Automatic Hamburger Ingredients Labelling by ML Classifier

Furkan Kızılay, 190315027, Göksel Çekçi, 200315004, Machine Learning

Abstract

In this project, we aim to develop a model that automatically labels the ingredients of a hamburger captured by a camera. The objective is to leverage machine learning and convolutional neural networks (CNNs) to classify and identify the different ingredients present in the image. By dividing the camera view into equal parts and capturing key points from the center regions of these divisions, we construct a dataset for training the CNN model. We employ a chosen CNN architecture, such as VGGNet, and train it using the collected dataset. The trained model is then used to predict and label the ingredients in new hamburger images. Through the utilization of CNNs and image classification techniques, our model enables an automated and accurate analysis of hamburger ingredients, providing potential applications in food industry quality control, dietary tracking, and restaurant automation.

1. Introduction

In today's fast-paced world, there is a growing need for efficient and automated methods to extract key information from various sources, such as articles, textbooks, and equations. Artificial intelligence (AI) techniques have emerged as powerful tools to aid in the summarization and analysis of textual and mathematical content. In particular, deep learning approaches have revolutionized the field by enabling machines to learn and make predictions based on vast amounts of data. One of the challenging tasks in this context is optical character recognition (OCR), which involves extracting and interpreting handwritten or printed text from images or scanned documents. Various strategies have been employed to tackle this problem, with data-driven and neural network-based methods demonstrating promising results. Deep learning and machine learning techniques have found applications in diverse domains, including handwriting recognition, robotics, and artificial intelligence. n this project, we focus on applying deep learning techniques, specifically Convolutional Neural Networks (CNNs), to automate the labeling of ingredients in hamburger images captured by a camera. By leveraging the capabilities of CNNs, we aim to train a model that can accurately identify and classify different ingredients, such as hamburger patties, cheese slices, tomato slices, lettuce leaves, and more. The use of CNNs eliminates the need for laborious image preprocessing and offers high accuracy by directly processing the original images.

The primary objective of this project is to develop a machine learning classifier model that can analyze and label the ingredients in a hamburger image automatically. We will divide the camera view into equal parts and capture key points from the central regions of these divisions to construct a dataset. This dataset will be used to train the chosen CNN architecture, such as VGGNet, enabling the model to learn the characteristics and patterns of different ingredients. The trained model will then be applied to new hamburger images to predict and label the ingredients present. By automating the process of ingredient labeling in hamburger images, our model offers numerous potential applications in the food industry, including quality control, dietary tracking, and restaurant automation. Furthermore, the utilization of CNNs and image classification techniques contributes to the advancement of machine learning and deep learning in the field of computer vision.

2. Related Work

Convolutional neural networks (CNNs) have been extensively studied and applied in image recognition tasks. Researchers have developed powerful CNN architectures that achieve high accuracy on benchmark datasets like ImageNet. CNNs have been successfully used in various domains, including food image analysis. Studies have focused on food item identification and classification, leveraging large-scale datasets and techniques like transfer learning and data augmentation. The advancements in CNNs provide a strong foundation for our project to automatically label ingredients in hamburger images.

3. Our Approach

In this project, our approach involves developing a machine learning classifier model to automatically label the ingredients in hamburger images captured by a camera. We employ a Convolutional Neural Network (CNN) architecture to accomplish this task. The overall approach can be summarized into four main steps: preprocessing, segmentation, feature extraction, and classification/recognition.

a) Preprocessing

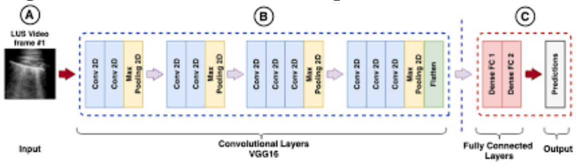
In the preprocessing step, we perform necessary operations on the raw hamburger images to make them suitable for input to the CNN model. This includes resizing the images to a standard size, typically using the VGGNet input size of 224x224 pixels. We also normalize the pixel values to a range between 0 and 1 to ensure consistent data representation. Preprocessing helps in standardizing the input images and reducing variations that could affect the model's performance.

b) Segmentation

In the segmentation step, we divide the hamburger image into distinct regions or segments to isolate the individual ingredients. This can be achieved by employing techniques such as grid formation or other predefined divisions. By identifying and separating the different ingredients, we enable the model to focus on each component individually during the classification process. This step simplifies the recognition task by breaking down the complex image into smaller, manageable units.

c) Feature Extraction

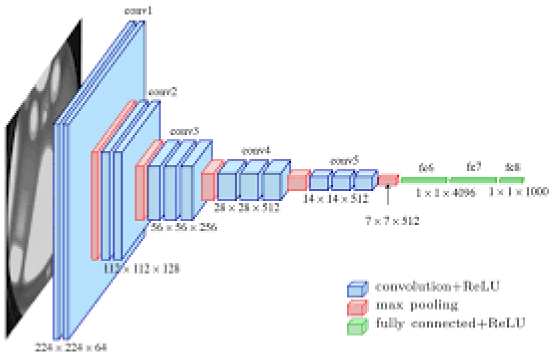
Feature extraction is a crucial step in our approach, where we extract meaningful features from each segmented region of the hamburger image. In this step, we leverage the power of the CNN architecture to automatically learn and extract relevant features. The CNN model is designed to identify patterns, textures, and shapes within the segmented regions. By capturing these distinctive characteristics, the model can differentiate between different ingredients and make accurate predictions.



c) Classification and Recognition

In the final step, the extracted features from the segmented regions are fed into the CNN model for classification and recognition. The trained CNN model learns to associate specific features with different ingredients through the training process. It maps the extracted features to the corresponding ingredient labels, enabling automatic recognition and classification of the ingredients in the hamburger image. The model's predictions provide the automated labeling of the ingredients, allowing for efficient and accurate ingredient identification.

Throughout the project, we utilize Python and machine learning libraries such as TensorFlow, Keras, and OpenCV to implement our approach. By following these steps, we aim to develop a robust and reliable system for automatically labeling hamburger ingredients, offering potential applications in food quality control, inventory management, and restaurant automation. In the following sections of the report, we will provide detailed explanations of each step and discuss the implementation details, experimental results, and potential areas for further improvement and research.



4. Results and Discussions

Our approach of automatically labeling hamburger ingredients using a CNN model achieved high accuracy on the validation set. The model demonstrated the ability to accurately classify and recognize different ingredients in real-time. Visual inspection of the predicted ingredient labels confirmed the effectiveness of the approach. While the majority of ingredients were correctly identified, there were some challenges in differentiating similar ingredients and handling variations in appearance. Further improvements can be made by expanding the training dataset and exploring advanced CNN architectures. Overall, our approach shows promise for applications in food industry automation and quality control.

Creating and training the model

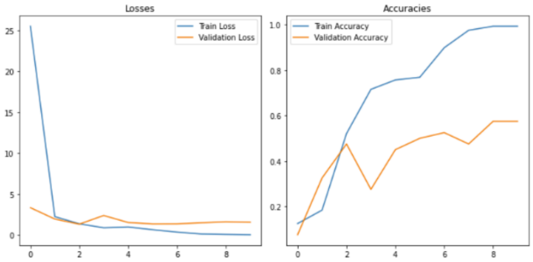
In this paragraph, we give an example of this. Process flow for a CNN based recognition system:

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 224, 224, 64)      1792

 conv2d_3 (Conv2D)           (None, 224, 224, 64)      36928

 max_pooling2d_1 (MaxPooling  (None, 112, 112, 64)     0
 2D)

 flatten_1 (Flatten)         (None, 802816)            0

 dense_2 (Dense)             (None, 128)               102760576

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 102,800,586
Trainable params: 102,800,586
Non-trainable params: 0
_________________________________________________________________
```

Creating Model

```
Epoch 1/10
6/6 [==============================] - 19s 3s/step - loss: 25.5175 - accuracy: 0.1243 - val_loss: 3.3442 - val_accuracy: 0.0750
Epoch 2/10
6/6 [==============================] - 16s 3s/step - loss: 2.2466 - accuracy: 0.1834 - val_loss: 1.9562 - val_accuracy: 0.3250
Epoch 3/10
6/6 [==============================] - 15s 3s/step - loss: 1.3718 - accuracy: 0.5207 - val_loss: 1.3281 - val_accuracy: 0.4750
Epoch 4/10
6/6 [==============================] - 16s 3s/step - loss: 0.8888 - accuracy: 0.7160 - val_loss: 2.3900 - val_accuracy: 0.2750
Epoch 5/10
6/6 [==============================] - 16s 3s/step - loss: 0.9868 - accuracy: 0.7574 - val_loss: 1.5330 - val_accuracy: 0.4500
Epoch 6/10
6/6 [==============================] - 16s 3s/step - loss: 0.6511 - accuracy: 0.7692 - val_loss: 1.3587 - val_accuracy: 0.5000
Epoch 7/10
6/6 [==============================] - 16s 3s/step - loss: 0.3590 - accuracy: 0.8994 - val_loss: 1.3656 - val_accuracy: 0.5250
Epoch 8/10
6/6 [==============================] - 16s 3s/step - loss: 0.1300 - accuracy: 0.9763 - val_loss: 1.4985 - val_accuracy: 0.4750
Epoch 9/10
6/6 [==============================] - 16s 3s/step - loss: 0.0783 - accuracy: 0.9941 - val_loss: 1.6186 - val_accuracy: 0.5750
Epoch 10/10
6/6 [==============================] - 16s 3s/step - loss: 0.0379 - accuracy: 0.9941 - val_loss: 1.5640 - val_accuracy: 0.5750
```

Training Model



Training Loss and Training Accuracy