A PyTorch-Based Deep Learning Approach for Enhanced Liquid Level Detection in Industrial Environments

Abhinav Narayan, Charan DK, Sneha Elizabeth Saji, Tianyang Fang and Jafar Saniie
ECASP Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago, IL, USA

***Abstract*— The accurate detection of liquid levels is of paramount importance across various industries, including pharmaceuticals, beverages, and chemicals. Traditionally, monitoring liquid levels within containers has relied on manual procedures or basic sensor technology, which often encounters challenges related to precision and speed. However, as the field of computer vision advances, there is a growing interest in leveraging more advanced techniques to overcome these limitations and enhance liquid-level monitoring. In this context, this research proposes the utilization of PyTorch, a powerful open-source deep learning framework, to tackle the intricacies associated with liquid-level monitoring. By transitioning from traditional computer vision methods to advanced deep learning techniques facilitated by PyTorch, this study aims to significantly improve the accuracy, efficiency, and reliability of liquid-level detection across industrial applications.**

Keywords—Computer Vision, Deep Learning, Convolutional Neural Networks (CNN), Loss function, accuracy, precision, recall

I. Introduction

The incorporation of PyTorch [1] marks a significant departure from conventional methodologies in liquid-level monitoring, ushering in a technologically advanced solution poised to elevate both accuracy and efficiency. PyTorch is anticipated to facilitate intricate operations such as perspective transformation, thresholding, and edge detection, thereby enabling precise identification of liquid levels within containers. This practical approach not only addresses immediate challenges in liquid level detection but also holds the potential to advance quality control practices and operational optimization across diverse industrial sectors.

To elucidate the rationale behind the adoption of PyTorch for liquid-level detection, it is imperative to delve into the specific methodology employed in this study. The proposed system operates by training and testing data using three convolutional network architectures: Densely Connected Convolutional Neural Networks (Dense-Net), Residual Neural Networks (Res-Net), and Visual Geometry Group Neural Networks (VGG-Net).

The process followed for each of these three models is as follows: Initially, data augmentation is performed on the images, followed by training and testing. The metrics utilized to evaluate the efficacy of these models include accuracy, categorical cross-entropy loss, precision, and recall.

There is a lot of contemporary literature related to the topic of liquid-level detection. For detecting liquid levels in amber glass bottles [2], images obtained were taken at a fixed distance in ambient lighting conditions. Once data was obtained, images were converted to grayscale, followed by 7x7 Gaussian blurring, and then edge detection with a threshold starting with pixel intensity value of 55, and ending at pixel intensity value of 110. Morphological operations were then performed to obtain specific boundaries to determine whether the bottle was underfilled, overfilled, or completely filled. After these steps, a square was drawn around the boundary of interest, showing the liquid's presence. Inside this square, vertical lines were drawn to measure whether the bottle was completely filled, over-filled, or underfilled.

Due to the impracticality of the current approach, an automated process was introduced to analyze different bottle levels without geometrical dependencies. This implementation detected the amount of viscous food [3] in a glass bottle without many dependencies. The authors used VGG-16, a 16-layer convolutional neural network. In this architecture, the authors acquired the image using a fixed lighting approach to obtain clear samples of dark viscous food. The images were then converted to grayscale and enhanced with Sobel and Canny Edge detection to distinguish liquid and food in the bottle. Morphological operations (dilation and erosion) were performed to minimize food particle size, followed by image segmentation to determine liquid level boundaries. Outputs were compressed from 1158×2489 to 128x128 for faster model training and testing. Data augmentation was applied to provide diversity to the dataset.

Upon training and testing, the obtained accuracy was 98.3%, with around 10% loss function. However, the required setup for obtaining images was fixed, potentially causing issues with data taken at different angles and lighting conditions. Therefore, designing a system that accommodates data captured in various lighting conditions is crucial.

Seeing this implementation is why we have implemented the VGG-16 model. This following implementation shows how to implement an AI system using EDGE[4], HD cameras, and networking to detect bottle levels – filled or empty. This is done by connecting an HD camera to an EDGE processor device. This processor communicates with the server to update the results of whether the bottle is empty or completely filled. The data given as input is that bottles with no caps are considered empty, and the bottles with caps are completely filled using a 2-stage CNN. Again, it is also to be noted that the camera for capturing the bottles was at a fixed height.

This upcoming implementation makes use of image segmentation schemes, followed by determining the area, height, width, and extent of the bottle to determine whether it is a completely filled or underfilled bottle. Another transform performed once image segmentation was completed is the Hough transform [5]. This transform determines the edges of an image. Based on these edges, the maximum, minimum, and overall level of the bottle is obtained. Once done, the results obtained from the Hough transform and the area of the given object after image segmentation are fed to a decision tree classifier. Based on these results, the bottles are classified as completely filled or underfilled. However, as Hough Transform is computationally expensive, we didn't go forward with implementing this approach.

This next approach evaluates datasets using a sequential net, Res-Net 50, Mobile-Net [6], and VGG-19 model to train and test data. The data was preprocessed by converting the original input to size 224x224. In addition to this, the following metrics were used in the analysis of the performance of each model – precision, recall, f1-score, and accuracy. From these scores, it was observed that the sequential-net model performed the best with 97% accuracy. The authors did not mention the use of a loss function; hence, sequential net was not included for the final comparison in this paper.

II. Level Detection System Design

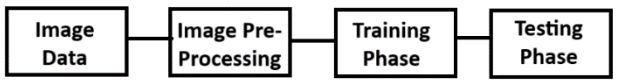


Fig. 1. System block diagram

Fig. 1 shows the system design. The image data is obtained, then the images are preprocessed using data augmentation, followed by training and testing using convolutional neural networks. In each CNN, there are some basic building blocks worth noting, which are discussed in depth in this paper. These basic building blocks include activation function, loss function, and optimization functions. Data [7] used for this implementation contained 486 images. 308 were completely filled water bottles, 139 were half-filled water bottles, and 39 were over-filled water bottles. Image Preprocessing is performed prior to training or testing the data. The technique used for this implementation is data augmentation.

Data augmentation is the process of making modifications to the dataset in order to increase the number of samples in the data to train and test. Plus, an added advantage of data augmentation is it introduces diversity to the training data to avoid overfitting. Some examples of implementing data augmentation include reducing the size of the image to reduce training and testing time, rotating the image by 45 degrees, and changing hue. For this implementation, we reduced the size of the image from the original size to 224x224 and rotated the image by 45 degrees.

III. Convolutional Neural Networks

Convolutional neural networks are a class of artificial neural networks used for various tasks like liquid-level classification. We have opted to use convolutional neural networks because they can detect patterns in large datasets. Some of the models we implemented are:

A. Residual Neural Networks (Res-Net)

The skyscraper analogy for Res-Net [8] highlights its innovative approach to building deep neural networks by introducing residual connections. As networks grow deeper, they tend to suffer from the vanishing gradient problem, where gradients become too small for effective learning in initial layers. Res-Net addresses this by adding shortcut connections that skip one or more layers. Fig. 2 shows the architecture of the Res-Net model.

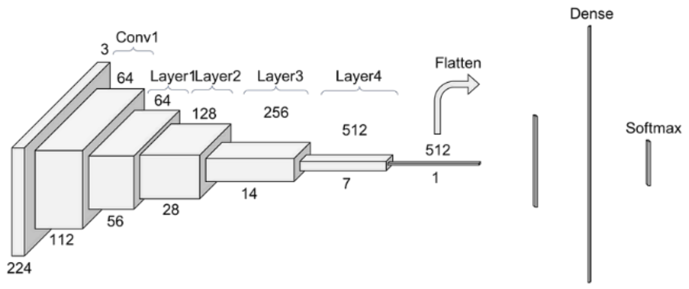


Fig. 2. Res-Net architecture implementation

Res-Net makes it feasible to train networks with unprecedented depth—over 100 layers—while maintaining performance gains. The residual connections act as alternative paths for gradient flow during backpropagation, mitigating the vanishing gradient problem and enabling the training of very deep networks without degradation in performance.

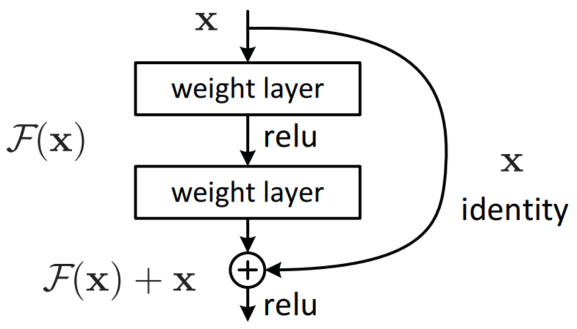


Fig. 3. Residual block with skip connections

Residual Blocks: The heart of Res-Net, these blocks feature two paths: the normal path through convolutional layers and the shortcut path that skips (see Fig. 3) one or more layers. The shortcut paths carry the input directly to the output of the block, allowing the network to learn the residual of the input to the output, rather than the full output. For this implementation, we have used Res-Net 50.

B. *Visual Geometry Group Neural Networks (VGG-16)*

VGG-Net's [9] design philosophy emphasizes depth and uniformity, using an architecture with repetitive blocks of convolutional layers. Each block in VGG-Net consists of several convolutional layers using small 3x3 filters, followed by a max-pooling layer. This uniformity and repetition, akin to an artist meticulously working on a canvas, allow the network to develop an increasingly complex and deep representation of the input data.

Convolutional Layers: VGG-Net uses multiple stacks of convolutional layers with small 3x3 filters, which are the smallest size, to capture the patterns of left/right, up/down, and center. This approach allows for capturing finer details in the input images across the network's depth.

Pooling Layers: Followed by convolutional layers, max-pooling layers reduce the spatial dimensions of the feature maps, condensing the information and reducing computational requirements.

Fully Connected Layers: After several blocks of convolutional and pooling layers, the network concludes with fully connected layers that perform the final classification based on the features extracted by the convolutional layers.

Depth and Performance: Despite its simplicity, VGG-Net architecture has shown remarkable success in image recognition tasks, proving the effectiveness of deep networks with repetitive structures. In this paper, VGG-Net 16 was implemented (see Fig. 4).

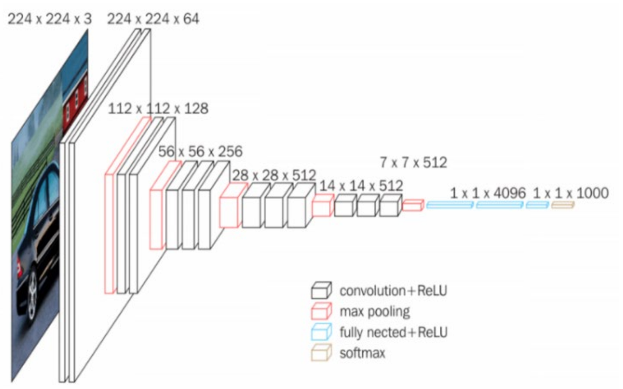


Fig. 4. VGG-Net architecture

C. *Densely Connected Convolutional Neural Networks (Dense-Net)*

In this paper, we have used Dense-Net 121, which is a 121-layer architecture. Dense-Net [10] architecture (see Fig. 5) facilitates information flow between layers in a deep neural network. Traditionally, layers in a convolutional neural network (CNN) pass information in a linear sequence from one layer to the next. Dense-Net changes this paradigm by connecting each layer to every other layer in a feed-forward fashion. In a Dense-Net, the output of each layer is concatenated to the inputs of all subsequent layers, creating a highly interconnected system. This means that the first layer output is fed directly into the second layer, the combined outputs of the first and second layers are fed into the third, and so on. This mechanism ensures that each layer receives the "collective knowledge" generated by all previous layers, enriching the feature maps with diverse and comprehensive information.

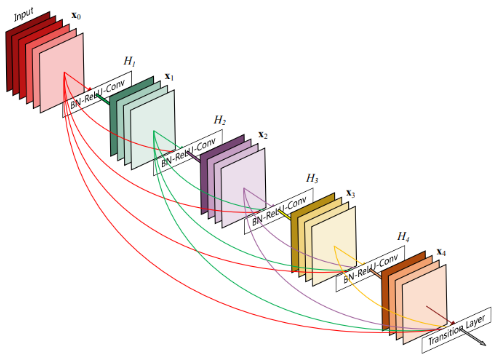


Fig. 5. Dense-Net architecture

The dense connectivity pattern significantly reduces the problem of vanishing gradients, as gradients from the loss function can flow directly to most layers in the network. Moreover, it enhances feature reuse, making the network more parameter-efficient compared to its counterparts. Dense-Net ability to leverage the full potential of the network enables it to achieve excellent performance with fewer parameters, reducing the risk of overfitting on smaller datasets.

There are two main blocks for implementing Dense-Net models:

Dense Blocks: The core of Dense-Net consists of dense blocks, where each layer receives concatenated feature maps from all preceding layers as input, and its own feature maps are then passed on to all subsequent layers. This setup ensures that each layer has access to all the raw and processed information from the input to the current layer.

Transition Layers: Between dense blocks, transition layers perform convolution and pooling operations to reduce the dimensionality of the feature maps, preparing the data for the next dense block. This helps to manage the model's complexity and computational demands.

IV. CNN FUNCTIONS

In each convolutional layer, activation functions are used to introduce non-linearities to the model. This will make it easier for the neural network to learn the images. Some that were used in these deep neural networks are as follows:

A. *Rectified Linear unit (ReLU)*

ReLU[11] is a popular activation function used in deep learning models. It introduces non-linearity to the neural network, allowing it to learn complex patterns and relationships in the data. This function $f(x) = \max(0, x)$ is used in the hidden layers.

B. *Softmax activation function*

A softmax classifier is a type of classification model used in the context of neural networks. It's commonly used when the task is to classify instances into multiple classes (in this case 3 classifications). The softmax function is used as the output layer activation function in neural networks for multi-class classification tasks. It transforms the raw output scores from the previous layer into probabilities corresponding to each class. The output of the softmax function is a probability distribution over the classes, with each class having a probability value between 0 and 1 and the sum of probabilities across all classes equaling 1. Mathematically, the softmax function is defined as follows:

$$p(y = j|x) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

$p(y = j|x)$ is the probability that training sample x belongs to class j is the raw score for class j. K represents the total number of classes. The softmax function takes the raw scores and normalizes them into a probability distribution. The class with the highest probability is then predicted as the output class.

C. Loss Function

Categorical cross-entropy [12] is a commonly used loss function in multi-classification tasks. It measures the dissimilarity between the true distribution of class labels and the predicted probability distribution outputted by the model.

Here's how categorical cross-entropy works:

1. True Distribution: Each training example is associated with a ground truth label, represented as a one-hot encoded vector. In this case, there are three classes: class 0 indicates the water bottle is fully filled, class 1 stands for a half-filled bottle, and class 2 indicates the water bottle is overfilled.

2. Predicted Probability Distribution: The model produces a probability distribution over classes for each example. This distribution is typically obtained by passing the raw output of the model through a softmax function, which converts the logits into probabilities, making sure that each probability sum equals 1.

3. Loss Calculation: Categorical cross-entropy computes the loss by comparing the true distribution (one-hot encoded vector) with the predicted probability distribution. It penalizes the model more heavily for predicting probabilities that diverge significantly from the true distribution.

The mathematical formula for categorical cross-entropy loss is as follows:

$$Loss = -\sum_{i}^{N} y_i \log(p_i)$$

where N represents the number of classifications. N = 3 in this case. y_i represents the true probability of classification i. p_i represents the predicted probability of classification i. The loss is summed over all classes, and the negative sign makes sure that the loss is minimized during training. Categorical cross-entropy loss provides a measure of how well the predicted probabilities align with the true distribution of class labels, guiding the training process to produce more accurate classification models.

D. Optimization Function

Stochastic Gradient Descent [13] (SGD) is an optimization algorithm used to train machine learning models, especially deep neural networks. Unlike traditional gradient descent, which computes the gradient of the loss function with respect to all training examples before updating the model parameters, SGD updates the parameters incrementally, one example at a time, or in small batches.

V. Experiment Results

Table I shows the testing results for each of the three models mentioned. Fig. 6 shows the training and testing accuracy graph of the best case – Dense-Net 121. From Table 1, it is clear that Dense-net 121 performed the best – the loss function clearly indicates 18%. This can be due to continuous concatenation of the features from the previous layers. From Table 1, it can be clear that Resnet 50 and VGG 16 were over-fitted. Training accuracies for both Res-net and VGG were high at 1.0 for both. This means 100% accuracy. Fig. 7 shows the predicted outputs of the Dense-Net 121 model.

TABLE I. TESTING RESULTS

Model	**Accuracy**	**Precision**	**Recall**	**Loss**
Dense-Net 121	0.93	0.948	0.93	0.18
Res-Net 50	0.83	0.84	0.93	0.4
VGG 16	0.86	0.86	0.93	0.54

VI. Conclusion

The advancements in liquid-level detection are steadily progressing. However, the necessity for a larger dataset is evident, given the limited number of images in the current dataset. This shortage likely contributed to over-fitting issues, alongside fewer concatenations found in Res-Net and VGG-Net. Moving forward, exploring other methods, such as transformer architectures for liquid-level detection, holds promise.

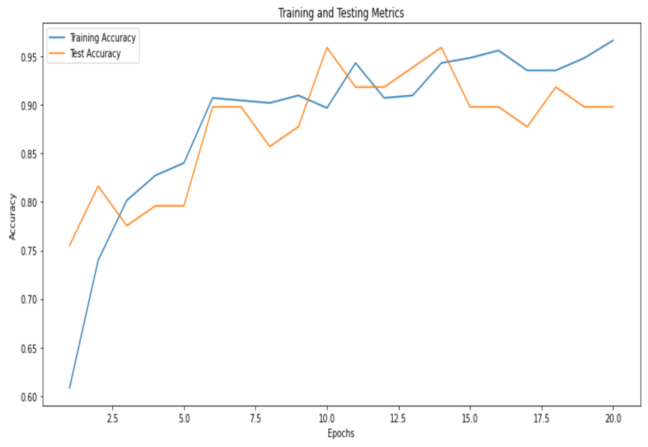
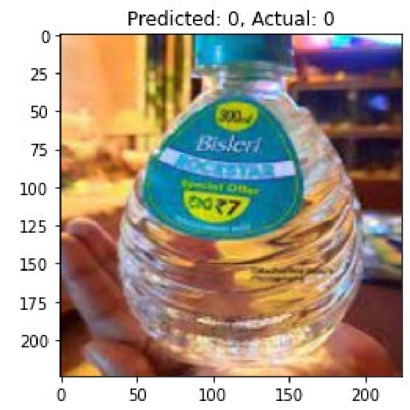
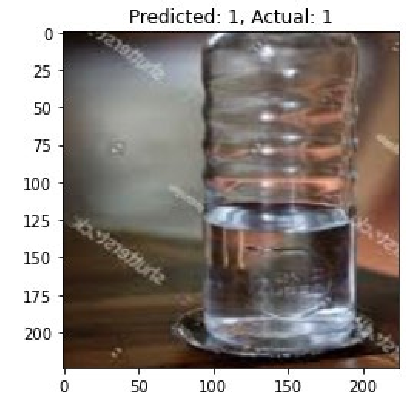


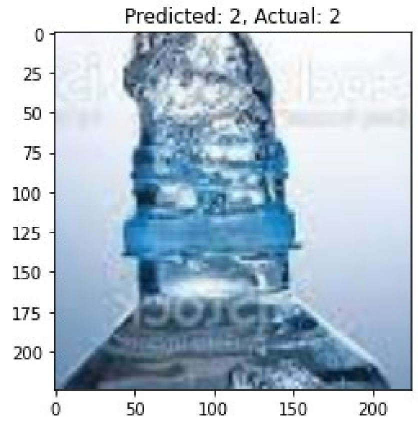
Fig. 6. Dense-Net 121 training and testing accuracy plot



(a): Predicted fully filled bottle output



(b): Predicted half filled bottle



(c): Predited over-filled bottle

Fig. 7. Dense-Net output classified images

References

[1] PyTorch website: https://pytorch.org/docs/stable/index.html

[2] Mikhael Anthony A. Felipe, Tanya V. Olegario, Nilo T. Bugtai, and Renann G. Baldovino, "Vision-based Liquid Level Detection in Amber Glass Bottles using OpenCV," in 2019 7th International Conference on Robot Intelligence Technology and Applications (RiTA) November 1 – 3, 2019, Daejeon, Korea

[3] Changfan Zhang, Dezhi Meng, and Jing He, "VGG-16 Convolutional Neural Network-Oriented Detection of Filling Flow Status of Viscous Food"

[4] Leendert Remmelzwaal, "Object Detection and Tracking for Crate and Bottle Identification in a Bottling Plant Using Deep Learning."

[5] Nor Nabilah Syazana Abdul Rahman, Norhashimah Mohd Saad, Abdul Rahim Abdullah, "Shape and Level Bottles Detection Using Local Standard Deviation and Hough Transform," International Journal of Electrical and Computer Engineering (IJECE) Vol.8, No.6, December 2018, pp. 5032~5040 ISSN: 2088-8708, DOI: 10.11591/ijece.v8i6.pp5032-5040

[6] Oluwaseun O. Martins, Mahdi M. Abdulhamid, Mariam O. Lawal, Osifalujo T. Olugbenga and Orimolade E. Okikiola, "Development of a Sequential Neural Network Model for Bottle-Fill Level Detection and Classification".

[7] Dataset for water bottle level detection:https://www.kaggle.com/datasets/chethuhn/water-bottle-dataset

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun," Deep Residual Learning for Image Recognition"

[9] Karen Simonyan, Andrew Zisserman,"Very deep convolutional networks for large scale image recognition"

[10] Gao Huang, Zhuang Liu, Laurens van der Maaten, "Densely Connected Convolutional Networks"

[11] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li,"Empirical Evaluation of Rectified Activations in Convolution Network"

[12] Zhilu Zhang, Mert R. Sabuncu, "Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels"

[13] Sebastian Ruder, "An overview of gradient descent optimization algorithms"

[14] Evaluation: From Precision, Recall and F-Measure To ROC, Informedness, Markedness & Correlation