PHENIKAA UNIVERSITY
FACULTY OF COMPUTER SCIENCE

Final-term Report

PizzaRco: AI application to recognize the status of Pizza

Name	ID	Class
Vuong Tuan Cuong	21011490	Computer Science
Nguyen Le Phuong Linh	22014068	Information Technology

Supervisor:

PhD. Pham Tien Lam

Ha Noi, July 11, 2024

Contents

List of Figures

List of Tables

Abstract

In the domain of image recognition, numerous advancements have been made in the application of AI for various tasks. In this project, we explore the potential of leveraging advanced methods for the task of Pizza condition recognition. We built a comprehensive dataset related to Pizza images and observed significant potential in this field. By employing state-of-the-art techniques such as YOLOv8 for prelabeling and fine-tuning the YOLO model, we achieved effective accuracy with a more detailed class structure compared to using the standard YOLO model alone. Additionally, we integrated the fine-tuned EfficientNet model for our multi-label classification problem. This study presents the current problem statements and our method of addressing them, detailing our approach and the results achieved. Ultimately, our implementation demonstrates the efficacy of combining these advanced techniques to enhance the accuracy and efficiency of Pizza condition recognition. Here is publication code: https://github.com/Phenikaa-University/cv-finalterm.git.

1 Introduction

Ensuring accuracy and efficiency in food service is critical, particularly in the context of serving pizza. Customers expect to receive the exact pizza they ordered, and manual identification can lead to mistakes, resulting in customer dissatisfaction and significant time loss for staff. The global pizza industry is rapidly expanding, with the US market generating annual revenues of $65.2 billion in 2022 according to IBISWorld [2]. Similarly, the number of pizza outlets in Vietnam has seen considerable growth in recent years, reflecting the popularity of this food item. This growth demands an effective solution for rapid and accurate pizza identification.

Our pizza recognition **PizzaRco** model addresses this need by leveraging image processing and artificial intelligence technologies to automatically identify various types of pizzas through images. This not only saves service time but also eliminates the possibility of order errors, thereby enhancing the customer experience. Additionally, our model has potential applications in automated pizza production factories, where it can ensure product quality by detecting issues such as burnt areas, missing toppings, or other defects, thus enhancing the consistency and quality of the final product.

The primary objective of this project is to develop a model capable of accurately identifying different types of pizzas from images, regardless of viewing angles and lighting conditions. We aim to address several specific challenges:

- Accurate Identification: Ensuring the model can accurately recognize different pizza types based on image inputs, detecting key features like crust type, toppings, and size.
- Robustness: Training the model to perform reliably under various viewing angles and lighting conditions.
- Multi-Label Classification: Building a multi-label classification model capable of identifying different types of pizza defects simultaneously.
- Efficiency: Automating the pizza identification and classification process to save time and reduce errors in service.
- Versatility: Applying the model across different sectors, including food service and automated production lines.

To address these challenges, we have implemented the following solutions:

- Dataset Construction: We constructed a comprehensive dataset containing 81,526 pizza images, each labeled with possible defects.
- Prelabeling with YOLOv8: We employed the YOLOv8 model for prelabeling to distinguish images containing pizzas from those without. This model, pre-trained on the COCO dataset, allowed us to filter out irrelevant images effectively.
- Data Augmentation: Post prelabeling, we cropped images to focus on the pizza regions, enhancing the model's ability to learn relevant features.

- Fine-Tuning EfficientNet: For multi-label classification, we fine-tuned the EfficientNet model, incorporating modifications to the fully connected layers to handle multiple labels.

- Deployment: We developed a web interface using FastAPI, enabling users to upload images and receive real-time predictions on pizza condition.

2 Related Work

Real-time object detection has seen significant advancements over recent years, with various models being proposed to enhance both speed and accuracy. Among these, the YOLO (You Only Look Once) family of models has gained widespread recognition for its real-time performance [3]. The latest iteration, YOLOv8 [1], developed by Ultralytics, introduces numerous architectural improvements that contribute to its superior efficiency and precision in real-time detection tasks. YOLOv8 employs a new C2f module and an anchor-free model design, allowing it to achieve higher accuracy and faster inference times compared to its predecessors and other object detection frameworks. This makes it particularly suitable for applications requiring immediate feedback, such as autonomous driving and real-time surveillance systems.

In the realm of image classification, several models have set benchmarks in terms of accuracy and computational efficiency. Convolutional Neural Networks (CNNs) [4] like AlexNet [5], VGG [6], and ResNet [7] have been foundational, each introducing new techniques to improve deep learning performance. However, EfficientNet [8], introduced by Google AI, has revolutionized this field by proposing a novel scaling method that uniformly scales all dimensions of depth, width, and resolution using a compound coefficient. EfficientNet models, particularly EfficientNet-B0 to B7, achieve state-of-the-art accuracy on ImageNet with significantly fewer parameters and FLOPS, making them ideal for resource-constrained environments. This efficiency is critical for deploying advanced AI solutions in real-world applications where computational resources may be limited.

While numerous studies have explored object detection and image classification, specific applications such as pizza recognition remain under-researched. Existing projects in food recognition often focus on broader categories and lack the granularity required for detailed defect detection in specific items like pizza. Our research aims to bridge this gap by integrating multiple advanced AI techniques in computer vision, combining the robustness of YOLOv8 for initial object detection with the precision of EfficientNet for multi-label classification. This integrated approach promises to deliver superior results in identifying various conditions of pizza, providing a comprehensive solution that enhances both accuracy and efficiency in food quality assessment.

3 Dataset

3.1 Data collection

As shown in Figure 1, to conduct data collection, we built a tool that automatically collects data using the Search API tool - SerpAPI - developed by Google's research team. In detail, we will proceed to pass on a key: pizza, to

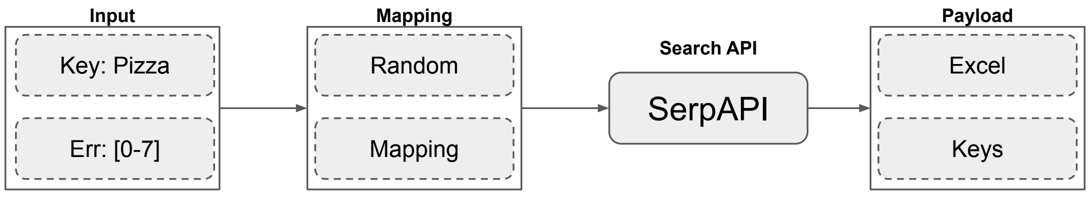


Figure 1: Pipeline for proposed crawl tool.

search for photos of pizza. Then we pass on a pair of random numbers from 0-7, which is equivalent to the errors mentioned later. After mapping the errors, we save the path of the image and save it on the AWS S3 platform. In addition, we also create an excel file to store the information of the data set. In this study, we have compiled a comprehensive dataset for the task of pizza condition recognition. The data collection process involved curating a list of pizza images, along with detailed labels for various types of defects. The dataset was organized in an Excel file containing the following attributes:

- id: Unique identifier for each image.
- img_url: URL link to the pizza image.
- store: Store information where the image was captured.
- oven: Oven type used for baking the pizza.
- exception: Any exceptions noted during data collection.
- create_at: Timestamp of image creation.
- update_at: Timestamp of last update.
- v__: Version information.
- error: General error indicator.
- err_adv_edge: Advanced edge defect.
- err_edge: Basic edge defect.
- err_shape: Shape defect.
- err_baking: Baking defect.
- err_size: Size defect.
- err_topping: Topping defect.
- err_ferment: Fermentation defect.
- version: Dataset version.
- label_err: Binary label indicating the presence of an error.

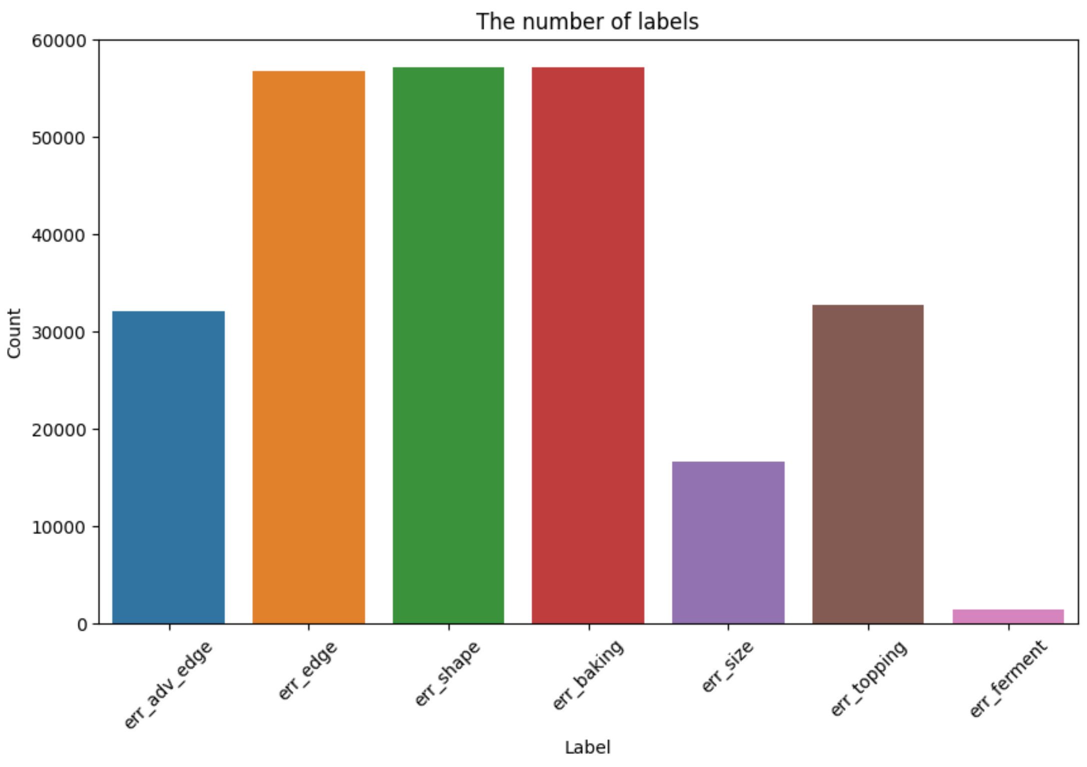


Figure 2: Distribute the total number of errors.

3.2 Overview of our dataset

As mentioned in Section 3.1, our dataset includes more than 80,000 of raw data images originally collected through the Search API engine stored on AWS S3. In general, our dataset consists of 18 attributes, of which 7 are different fault states of Pizza. However, because the problem we determine is to identify many errors of Pizza at the same time, it will turn into a multi-label classification problem, so there will be many duplicate errors on top of each other, so the total number of actual errors if calculated will be greater than the number of pizza images I collect, as shown in 2. Therefore, based on these error properties, we will proceed with the one-hot encoding method to convert to a new attribute label_detail: An array containing errors occurs on a pizza element, the number of patterns is visualized as shown in 3, details the errors below:

Table 1: Details of labels and their mappings in our dataset

Error Field	Description	Mapping
err_adv_edge	Advanced edge defects	1
err_edge	Basic edge defects	2
err_shape	Defects related to the shape of the pizza	3
err_baking	Baking defects, such as overcooked or undercooked	4
err_size	Size defects, such as pizzas being too large or too small	5
err_topping	Topping defects, including uneven distribution	6
err_ferment	Fermentation defects, such as uneven rising	7

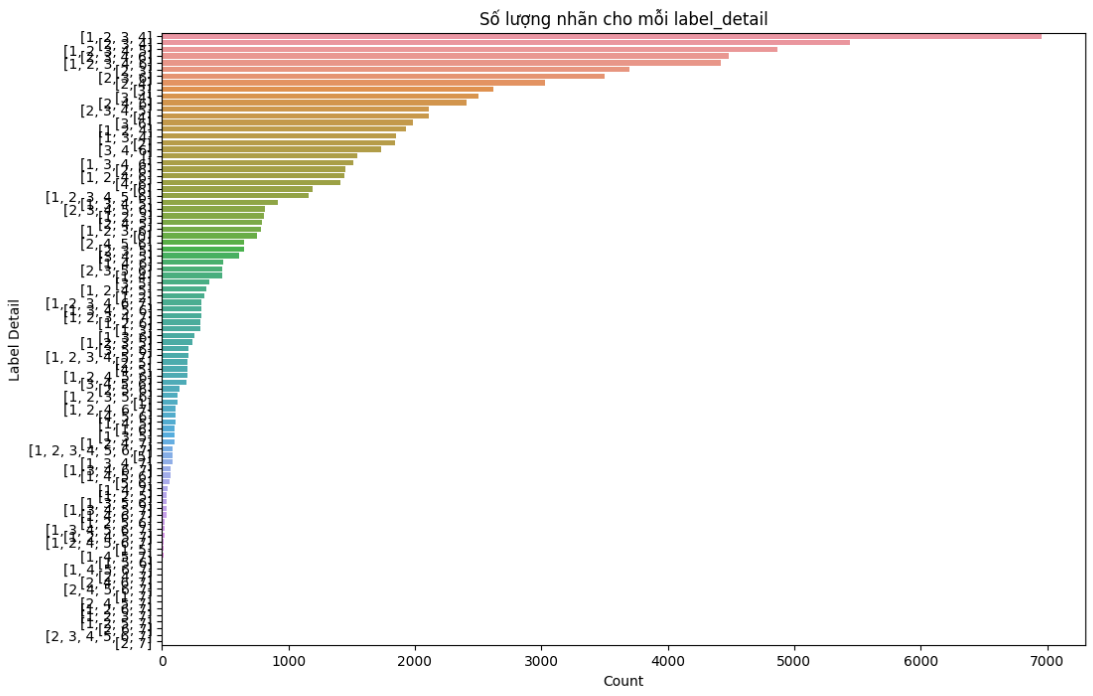


Figure 3: Visually the number of detailed labels.

4 Our Approach

In this section, we delineate the proposed model architecture and approach. In section 4.1, we provide a high-level schematic of the end-to-end model pipeline. Next, we detail the data preprocessing steps taken to prepare the input data for summarization in section 4.2. After the data preprocessing process, we will train the Deep Learning - Efficientnet model and finetune YOLOv8 to apply to our multi-label classification problem to obtain a model that recognizes and classifies the conditions of pizza, details of the Efficientnet model will be detailed by us in Section 4.3 and Section 4.4.

4.1 Overview

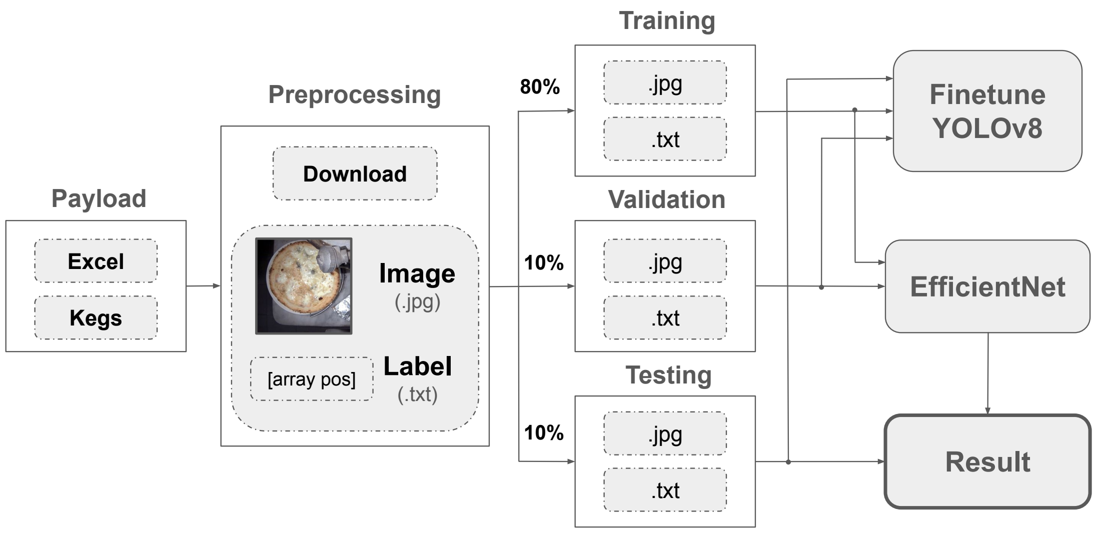


Figure 4: Pipeline for proposed model.

Our approach integrates advanced image processing techniques and deep learning models to construct a robust pizza monitoring system. The initial dataset includes images with and without pizzas. To optimize data processing, we employed the pre-trained YOLOv8x model to identify and extract coordinates of pizzas within images. This pre-labeling step significantly improves the data quality and relevance for subsequent training phases Figure 4

4.2 Data Pre-processing

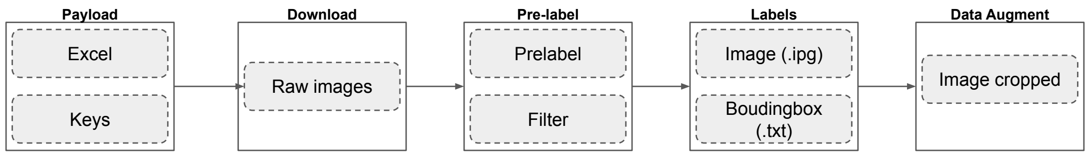


Figure 5: Pipeline for pre-processing phase.

As illustrated in Figure 5, our data pre-processing pipeline involves two key steps: pre-labeling and data augmentation. First, after receiving the payload information, we proceed to download images that have been saved on AWS S3, however because the images are collected using the Search API method, there will be cases of noisy images. To remove noisy images, we use the Pre-label method using YOLOv8. The basis for us to choose the YOLOv8 model is that the YOLO model is trained in 2 large data sets, Imagenet and COCO, and which contains the class is pizza. Based on this, we can pre-label the downloaded images and screen the images containing pizza, after the pre-label process the results will have the difference as shown in Figure 6. From there, we will receive the bounding box information of pizza in the image with the ".txt" file. In the data augment step, we will crop the bounding box images to filter the clearest pizza image.

After completing the data augment process, we proceed to arrange the division of data sets including images and corresponding label files into 3 data sets: training datasets, evaluation datasets and test datasets. These three datasets will be the basis for me to finetune the YOLOv8 model and train the EfficientNet model, with a ratio of 8:1:1.

4.3 YOLOv8

YOLOv8 uses the same backbone as YOLOv5 with some changes on the CSPLayer, now known as the C2f module. The C2f module (partially bottled between stages with two convolutions) combines high-level features with contextual information to improve detection accuracy.

YOLOv8 uses an anchor-free model with a detached head to independently handle objectivity, classification and regression tasks. This design allows each branch to focus on its task and improve the overall accuracy of the model. In the output class of YOLOv8, they used the sigmoid function as the trigger function for the object score, representing the probability that the limit box contains an object. It uses the softmax function for class probability, which represents the probability of objects belonging to each possible class.

YOLOv8 uses the CIoU [9] and DFL [10] loss functions to lose the limit box and binary cross entropy to classify the loss. These losses have improved object detection performance, especially when handling smaller objects.

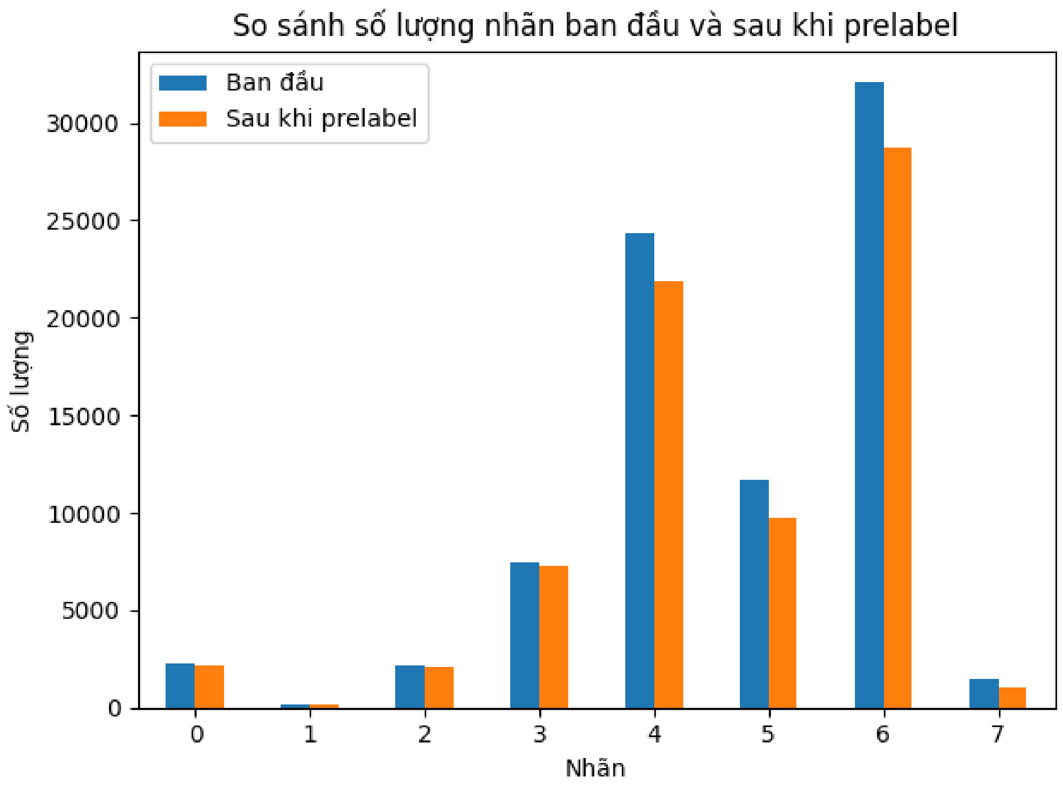


Figure 6: Visually compare labels after being pre-labeled

YOLOv8 also provides a semantic segmentation model called the YOLOv8-Seg model. The backbone is a CSPDarknet53 feature extractor, followed by the C2f module instead of the traditional YOLO ancient architecture. The C2f module is followed by two segment heads, learning to predict semantic segment masks for input images. The model has detection heads similar to YOLOv8, consisting of five detection modules and a prediction layer. The YOLOv8-Seg model has achieved state-of-the-art results on various object detection benchmarks and semantic segments while maintaining high speed and efficiency.

However, our problem has a specific amount of pizza data, and there are more detailed data labels than just pizza identification, we want it to be able to identify what problem this pizza is having. Therefore, we fine tune the YOLOv8 model here we fine tune the YOLOv8x model, the result of the finetune YOLOv8 model we will show in the Section 5.2.

4.4 EfficientNet model

EfficientNet uses Mobile Inverted Bottleneck (MBConv) classes, as shown in Figure 8, which are a combination of convolutions that can be separated in depth and the remaining blocks in reverse. Additionally, the model architecture uses Squeeze-and-Excitation (SE) optimization to further enhance the model's performance.

EfficientNet represents a significant advancement in convolutional neural network (CNN) design, achieving state-of-the-art performance with fewer parameters and computational resources. The architecture of EfficientNet is based on a novel scaling method that uniformly scales the network's depth, width, and resolution using a compound coefficient. This balanced scaling approach allows EfficientNet

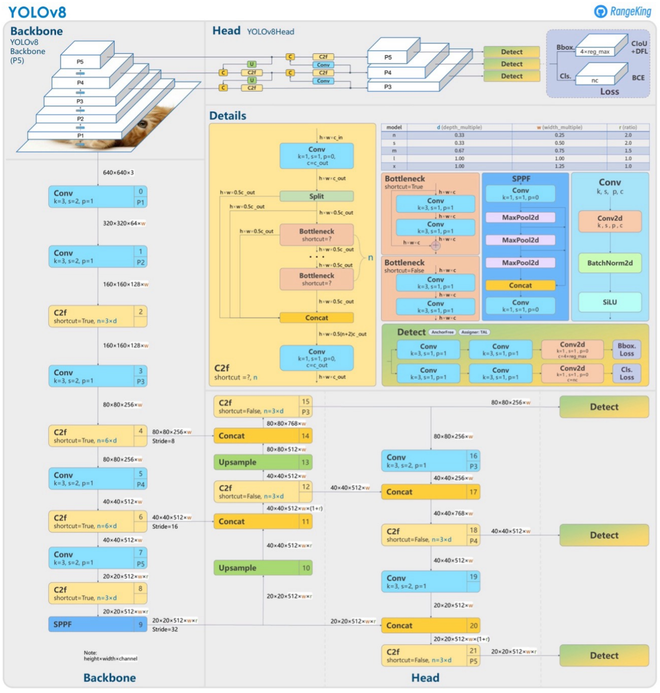


Figure 7: Architecture of YOLOv8 model [1].

to maintain high accuracy while being computationally efficient, making it ideal for real-world applications.

At the core of EfficientNet are Mobile Inverted Bottleneck (MBConv) blocks, which are inspired by MobileNetV2. These blocks consist of depthwise separable convolutions and inverted residuals, which expand the input channels before projecting them back to a lower-dimensional space. This design enhances the network's capacity to learn complex features while keeping the parameter count low. Additionally, EfficientNet incorporates Squeeze-and-Excitation (SE) blocks, which dynamically recalibrate channel-wise feature responses by modeling interdependencies between channels. This mechanism further boosts the network's representational power and accuracy.

EfficientNet has different variations, such as EfficientNet-B0, EfficientNet-B1, etc., with different scale coefficients. Each variant represents a different trade-off between the size and accuracy of the model, allowing users to choose the appropriate model variant based on their specific requirements.

Figure 9 describes the EfficientNet curve marked with a red line. On the horizontal axis is the model size, while the vertical axis represents the correct speed. Just looking at this illustration is enough to emphasize the power of EfficientNet. In terms of accuracy, EfficientNet is only 0.1% ahead of its predecessors, slightly ahead of the previous modern model, GPipe. It is worth noting that the method used to achieve this accuracy. While GPipe relies on 556 million parameters, EfficientNet does the same thing with only 66 million – a huge contrast. In real-life situations, a 0.1% marginal accuracy increase may go unnoticed. However, a remarkable eight-fold rate greatly enhances the usability of the network and its potential for real-world industrial applications.

To adapt EfficientNet for our multi-label classification task, we initialized the model with pre-trained weights on the ImageNet dataset. The original fully connected (FC) layer, designed for 1000-class classification, was replaced with a custom FC layer tailored to our specific problem. The custom FC layer con-

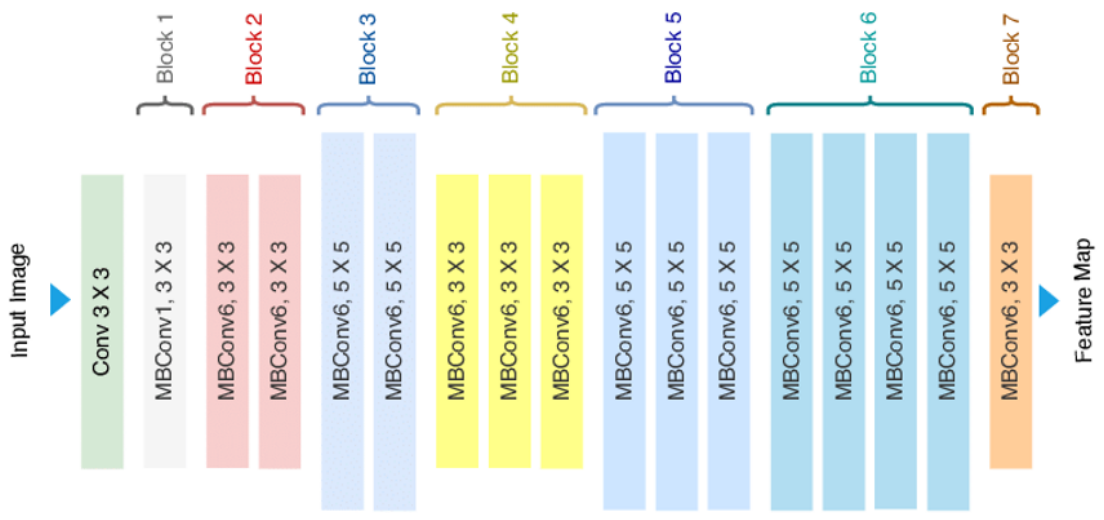


Figure 8: The backbone structure of the Efficienet model.

sists of multiple linear layers with ReLU activations and dropout regularization, culminating in an output layer with a sigmoid activation function to handle the multi-label nature of our dataset (Figure 10).

The final architecture of the adapted EfficientNet model includes:

- Linear layer 1024 dimension
- ReLU activation
- Dropout 0.3
- Linear layer 512 dimension
- ReLU
- Dropout 0.3
- Linear layer 128 dimensional
- ReLU
- Dropout 0.3
- Linear layer 8-dimensional output with sigmoid function.

This architecture enables the model to estimate the probability of each defect type independently, allowing for multiple defects to be identified in a single image.

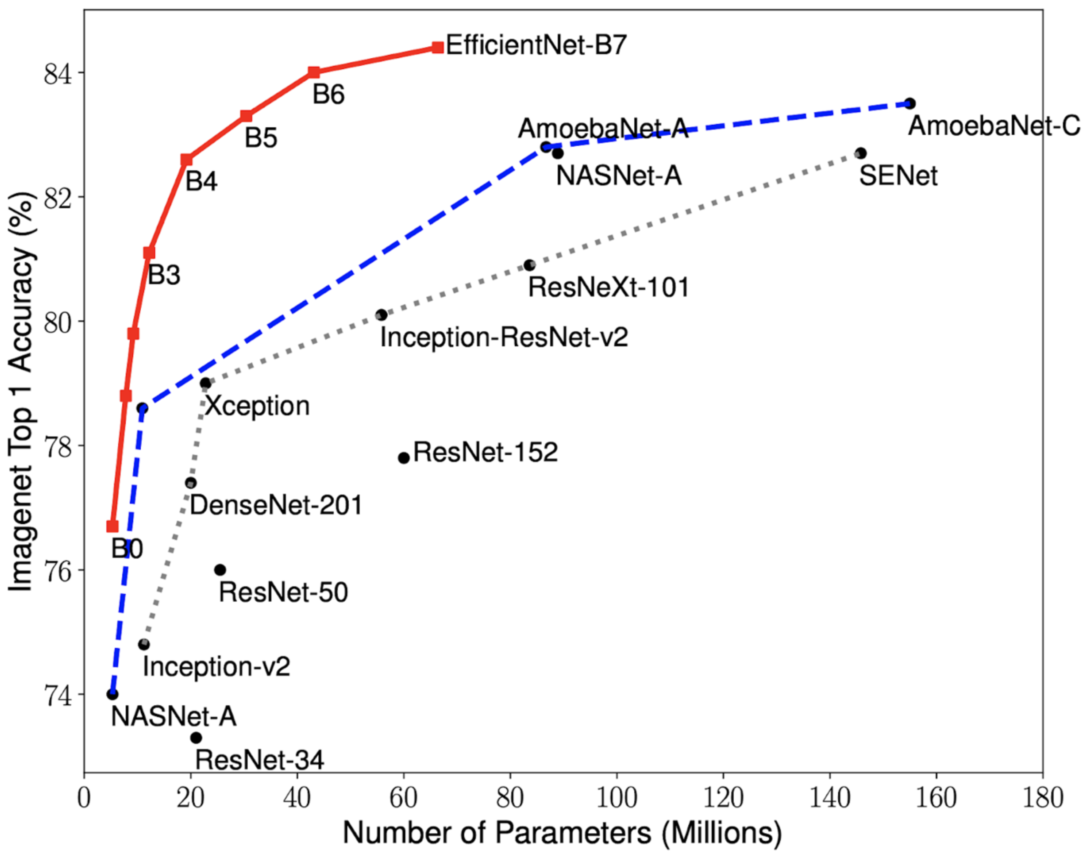


Figure 9: Compare the number of parameters of the EfficientNet model compared to the rest of the models.

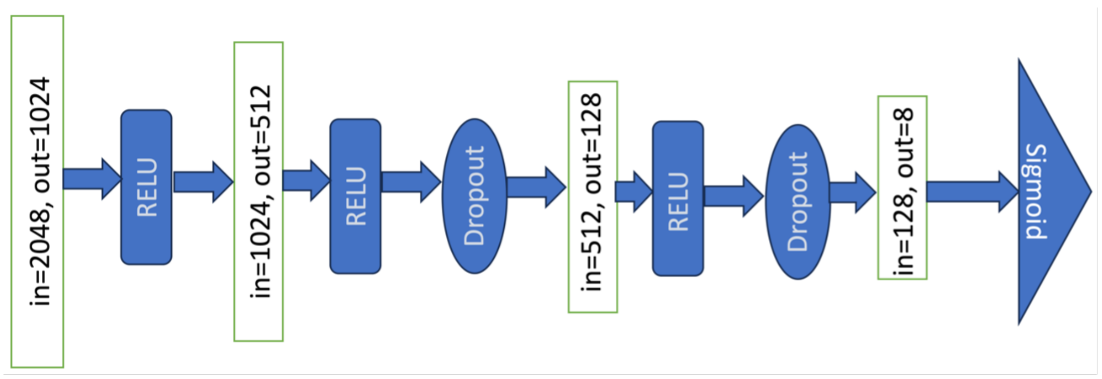


Figure 10: Our Fully Connected Class Architecture.

5 Evaluation

5.1 Metrics and experimental environment

In our topic, the main assessment method we use is the accuracy, hamming loss, recall, precision of the algorithm based on the predicted classes compared to the original class. The hardware structure we use when conducting pizza classification model training is detailed in Table 2

Table 2: Hardware characteristics

Unit	Description
Processor	Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz
RAM	Quadro RTX 8000 - 48GB
Operating System	Ubuntu 20.04.6 LTS

- **Hamming Loss:** It reports how many times on average, the relevance of an example to a class label is incorrectly predicted. Therefore, hamming loss takes into account the prediction error (an incorrect label is predicted) and missing error (a relevant label not predicted), normalized over total number of classes and total number of examples.

$$\text{Hamming Loss} = \frac{1}{nL}\sum_{i=1}^{n}\sum_{j=1}^{L} I\left(y_i^j \neq \hat{y}_i^j\right) \tag{1}$$

- **Accuracy**: Accuracy for each instance is defined as the proportion of the predicted correct labels to the total number (predicted and actual) of labels for that instance. Overall accuracy is the average across all instances. It is less ambiguously referred to as the *Hamming score.*

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i \cap \hat{y}_i|}{|y_i \cup \hat{y}_i|} \tag{2}$$

- **Recall**: It is the propotion of predicted correct labels to the total number of predicted labels, averaged over all instances.

$$\text{Recall} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i \cap \hat{y}_i|}{|y_i|} \tag{3}$$

- **Precision**: It is the propotion of predicted correct labels to the total number of predicted labels, averaged over all instances.

$$\text{Precision} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i \cap \hat{y}_i|}{|\hat{y}_i|} \tag{4}$$

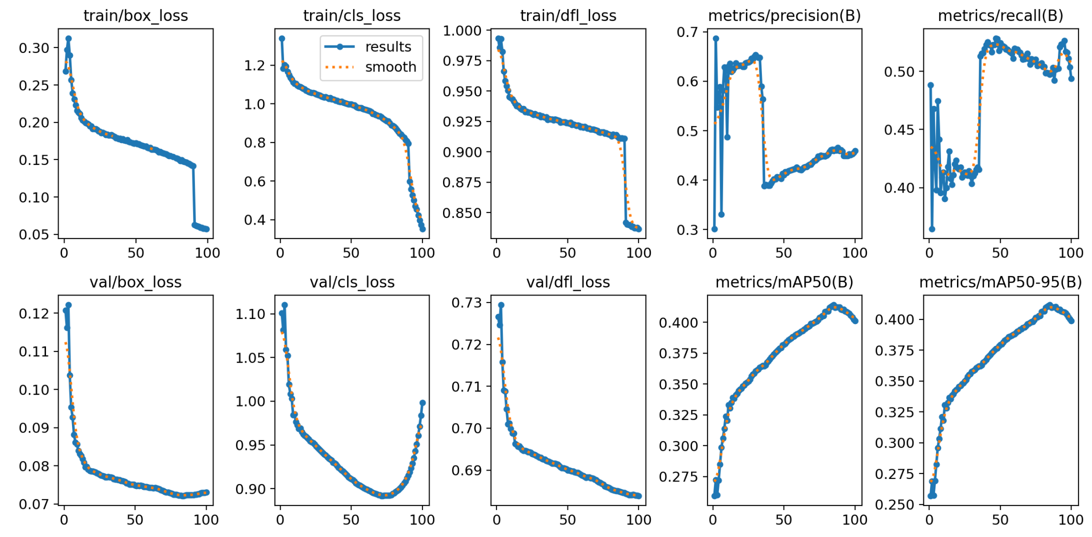


Figure 11: YOLOv8 model finetune process results.

	Accuracy	Hamming loss	Precision	Recall
Our model	0.8	0.2	0.82	0.79

Table 3: Results of the Efficienetb5 model on our test dataset

5.2 Result

YOLOv8: The YOLOv8 model shows a consistent improvement in performance metrics throughout the training process. Training losses for box localization, classification, and distributional focal loss (DFL) all decrease steadily, indicating effective learning and refinement of the model, as detailed in Figure 11. Similarly, validation losses follow a downward trend, suggesting good generalization to unseen data. The metrics for precision and recall show an initial fluctuation but stabilize and improve, reflecting a reduction in false positives and an increase in true positives. Notably, there is a significant drop around epoch 60 in both training and validation losses, potentially due to a change in training strategy, leading to a marked improvement in performance metrics thereafter.

The final results highlight the model's robust performance, with significant improvements in mean Average Precision (mAP) metrics, both at IoU 0.5 and across different IoU thresholds (mAP50-95). This indicates that the YOLOv8 model is proficient in detecting objects with high accuracy and confidence. The model's ability to generalize well across various validation datasets showcases its robustness and reliability for real-world applications. The benefits of this model include enhanced object detection capabilities, improved precision and recall rates, and overall higher accuracy in diverse environments, making it highly suitable for deployment in various practical scenarios requiring efficient and reliable object detection.

Based on the evaluation of the EfficientNetB5 model using the specified metrics, the results demonstrate a commendable performance suitable for practical applications. The model achieved an accuracy of 80%, indicating a high level of correct predictions. The Hamming loss, calculated at 0.20, suggests that the model has a relatively low rate of incorrect classifications across multiple labels. Furthermore, the precision and recall metrics are both at 82% và 79%, reflecting a balanced performance with the model effectively identifying true positives while maintaining a low rate of false positives and false negatives. These results indicate that the EfficientNetB5 model has been trained to a level of performance that is reliable and robust, making it well-suited for deployment in real-world scenarios where high accuracy, precision, and recall are critical.

6 Our Application

To implement our model in practice, we will embed our model on a server deployed based on FastAPI, and interact with users through the Front-end built on HTML [link-demo], the demo link is publicized by using the ngrok method. Here is publication code: https://github.com/Phenikaa-University/cv-finalterm.git.

Upload an Image for Prediction
Choose File no file selected
Upload
Prediction:

Figure 12: User's homepage interface.

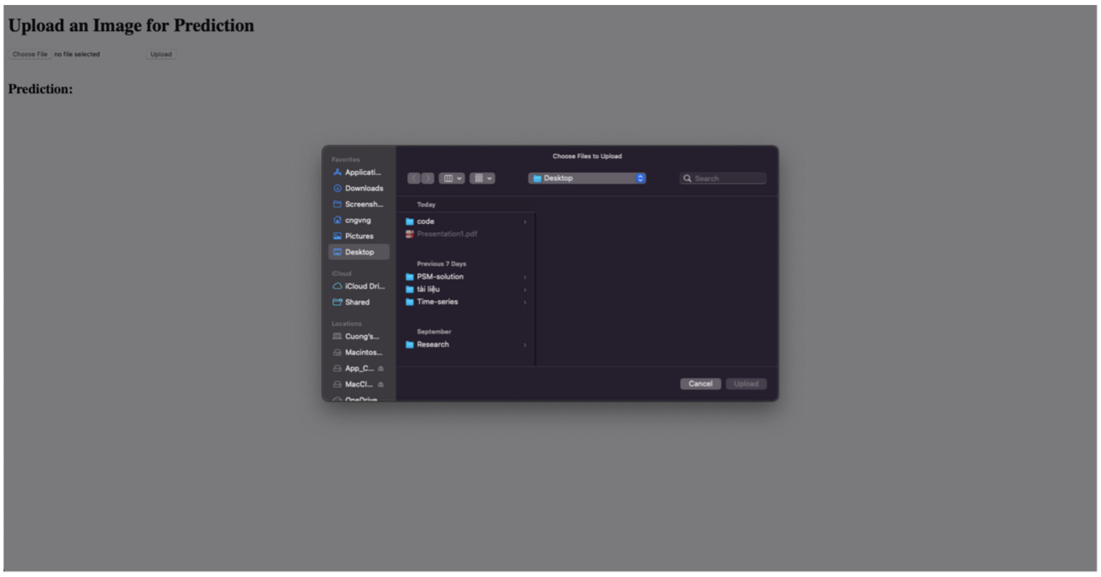
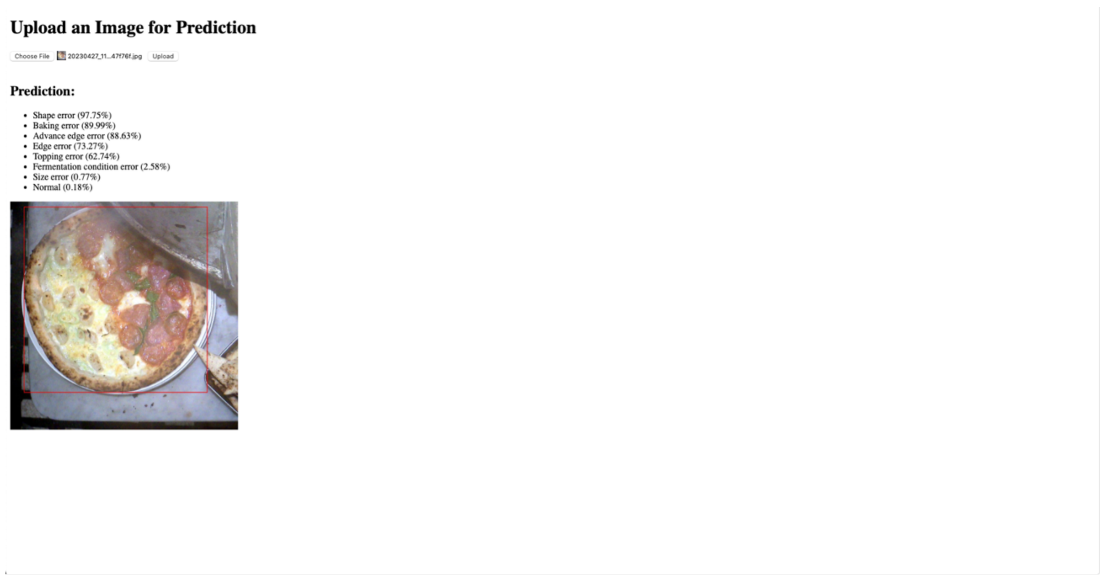


Figure 13: Interface when the user proceeds to upload the user's image.



Upload an Image for Prediction

Prediction:

- Shape error (97.75%)
- Baking error (89.99%)
- Advance edge error (88.63%)
- Edge error (73.27%)
- Topping error (62.74%)
- Fermentation condition error (2.58%)
- Size error (0.77%)
- Normal (0.18%)

Figure 14: User Return Result Interface

7 Conclusion

In this topic, we have explored and built the foundation for the topic of "PizzaRco: AI application to recognize the status of Pizza" based on a large data set of 81,526 pizza images and accompanying label information. We have taken the following important steps:

Identify Data Sets: We have identified and collected datasets containing pizza images along with label information for each image. This dataset contains common types of errors that can appear on the pizza.

Building Artificial Intelligence Models: We have been building AI models to predict and classify types of errors on pizza based on images. This model is an important part of automating the pizza product quality inspection process.

Practical implementation: We have proceeded to build a tool to proceed in a practical process with allowing users to provide pizza images to make predictions and detect errors contained in the provided pizza images.

Next Goal: By building the foundation and customizing the model, we have prepared for the next stages of the topic. The potential applications of this project are to improve the production process and check the quality of pizza products, thereby improving the customer experience and optimizing the business process.

This topic marks the beginning of the topic of classifying and predicting errors in pizza images, and we are very excited about the potential and opportunities it offers. We hope that through the development of models and practical applications, this project will contribute significantly to the food industry and pizza production.

References

[1] G. Jocher *et al.*, "Yolo by ultralytics," *Ultralytics*, 2023.

[2] N. OD4320. Pizza restaurants in the us - market size, industry analysis, trends and forecasts (2024-2029). [Online]. Available: https://www.ibisworld.com/united-states/market-research-reports/pizza-restaurants-industry/#IndustryStatisticsAndTrends

[3] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[4] K. O'shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *International Conference on Machine Learning*, pp. 6105–6114, 2019.

[9] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 12 993–13 000.

[10] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 002–21 012, 2020.

A Division of work:

Below is the division of the work of the team members who contributed to this project.

Vuong Tuan Cuong - 21011490:

- Make an idea of the topic and plan the division of work.
- Crawling data, create tool data.
- Deploy backend and AI server.
- In charge of model building and data preprocessing for datasets.
- Frame the report and write the report section: Abstract, Introduction, Related Work, Our Model, Result, Conclusion.

Nguyen Le Phuong Linh - 22014068:

- Make flow-chart of collecting data, building interface for inference.
- Deploy Front-end phase.
- In charge of assist preprocessing dataset.
- Frame the report and write the report section: Dataset , Application, Conclusion.