Electrical Component Classification Using Machine Learning

A Kalyan

Electrical and Electronics Engineering

Amrita School of Engineering

Amrita Vishwa Vidyapeetham

Coimbatore, India

cb.en.u4elc22004@cb.students.amrita.edu

K Anil

Electrical and Electronics Engineering

Amrita School of Engineering

Amrita Vishwa Vidyapeetham

Coimbatore, India

cb.en.u4elc22066@cb.students.amrita.edu

K Praveen

Electrical and Electronics Engineering

Amrita School of Engineering

Amrita Vishwa Vidyapeetham

Coimbatore, India

[cb.en.u4elc22016@cb.students.amrita.edu](mailto:cb.en.u4elc22016@cb.students.amrita.edu)

Dr. M. Venkateshkumar

Electrical and Electronics Engineering

Amrita School of Engineering

Amrita Vishwa Vidyapeetham

Coimbatore, India

m\_venkateshkumar@cb.amrita.edu

***Abstract*—** *This paper introduces a real-time CNN-based classification and detection solution for Breadboard, LED (Light Emitting Diode), LCD (Liquid Crystal Display), Battery, and Arduino as common electronics components. This system trains a CNN using an image dataset and successfully classifies the images with over 92% accuracy on the test set. We propose a real-time detection framework for identifying and locating the components using OpenCV with a laptop webcam and Arduino IDE with ESP32 CAM upon live video streams. The model reads frames from the webcam, identifies regions of the object, and classifies a component with an associated confidence score greater than 70 percent while bounding boxes are drawn around detected objects in each frame. With a few minor misclassifications between meats that look like components, the system has great potential to be applied in electronics sorting and education for manufacturing automation. We plan to continue adding more data, improving the detection accuracy of the detector, and integrating object tracking for better performance.*

**KEYWORDS:**

Arduino, LED, Battery, LCD, Breadboard, Neural Networks, Convolution Neural Network (CNN), Recurrent Neural Network (RNN), ESP32 CAM, Webserver, Accuracy, Confusion Matrix, Graphical User Interface (GUI)

# I. INTRODUCTION

The importance of electronic & electrical components is rising drastically because we are in an era where technology, electronics, and systems have become our daily routine. These components are key to the functioning of some basic but highly sophisticated devices, from smartphones and laptops to industrial machinery that drives so much today in our world. In an era where the emergence of new industries and electronic devices make them more complex, time management, identification, and utilization have increasingly become essentials. Identifying electronic components accurately plays a key role in enabling efficient design, manufacturing, and maintenance processes for everything from consumer electronics to large-scale industrial systems. Identifying and differentiating between the components used in electronic systems, such as transistors, LCDs, capacitors, and microcontrollers is not only crucial for the smooth manufacturing process but also for resolving issues related to maintenance or debugging. Missed the application heads — You identified using the wrong components, which can cause System Failure, Cost Increase, and more downtime. While manual identification used to be the only way, it takes too much time and is prone to error because of a large variety of components becoming more specialized by an ever-expanding market. ML (Machine Learning) algorithms, specifically the use of Convolutional Neural Networks (CNNs), which is a popular model in deep learning that is highly effective at learning features can solve these problems by automating the identification of electronic components[1].

Image recognition is a subset of machine learning, and in this case, we have used (CNNs) that can learn to recognize the visual patterns and features organically manifesting into our electronic components: e.g. shape or color etc CNN[6]. By classifying components from images automatically and accurately, the efficiency of identification can be greatly increased. ML-powered electronic component recognition systems can dramatically speed up manufacturing processes and make assembly, testing also repair settings far more accurate putting a stop to the incorrect use of components which often leads to malfunction. In industries where components must be identified accurately and quickly to prevent costly delays or errors, this phase is particularly important. And on top of that, when we employ the machine-learning layer along with an electronic system our process becomes fundamental for other automated systems to improve upon smart manufacturing and possibly predictive maintenance. Integrating component recognition into workflows ensures maximal electronic system uptime, minimized downtime, and optimal device performance for manufacturers/technicians alike. Consequently, employing machine learning for component recognition is a stepping stone when it comes to bringing the electronics industry into modernity making a key part of this industrial vertical resilient and efficient in an ever more complex technological environment.

II. Methodology

The choice of Convolutional Neural Networks (CNN) as the neural networks in this project is justifiable by their unmatched predominance over other architectures for image recognition tasks and therefore, they were still highly efficient in performing models on datasets depicting electronic component visualization due to their visual characteristics. CNNs are especially suited to capturing complex hierarchical features in images, and electronic components possess a variety of visual characteristics. With the specific architecture of CNNs optimized to process images, they are good at identifying electronic components. CNNs have convolutional layers which help in extracting features from visual inputs.

1. Dataset Preparation: In the very first step, we need to divide and preprocess our data set which contains 6 categories of electrical components: Breadboard, light-emitting diode (LED), LCD, Battery, Arduino, etc. The data is zipped, and we need to unzip it along with formatting the dataset in order to feed it into the CNN model.

Extract Zip Files: A Python function extract\_zip\_files () is coded which unzips the dataset zip files into a structured folder catalog. Every folder is a class (for example: 'BREADBOARD', 'LED')

Load and Preprocess images: The function load\_images() is responsible for performing loading as well as image processing. The images are resized to (128, 128) px and then normalized by dividing the pixel values with 255.0, associated with a label that denotes its class. The dataset, where the data is stored in two arrays: images and labels (X, y)

The Data split: The dataset is divided into the training set (80%) and test set (20%) by using the train\_test\_split () function. Data augmentation is an effective means to overcome the lack of training data and avoid overfitting issues [5]. The training Set contains the data used to train the model, whereas the Test set contains the dataset on which that model is validated.

1. Model Development:

A Convolutional Neural Network model is built by using TensorFlow. It is a type of architecture that learns the patterns in images and can predict accurately what class each image belongs to

Model Architecture: A Sequential model is defined using the following layers:

1. Convolutional Layers: Models three convolutional layers (32, 64, and 128 filters) to locate character features in the image which is then followed by a max-pooling layer to down-sample spatial resolution. This method is used to find the low-intensity pixel from the hazy image by applying a filter [2-3]. We didn't use average pooling because it simply performs dimensionality reduction as a noise-suppressing mechanism [10].
2. Dense Layer: Here we flatten the given features and then here is a fully connected layer enabling 128 units followed by ReLU activation as before after which there are Dropout layers present in between to avoid overfitting. The model uses dropout to control the model complexity of the fully connected layer with 𝑝=0.5 to avoid the overfitting problem[7].
3. Output Layer: It has a SoftMax activation function, which provides probability distribution over the six class labels.
4. Compile and Train: Compile the model Adam optimizer, sparse categorical cross-entropy loss function, and accuracy as an evaluation metric. Adam (Adaptive Moment Estimation) is an optimization method that adapts learning rates for better efficiency[16]. The model is then trained for 20 epochs utilizing the training data where the test data will act as validation.

The functionality of each component has been illustrated below. CNN is a model that is gaining attention because of its classification capability based on contextual information[12].

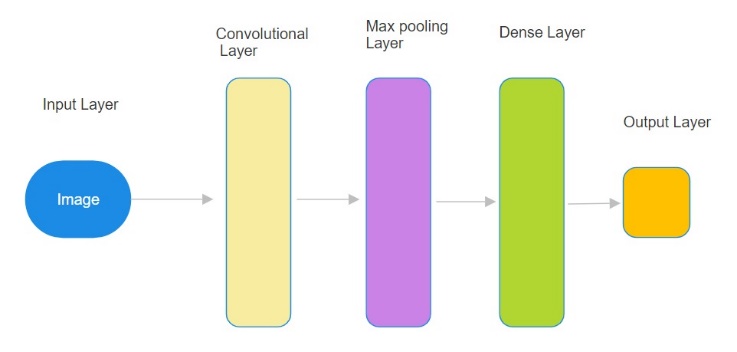


Figure 1: Basic Convolutional Neural Network (CNN) Architecture for Image Classification

1. Model Evaluation: After training, we validate the performance of our ML model using multiple measures, this plot shows training and validation accuracy (above) vs loss (below), to make sure that your model is learning something but doesn't overfit. To reduce reconstruction loss, the best idea is to drive its features large to reduce the artifacts caused by the Gaussian noise[11]. Confusion Matrix- Utilising all 6 classes a confusion matrix to look at classification performance across It creates a matrix image with the help of Matplotlib to understand how misclassification is happening. The final model is then evaluated using a test set and reports the accuracy as a fraction of examples it got right.
2. Real-Time Detection: We are using three types of modes for webcam usage. The UI is created when the model is executed, It pops up a UI where it shows three options- Laptop Webcam, Mobile Webcam, and ESP32 CAM webserver. The Laptop Webcam can be used directly, whereas the Mobile Camera needs some application that is 'The IP Webcam needs to be installed and a server will be created, and the IP address of that server is used in the code to use the Mobile Camera. For ESP32 CAM, the first thing we should do is write code in Arduino IDE which connects to the local wi-fi and gives a web server URL, which is used in the code for real-time processing.

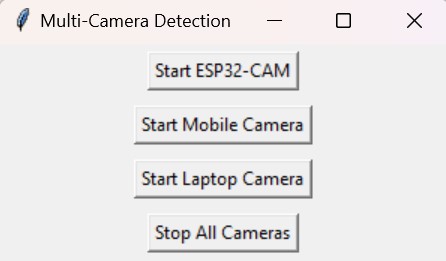


Figure 2: Multi-Camera Detection Interface for ESP32, Mobile, and Laptop Camera

III. Machine Learning Algorithm

##### Why CNN?

Convolutional Neural Networks (CNNs) are a subclass of deep learning, which belongs to the sub-family Learning from Machines. It was developed to deal with structured grid data such as images. It is designed to learn hierarchical representations of data, which makes it well-suited for image recognition. CNN Algorithm for Image Classification: An Overview of the CNN algorithm used in image classification tasks like this. The following is a detailed description of the algorithms used in this task:

CNN layers:

Input Layers: The first layer is the Input Layer as its name implies it will be the layer where we present our data to the model. The number of neurons in total corresponds to the number of features we have for our dataset, i.e. pixels that make up each image.

Hidden Layers:

The hidden layers — these take input from the Input layer and can be varied as per model/ dataset size. Typically, each layer can have many hidden neurons (often more than the number of features). The output of any layer is given by the matrix multiplication between the previous layers' output and its learnable weights, followed an a summation with its learnable biases that will be then passed through some type of non-linear activation function to give us these actual outputs.

Output Layer: The output of hidden layers is followed by a logistic function like sigmoid or SoftMax to convert the specified class-wise output into score probability.

##### Why Not RNN?

As far as Recurrent Neural Networks are concerned, these tend to perform poorly in object recognition tasks because of the type of data these are meant to work with and the type of patterns these are good at recognizing: Specifically Structured for Sequential Data: These look to be particularly useless for anything that is based on time or a sequence of events, because these do remember their previous outputs and consider those in the processing of current inputs. Such tasks tend to include speech, NLP, and time-based prediction tasks thus such models are memory-based and focused. Sequential tasks such as image recognition though concern spatial data as opposed to temporal data. This means that each pixel within a frame is related to its geographical counterparts rather than in the given time frame. As a result of this, spatial hierarchies and local patterns of images are amplified by CNNs.

Inability to Cope with Spatial Relationships:

Recognition of objects consists of finding various edges, shapes, textures, etc. in some portions of an image. Convolutional Neural Networks (CNNs) include convolutional layers that make it possible to learn such spatial relationships by filter application which scans the image in order to extract the features[4].

In contrast, Recurrent Neural Networks (RNNs) linearly take the inputs. They lack, however, a mechanism for understanding spatial hierarchies which is very vital when learning to detect and recognize objects. They would find it hard to understand advanced spatial characteristics like 'corners' or 'textures', which for example characterize objects in images. CNNs are special in capturing spatial features from images, whereas RNNs are special for handling sequential data and are not the go-to choice for analyzing static images.

Efficiency:

Convolutional Neural Networks (CNNs) tend to be optimized when it comes to the computation of images with large data. They are able to carry out convolutional operations over the image as a whole at once, thus rapidly acquiring pertinent local characteristics.

The recurrent neural network (RNN) architecture processes time-ordered inputs one after the other, therefore they would also need to scan the image row by row which slows them down considerably and makes them inefficient concerning imaging tasks.

The superiority of Convolutional Neural Networks in Object Recognition:

Using Local Receptive Fields: Since CNNs analyze a small area of the image at one time, it becomes simple to look for any distinguishing physical characteristics.

Weight Sharing: CNNs apply the same filters in different regions of the image which increases their productivity.

Pooling: CNNs allow for the decline of the spatial volume of the image with the retention of significant characteristics and decrease the time expenses.

At last, Convolutional Neural Networks (CNNs) are much appreciated in image recognition applications because they are capable of dealing with spatial relationships in images and retaining local information and its structure. Extending temporal recurrent neural networks to work in the image-based dimensionality is lastly impractical and ineffective courtesy of the 2-D nature of image recognition.

##### IV. Result

The Model successfully detects the electronic components in real-time with a frame showing the confidence rate. The training and validation Accuracy with respect to Epoch is plotted, it tells how accurate the model will be and the results are appreciable. The Training and Validation loss w.r.t to Epoch is also plotted, it tells how it learns. Loss is calculated by the gradient of the loss function as per the parameters [9].

Epoch = 20, Both graphs depict that the model is improving steadily in both accuracy and loss, These tell us how the the model is fitting also. The training Accuracy reaches around 90-95%, whereas the training loss reaches around 20%. The total dataset is converted into 41 – 42 modules and the time to train the model takes around 15 – 20 minutes.

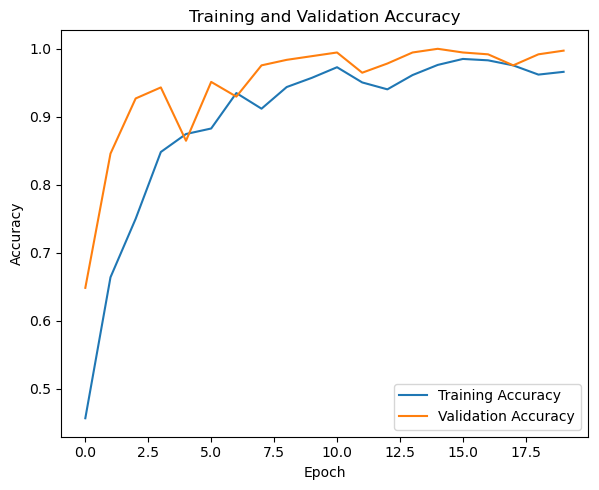


Figure 3: Epoch-wise Comparison of Training and Validation Accuracy of CNN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | Accuracy (%) | Precision (%) | Recall (%) | F1 Score (%) |
| RNN | 93.47 | 92.64 | 89.3 | 89.82 |
| CNN | 99.40 | 99.47 | 99.45 | 99.46 |

Table 1: Model Performance Metrics

The Accuracy of RNN is low when compared to the Accuracy of CNN, also the runtime for RNN is very high for training the model. The Model is overfitting for the CNN because the dataset provided is somewhat the same as the test data. During training of the CNN model, it reached peak accuracy at the 10th epoch but when trained using the RNN model it almost reached peak accuracy at the 20th epoch. The Confusion Matrix of the Model is also plotted. The confusion matrix (accuracy of the model) is the most basic, intuitive, and easiest way to measure models' accuracy [8].

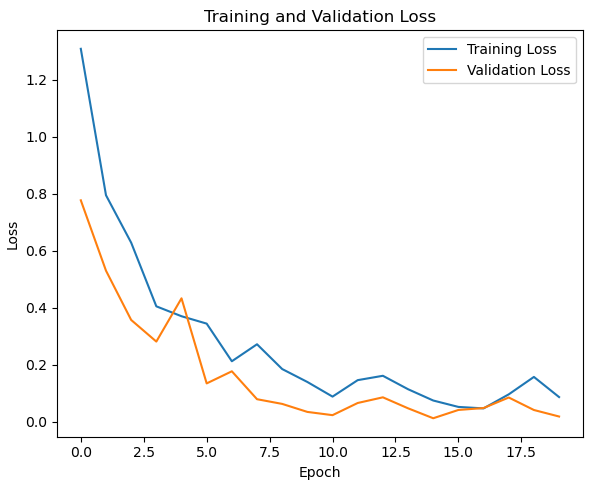


Figure 4: Epoch-wise Comparison of Training and Validation Loss of CNN

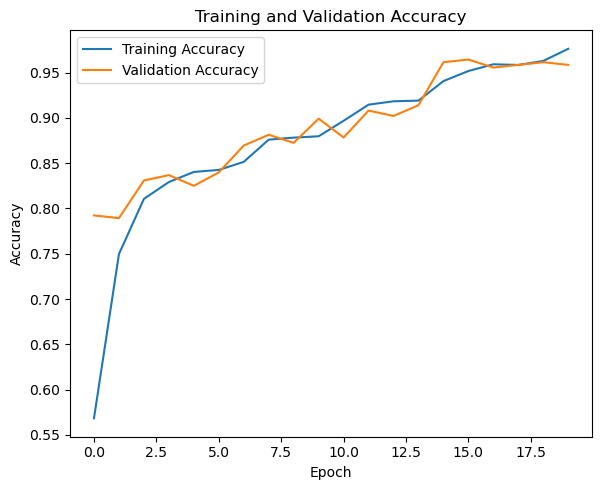


Figure 5: Epoch-wise Comparison of Training and Validation Accuracy of RNN

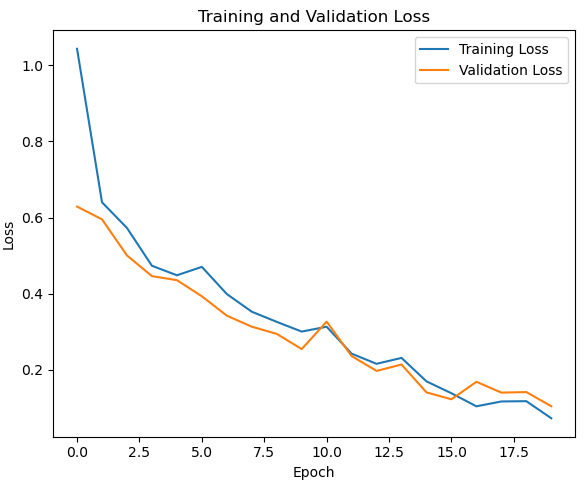


Figure 6: Epoch-wise Comparison of Training and Validation Loss of RNN

Confusion Matrix Analysis tells how well did the model classify each class. It visualizes a confusion matrix with a performance graph of the classifier. Predicted labels lie on a horizontal axis whereas true classes lie on a vertical axis which is plotted on six classes: Breadboard, LCD, Resistor, Battery, Arduino, and Default.

Diagonal values are true classifications, and non-diagonal values represent misclassifications.

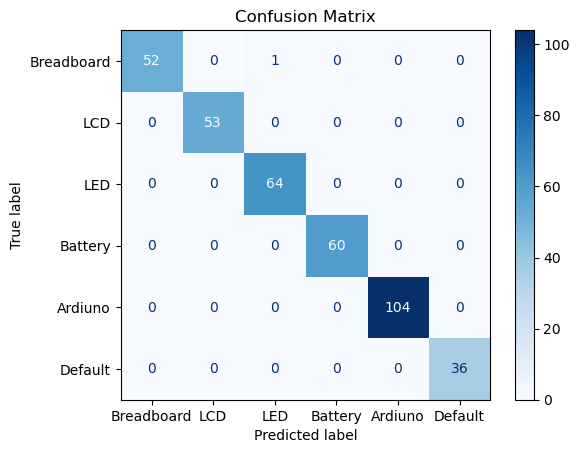


Figure 7: Confusion Matrix of Predicted vs. True Labels of CNN

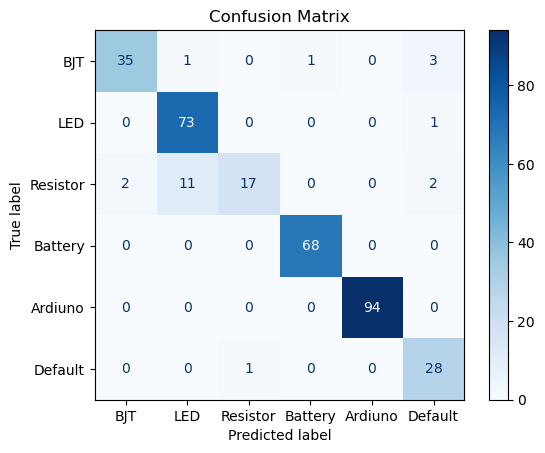


Figure 9: Confusion Matrix of Predicted vs. True Labels of RNN

Overall, the model is pretty good with minor errors in it, and mostly the errors occur for the "LCD" and "Resistor" classes. The color gradient shows the distribution of prediction, and the darker color represents the higher number of correct predictions. This confusion matrix depicts how well the model performs. For 59 instances of "Breadboard," the model gives no false positives. In a nutshell, the LCD class makes correct predictions in 55 instances while wrongly predicting only with 1 instance of the "Battery" class. The Resistor class produces 29 correct predictions with it making horrible predictions for the 2 instances as "Default." The Battery class's predictions are pretty sharp with 61 instances where the model got correct. For the Arduino class, the model performed excellently, with 95 correct predictions without even a single error. Lastly, the "Default" class gives 26 correct predictions. Generally, the confusion matrix provides good classification accuracy without much error, especially in classes "LCD" and "Resistor".

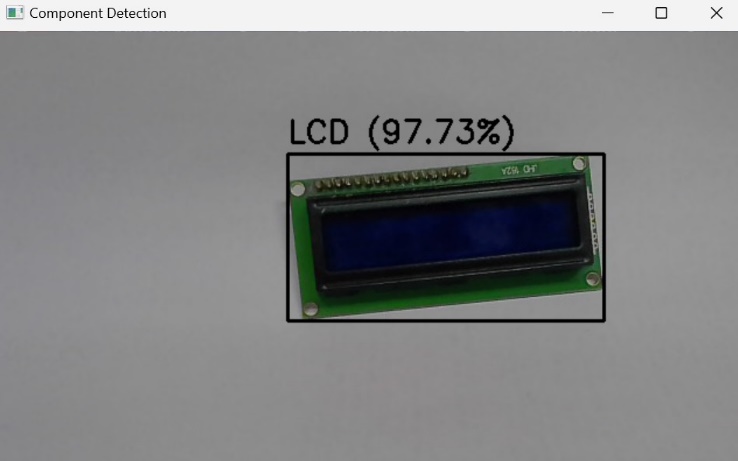


Figure 8: Real-Time CNN Component Detection: LCD

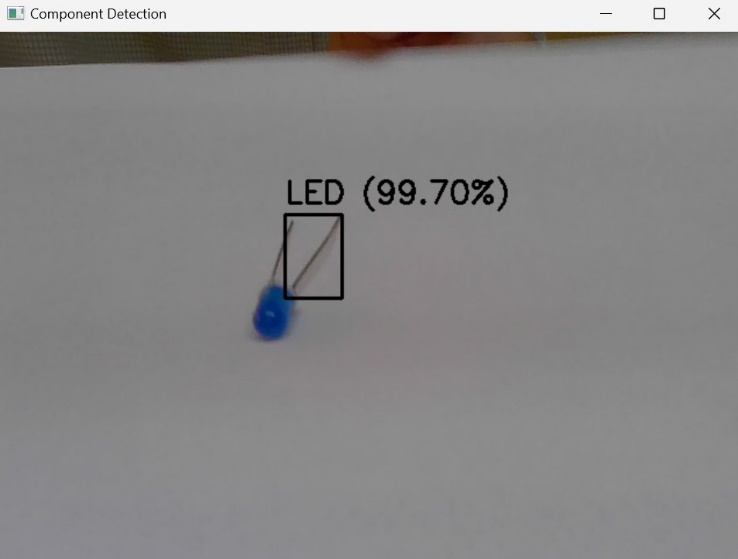


Figure 9: Real-Time CNN Component Detection: LED



Figure 10: Real-Time CNN Component Detection: Arduino

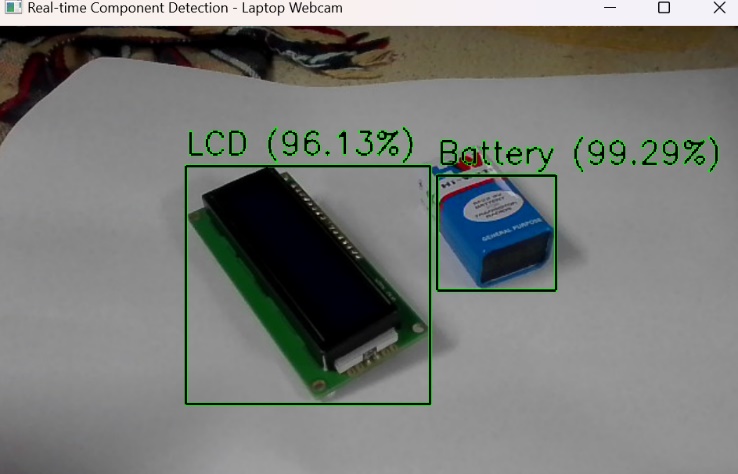


Figure 11: Real-Time CNN Component Detection: LCD & Battery

V. Conclusion

The overall ranking-Mobile Camera > Laptop Camera > ESP32 Demonstrates the high image quality and processing capability of mobile devices, making it possible for the CNN model to recognize electronic components with a very high degree of accuracy. Laptop cameras are good but fall short of the bar because they have low resolution, while the ESP32 CAM is the least effective since it was designed for lightweight applications and has poor imaging capabilities. The Phone camera is around 50 MP, whereas the Laptop camera is 12 MP, but the ESP32 CAM is 2MP. The model is understood well and is in high resolution.

Convolutional Neural Networks (CNNs) are much appreciated in image recognition applications because they are capable of dealing with spatial relationships in images and retaining local information and its structure. Extending temporal recurrent neural networks to work in the image-based dimensionality is lastly impractical and ineffective courtesy of the 2-D nature of image recognition. The model with RNN is very much inaccurate and the model with CNN is very much accurate and can easily recognize the object.

##### References

1. Wang Zhiqiang, Liu Jun, “A Review of Object Detection Based on Convolutional Neural Network”, 36th Chinese Control Conference, Volume: 7, Issue: 26-28 July 2017, pp 11104- 11109.
2. Xiang, Y., et al. (2013). Hazy image enhancement based on the full-saturation assumption. Multimedia and Expo Workshops (ICMEW), 2013.
3. C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” J. Big Data, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.
4. C. Junliang, "CNN or RNN: Review and Experimental Comparison on Image Classification," 2022 IEEE 8th International Conference on Computer and Communications (ICCC), Chengdu, China, 2022, pp. 1939-1944, doi: 10.1109/ICCC56324.2022.10065984.
5. E. Soylu and İ. Kaya, “Classification of Electronics Components using Deep Learning”, SAUCIS, vol. 7, no. 1, pp. 36–45, 2024, doi: 10.35377/saucis...1391636.
6. Radhesyam Vaddi, Prabukumar Manoharan, Hyperspectral image classification using CNN with spectral and spatial features integration,

Infrared Physics & Technology,Volume 107,2020,103296,

1. ISSN 1350-4495, H. M and S. M.N, “A Review on Evaluation Metrics for Data Classification Evaluations,” Int. J. Data Min. Knowl. Manag. Process, vol. 5, no. 2, pp. 01–11, 2015, doi: 10.5121/ijdkp.2015.5201.
2. Y.LeCun et al., Gradient-based learning applied to document recognition, Proc. of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
3. Digit Recognition Using Machine Learning and Convolutional Neural Network “by Snehal Kalbhor Ashwini M. Deshpande November 2014
4. K. K. Singh, P. Mishra, and K. Singh, “A comprehensive review on convolutional neural network,” in Proceedings of the 3rd International Conference on Intelligent Computing and Control Systems (ICICCS 2018), IEEE, 2018, pp. 722-728. [Online]. Available: https://doi.org/10.1109/ ICCONS.2018.866321
5. Kartik Audhkhasi, Osonde Osoba, Bart Kosko, "Noise-enhanced convolutional neural networks", 2015.
6. Sakshi Indolia, Anil Kumar Goswami, S.P. Mishra, Pooja Asopa, “Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach”, International Conference on Computational Intelligence and Data Science(ICCIDS), Volume: 5, Issue: May 2018, pp 679-688
7. K. Tong and Y. Wu, "Deep learning-based detection from the perspective of small or tiny objects: A survey", *Image Vis. Comput.*, vol. 123, Jul. 2022.
8. W. Wang, X. Wu, X. Yuan, and Z. Gao, "An experiment-based review of low-light image enhancement methods", *IEEE Access*, vol. 8, pp. 87884-87917, 2020.
9. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 770-778, Jun. 2016.
10. Zhang, Z. (Ed.). (2019). Improved Adam Optimizer for Deep Neural Networks. 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS).