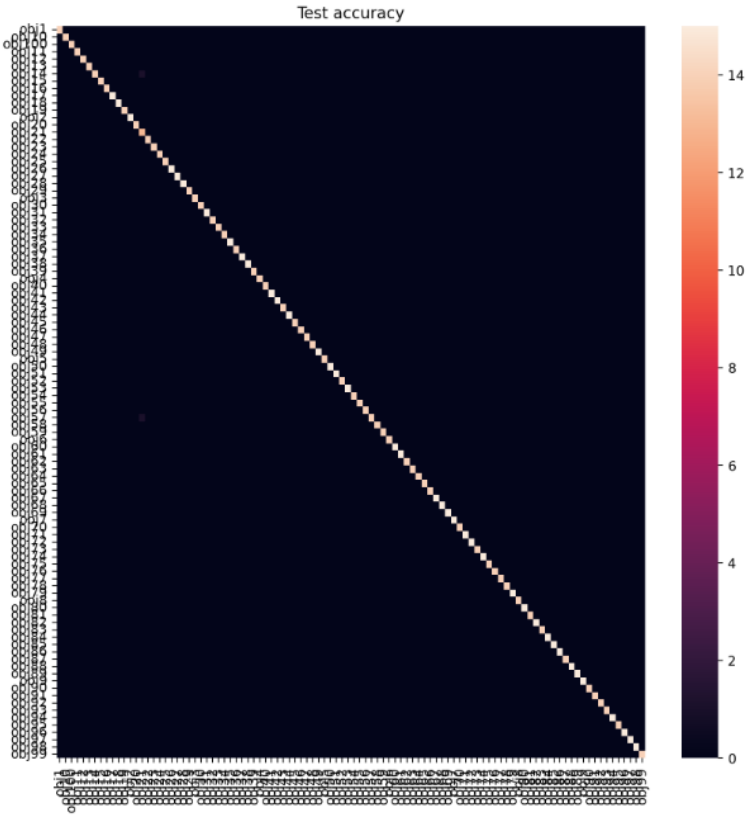
Performance metrics:

Accuracy: 99.861109 loss: 0.408548

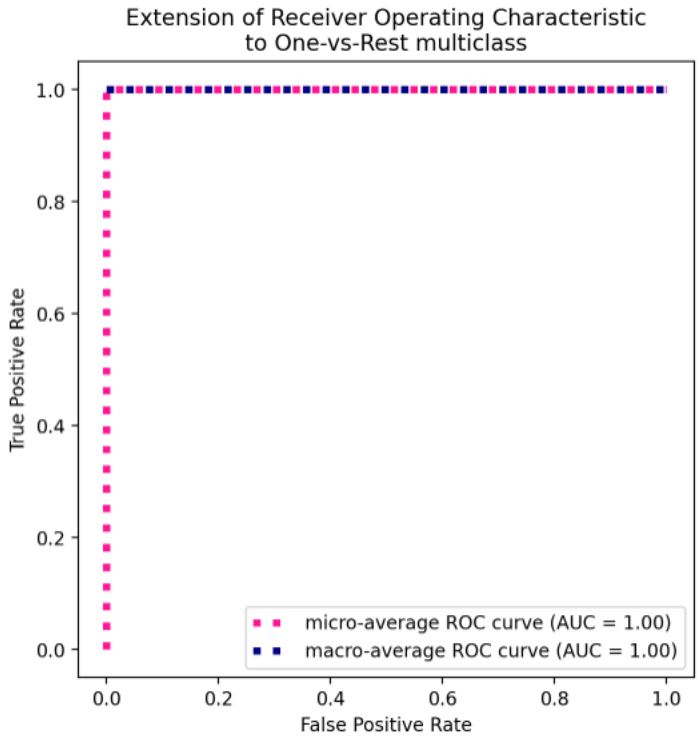
Classification Report for Test Set:

	precision	recall	f1-score	support
obj1	1.00	1.00	1.00	14
obj10	1.00	1.00	1.00	14
obj100	1.00	1.00	1.00	14
obj11	1.00	1.00	1.00	14
obj12	1.00	1.00	1.00	14
obj13	1.00	1.00	1.00	14
obj14	0.93	1.00	0.97	14
obj15	1.00	1.00	1.00	14
obj16	1.00	1.00	1.00	14
obj17	1.00	1.00	1.00	15
obj18	1.00	1.00	1.00	15
obj19	1.00	1.00	1.00	14
obj2	1.00	1.00	1.00	15
obj20	1.00	1.00	1.00	14
obj21	1.00	0.87	0.93	15
obj22	1.00	1.00	1.00	14
obj23	1.00	1.00	1.00	14
obj24	1.00	1.00	1.00	14
obj25	1.00	1.00	1.00	14
obj26	1.00	1.00	1.00	15
obj27	1.00	1.00	1.00	15
obj28	1.00	1.00	1.00	15
obj29	1.00	1.00	1.00	14
obj3	1.00	1.00	1.00	14
obj30	1.00	1.00	1.00	14
obj31	1.00	1.00	1.00	15
obj32	1.00	1.00	1.00	14
obj33	1.00	1.00	1.00	14
obj34	1.00	1.00	1.00	14
obj35	1.00	1.00	1.00	15
obj36	1.00	1.00	1.00	14
obj37	1.00	1.00	1.00	15
obj38	1.00	1.00	1.00	15
obj39	1.00	1.00	1.00	14
obj4	1.00	1.00	1.00	14
obj40	1.00	1.00	1.00	14
obj41	1.00	1.00	1.00	15
obj42	1.00	1.00	1.00	15
obj43	1.00	1.00	1.00	14
obj44	1.00	1.00	1.00	15
obj45	1.00	1.00	1.00	14
obj46	1.00	1.00	1.00	14
obj47	1.00	1.00	1.00	14
obj48	1.00	1.00	1.00	14
obj49	1.00	1.00	1.00	15
obj5	1.00	1.00	1.00	14
obj50	1.00	1.00	1.00	15
obj51	1.00	1.00	1.00	15
obj52	1.00	1.00	1.00	14

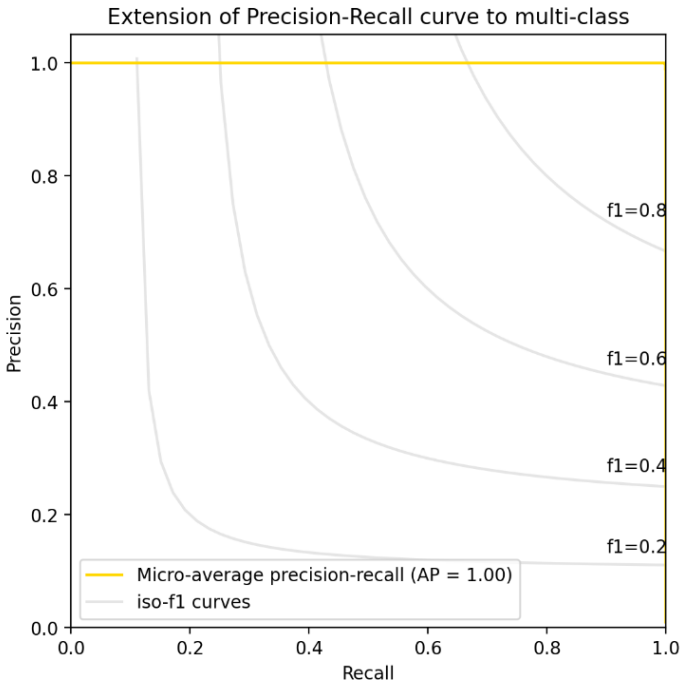
obj53	1.00	1.00	1.00	15
obj54	1.00	1.00	1.00	14
obj55	1.00	1.00	1.00	14
obj56	1.00	1.00	1.00	14
obj57	0.93	1.00	0.97	14
obj58	1.00	1.00	1.00	14
obj59	1.00	1.00	1.00	14
obj6	1.00	1.00	1.00	14
obj60	1.00	1.00	1.00	15
obj61	1.00	1.00	1.00	15
obj62	1.00	1.00	1.00	14
obj63	1.00	1.00	1.00	14
obj64	1.00	1.00	1.00	14
obj65	1.00	1.00	1.00	14
obj66	1.00	1.00	1.00	14
obj67	1.00	1.00	1.00	15
obj68	1.00	1.00	1.00	15
obj69	1.00	1.00	1.00	15
obj7	1.00	1.00	1.00	15
obj70	1.00	1.00	1.00	14
obj71	1.00	1.00	1.00	15
obj72	1.00	1.00	1.00	15
obj73	1.00	1.00	1.00	14
obj74	1.00	1.00	1.00	15
obj75	1.00	1.00	1.00	14
obj76	1.00	1.00	1.00	14
obj77	1.00	1.00	1.00	14
obj78	1.00	1.00	1.00	14
obj79	1.00	1.00	1.00	15
obj8	1.00	1.00	1.00	14
obj80	1.00	1.00	1.00	15
obj81	1.00	1.00	1.00	14
obj82	1.00	1.00	1.00	15
obj83	1.00	1.00	1.00	14
obj84	1.00	1.00	1.00	15
obj85	1.00	1.00	1.00	15
obj86	1.00	1.00	1.00	15
obj87	1.00	1.00	1.00	14
obj88	1.00	1.00	1.00	15
obj89	1.00	1.00	1.00	15
obj9	1.00	1.00	1.00	15
obj90	1.00	1.00	1.00	14
obj91	1.00	1.00	1.00	14
obj92	1.00	1.00	1.00	14
obj93	1.00	1.00	1.00	14
obj94	1.00	1.00	1.00	15
obj95	1.00	1.00	1.00	14
obj96	1.00	1.00	1.00	15
obj97	1.00	1.00	1.00	15
obj98	1.00	1.00	1.00	15
obj99	1.00	1.00	1.00	14
accuracy			1.00	1440
macro avg	1.00	1.00	1.00	1440
weighted avg	1.00	1.00	1.00	1440


Test accuracy
14
12
10
8
6
4
2
0

ROC-Curve:


Extension of Receiver Operating Characteristic
to One-vs-Rest multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (AUC = 1.00)
macro-average ROC curve (AUC = 1.00)

Precision-Recall Curve:



C) DISCUSSION AND COMPARISON:

To recognize objects using COIL-100 dataset we implemented a model on the dataset. The model worked well on the dataset. Accuracy, Precision, Recall, F1-Score, ROC-Curve, Precision-Recall curves are the performance metrics of a model.

Batch size: 64, No of epochs: 50

Model	**Overall Accuracy**	**Loss**
DCNN-ReLU	99.86%	0.4085

Table: Accuracy value of different model

From the results we can see that Deep Convolutional Neural Network (DNN) using ReLU activation function implemented well on the dataset with accuracy of 0.9986.

CONCLUSION:

In this study, we developed and evaluated a Deep Convolutional Neural Network (DCNN) model designed for object detection in real-world scenarios using a dataset of 100 different item classes. Our model, distinguished by its meticulously designed architecture, which includes convolutional layers with batch normalization, max-pooling layers strategically placed, and a fully connected layer with Softmax activation, represents a comprehensive approach to addressing the challenges of diverse object recognition. We were able to investigate the model's adaptability under various settings after training it for 50 epochs with changing batch sizes. Our findings show that, while the DCNN model is capable of recognizing a wide range of objects, it is also capable of efficient generalization and computational performance. The use of various performance indicators, such as accuracy, F1-score, ROC analysis, and precision-recall curves, allowed a comprehensive evaluation of the model's capabilities. The work emphasizes the importance of DCNN models in real-world object recognition, as well as the need for comprehensive examination of architecture, hyperparameters, and performance measures in order to satisfy the demands of realistic applications. Looking ahead, our study provides a solid platform for enhancing object recognition in complicated and diverse real-world environments.

LIMITATIONS:

This work has some limitations even if it offers a strong basis for object detection in the actual world. First, while the dataset is large and diverse, it may not fully depict the intricacies of real-world settings. The hyperparameter sensitivity and dataset-specific nature, the challenge of achieving computational efficiency in resource-constrained situations and continued worries regarding overfitting mitigation in scenarios with minimal labeled data are all possible areas for development. Variability in preprocessing methodologies, hardware dependence, and the absence of real-world environmental elements such as lighting conditions and occlusions necessitate additional investigation. Furthermore, future study will examine tackling interclass confusion, particularly when items exhibit visual similarities across classes. These constraints highlight the importance of ongoing research and development to improve Deep Convolutional Neural Network models' adaptability and robustness for real-world object detection applications.

REFERENCES:

- J. Ponce, T. L. Berg, M. Everingham, D. A. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. C. Russell, A. Torralba et al., *"Dataset issues in object recognition,"* in Toward category-level object recognition. Springer, 2006, pp. 29–48.

- L. Fei-Fei, R. Fergus, and P. Perona, *"One-shot learning of object categories,"* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 4, pp. 594–611, 2006.

- G. Griffin, A. Holub, and P. Perona, *"Caltech-256 object category dataset,"* Technical Report 7694, California Institute of Technology, Tech. Rep., 2007.

- S. A. Nene, S. K. Nayar, H. Murase et al., *"Columbia object image library (coil-20),"* Technical Report CUCS-005-96, Tech. Rep., 1996.

- S. K. Nayar, S. A. Nene, and H. Murase, *"Columbia object image library (coil 100),"* Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96, 1996.

- A. Torralba, R. Fergus, and W. T. Freeman, *"80 million tiny images: A large data set for nonparametric object and scene recognition,"* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 11, pp. 1958–1970, 2008. JOURNAL OF LATEX CLASS FILES, VOL. 13, NO. 9, SEPTEMBER 2015 9

- B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman, *"Labelme: a database and web-based tool for image annotation,"* International Journal of Computer Vision, vol. 77, no. 1-3, pp. 157–173, 2008.

- S. Lazebnik, C. Schmid, and J. Ponce, *"Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,"* in IEEE Conference on Computer Vision and Pattern Recognition, vol. 2. IEEE, 2006, pp. 2169–2178.

- J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, *"Sun database: Large-scale scene recognition from abbey to zoo,"* in IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2010, pp. 3485–3492.

- J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *"ImageNet: A large-scale hierarchical image database,"* in IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2009, pp. 248–255.

- M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, *"The pascal visual object classes challenge: A retrospective,"* International Journal of Computer Vision, vol. 111, no. 1, pp. 98–136, 2014.

- A. Torralba and A. A. Efros, *"Unbiased look at dataset bias,"* in IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2011, pp. 1521–1528.