# Problem Statement

- Ninjacart is India's largest fresh produce supply chain company. They are pioneers in solving one of the toughest supply chain problems of the world by leveraging innovative technology. They source fresh produce from farmers and deliver them to businesses within 12 hours. An integral component of their automation process is the development of robust classifiers which can distinguish between images of different types of vegetables, while also correctly labeling images that do not contain any one type of vegetable as noise.
- As a starting point, ninjacart has provided us with a dataset scraped from the web which contains train and test folders, each having 4 sub-folders with images of onions, potatoes, tomatoes and some market scenes. We have been tasked with preparing a multiclass classifier for identifying these vegetables. The dataset provided has all the required images to achieve the task.

# Ninjacart Business Case: Image Classification for Agricultural Supply Chain Optimization

## Why Ninjacart Needs Image Classification Modeling

Ninjacart, a leading agri-tech platform, connects farmers directly with retailers, restaurants, and other buyers, streamlining the agricultural supply chain in India. The provided dataset, consisting of images of **Onions**, **Potatoes**, **Tomatoes**, and **Indian market** scenes, indicates a focus on automating the identification and classification of agricultural produce and market environments. Image classification modeling is critical for Ninjacart's operations due to the following reasons:

- **Quality Control and Sorting**: Accurate classification of produce (e.g., Onions, Potatoes, Tomatoes) ensures consistent quality assessment. Automated identification can detect defects, categorize produce by type, and ensure only high-quality items reach customers, reducing returns and enhancing customer satisfaction.
- **Inventory Management**: Classifying produce and market-related images helps track inventory in real-time. For example, distinguishing between Onions, Potatoes, and Tomatoes allows Ninjacart to monitor stock levels, predict demand, and optimize procurement from farmers, minimizing waste and shortages.
- **Market Insights**: The "Indian market" class suggests a need to analyze market environments, possibly to identify vendor stalls, storage conditions, or market

trends. This can inform strategic decisions, such as targeting specific markets or optimizing logistics routes.

- **Operational Efficiency**: Manual sorting and identification are labor-intensive and error-prone. Automated image classification reduces human effort, speeds up processes, and lowers operational costs, enabling Ninjacart to scale efficiently.
- **Supply Chain Transparency**: Accurate classification enhances traceability by ensuring that produce is correctly identified from farm to buyer. This builds trust with customers and supports compliance with quality standards.

## How Image Classification Improves Ninjacart's Business

The development and deployment of advanced image classification models (both custom CNNs and pretrained models like ResNet50, InceptionV3, VGG16, and MobileNetV2) directly contribute to Ninjacart's business objectives in the following ways:

- **Enhanced Accuracy and Customer Trust**:
  - **Pretrained Models**: Fine-tuned ResNet50 (94.59%) and InceptionV3 (93.16%) achieve high accuracy, ensuring reliable classification of Onions, Potatoes, Tomatoes, and Indian market scenes. This reduces errors in produce identification, ensuring customers receive the correct items.
  - **Custom Models**: Model 5 (91.74%) offers competitive performance for resource-constrained environments, supporting cost-effective deployment.
  - **Impact**: Accurate classification minimizes customer complaints, enhances brand reputation, and fosters repeat business from retailers and restaurants.
- **Cost Reduction**:
  - **MobileNetV2**: With fewer parameters (3.2M–6.9M), it enables efficient deployment on edge devices (e.g., mobile apps for farmers or warehouse staff), reducing computational costs.
  - **Custom Models**: Models 2–7 (~462K parameters) are lightweight, making them suitable for low-resource settings like rural warehouses.
  - **Impact**: Lower operational costs through automation and efficient models allow Ninjacart to maintain competitive pricing and invest in scaling operations.
- **Improved Supply Chain Efficiency**:
  - Automated classification speeds up sorting and inventory tracking, reducing delays in the supply chain.
  - **Pretrained Models**: High accuracy (e.g., ResNet50 at 94.59%) ensures precise identification, enabling faster processing from farm to market.
  - **Custom Models**: Model 5's balance of accuracy and efficiency supports rapid deployment in warehouses.
  - **Impact**: Faster processing reduces spoilage of perishable goods like Tomatoes, improving profit margins and ensuring fresher produce for customers.
- **Data-Driven Decision Making**:

    - Classifying **Indian market** images can provide insights into market dynamics, such as vendor density or produce quality in specific regions.
    - **Pretrained Models**: Advanced architectures (e.g., InceptionV3) extract robust features, potentially identifying subtle market characteristics.
    - **Impact**: Ninjacart can optimize sourcing strategies, prioritize high-demand regions, and tailor logistics to market conditions, enhancing operational agility.
- **Scalability and Adaptability**:
    - **Pretrained Models**: Leverage pretrained weights from large datasets (e.g., ImageNet), requiring fewer training epochs (5–9 for VGG16) and less labeled data.
    - **Custom Models**: Require more tuning but are adaptable to specific needs with techniques like Data Augmentation (Model 7) or L2 regularization (Model 5).
    - **Impact**: Scalable models support Ninjacart's growth across new regions and product categories, while adaptability ensures robustness to diverse produce types or market conditions.

# Challenges and Recommendations

- **Persistent Misclassifications**:
    - Both custom and pretrained models (e.g., VGG16, custom Model 5) struggle with **Tomato ↔ Indian market** (104–106 cases) and **Indian market ↔ Onion** misclassifications, likely due to visual similarities (e.g., color, texture).
    - **Recommendation**: Use feature visualization (e.g., Grad-CAM) to identify problematic features, collect more diverse **Indian market** data, and apply class-weighted loss to prioritize these classes.
- **Overfitting in Custom Models**:
    - Custom models (e.g., Models 3–6) show significant overfitting (training: 95–99%, validation: 90–94%), unlike pretrained models with minimal overfitting.
    - **Recommendation**: Enhance regularization (stronger L2, higher Dropout) or refine Data Augmentation in Model 7 to balance generalization and accuracy.
- **Model Selection for Ninjacart**:
    - **High-Accuracy Needs**: Deploy **fine-tuned ResNet50 (94.59%)** for warehouses or centralized processing units to maximize accuracy and reliability.
    - **Resource-Constrained Settings**: Use **MobileNetV2 (92.00%)** or custom **Model 5 (91.74%)** for edge devices or rural operations, balancing performance and efficiency.
    - **Hybrid Approach**: Combine ResNet50 for high-stakes tasks (e.g., quality control) and MobileNetV2 for field operations (e.g., farmer-facing apps).
- **Future Improvements**:
    - Explore ensemble models (e.g., ResNet50 + InceptionV3) to boost accuracy and robustness.

- Increase input resolution for custom models (e.g., from 128x128 to 224x224) to match pretrained models, if resources allow.
- Develop real-time classification apps for farmers to upload produce images, streamlining sourcing.

## Conclusion

Image classification modeling is pivotal for Ninjacart to enhance quality control, streamline inventory management, gain market insights, and improve operational efficiency. **Pretrained models**, particularly **fine-tuned ResNet50 (94.59%)**, offer superior accuracy and generalization, making them ideal for high-precision tasks. **Custom Model 5 (91.74%)** and **MobileNetV2 (92.00%)** provide cost-effective alternatives for resource-limited environments. By addressing misclassifications and overfitting, Ninjacart can leverage these models to reduce costs, improve customer satisfaction, and scale its agri-tech platform, ultimately strengthening its position in India's agricultural supply chain.

# Dataset Link

https://drive.google.com/file/d/1clZX-lV_MLxKHSyeyTheX5OCQtNCUcqT/view?usp=sharing

# Context

- This dataset contains images of the following food items: noise-Indian market and images of vegetables- onion, potato and tomato.

# Data Collection

- The images in this dataset were scraped from Google.

# Content

- This dataset contains a folder train, which has a total of 3135 images, split into four folders as follows:
  - Tomato : 789
  - Potato : 898
  - Onion : 849
  - Indian market : 599

- This dataset contains another folder test which has a total of 351 images, split into four folders
    - Tomato : 106
    - potato : 83
    - onion : 81
    - Indian market : 81

# Inspiration

The objective is to develop a program that can recognize the vegetable item(s) in a photo and identify them for the user.

# Process:

- There are numerous libraries that can be used to work with images. Importing the libraries we feel most confident with is our first step: TensorFlow, keras, matplotlib, opencv, seaborn etc
- Download the Data (you can use gdown) with the provided Dataset Link and unzip it
- Visualize the data, use the dataset directory to create a list containing all the image paths in the training folder. You can use matplotlib or tensorflow to plot a grid sample of the images you fetched from the list of image paths.
- Plot a few of the images of each class to check their dimensions. [Note that the images are not all of uniform dimensions]
- Verify the count of images in each train and test folder by plotting histogram .
- Check each folder to see if the number of images matches the reported number.
- Split the dataset to a train and validation set.
- The provided data does not contain separate training and validation folders. For us to do hyperparameter tuning of our models, it is important to divide the dataset into an 80-20 split for training and validation respectively.
- Before fitting data to our model, we must make sure that each image is square-shaped so that we may resize it to the required dimensions and also perform rescaling which will rescale the inputs between 0-1 by dividing each value by 255.
- Use a model of your choice (could be vgg, resnet and mobilenet) and train it with an appropriate batch size.

- Using the pretrained weights of popular networks is a great way to do transfer learning, since the size of our original dataset is small.
- Obtain the testing accuracy to see how well your model generalizes.
- Compare and check the performance of multiple models using a confusion matrix.
- Implement a TensorBoard callback to log each of our model metrics for each model during the training process. [ recommended google colab or jupyter notebook]
- Plot the train/valid accuracy and loss graph for each model.
- If the model is overfitting , Try to tune your model by applying - :
- BatchNormalization and Dropout
- Callbacks : EarlyStopping, ModelCheckpoint and TensorBoard callback
- Data Augmentation

In [ ]:
```
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packa
ges (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-p
ackages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dis
t-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/
dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/
lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/d
ist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist
-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dis
t-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packag
es (from tensorflow) (25.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.2
1.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.11/dist-packages (from
tensorflow) (5.29.5)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/d
ist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packa
ges (from tensorflow) (75.2.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-pack
ages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist
-packages (from tensorflow) (3.1.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python
3.11/dist-packages (from tensorflow) (4.14.1)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-pa
ckages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/d
ist-packages (from tensorflow) (1.73.1)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.
11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-pac
kages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/
dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-pac
kages (from tensorflow) (3.14.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.
11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/loca
l/lib/python3.11/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/di
st-packages (from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (f
rom keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (0.1.0)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages
(from keras>=3.5.0->tensorflow) (0.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-pac
kages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/di
```

```
st-packages (from requests<3,>=2.21.0->tensorflow) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/di
st-packages (from requests<3,>=2.21.0->tensorflow) (2025.7.14)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-
packages (from tensorboard<2.19,>=2.18->tensorflow) (3.8.2)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/loca
l/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-
packages (from tensorboard<2.19,>=2.18->tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dis
t-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.1
1/dist-packages (from rich->keras>=3.5.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.
11/dist-packages (from rich->keras>=3.5.0->tensorflow) (2.19.2)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packa
ges (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)
```

In [ ]:
```python
# Import All Required Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import drive
import random
import glob
import os
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from google.colab import files
from tensorflow.keras import Input
# from tensorflow.keras import metrics
from sklearn import metrics
from tensorflow.keras import layers, regularizers
import functools
from tensorflow.keras.optimizers import Adam,SGD
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import VGG16
from tensorflow.keras.applications import vgg16
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications import resnet50
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.applications import inception_v3
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.applications import MobileNetV2
from IPython.display import FileLink, display
from tensorflow.keras.applications import EfficientNetB0
```

In [ ]:
```
!sudo apt install tree
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  tree
0 upgraded, 1 newly installed, 0 to remove and 35 not upgraded.
Need to get 47.9 kB of archives.
After this operation, 116 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 tree amd64 2.0.2-1 [4
7.9 kB]
Fetched 47.9 kB in 1s (66.5 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based fronten
d cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 78, <> line
1.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package tree.
(Reading database ... 126284 files and directories currently installed.)
Preparing to unpack .../tree_2.0.2-1_amd64.deb ...
Unpacking tree (2.0.2-1) ...
Setting up tree (2.0.2-1) ...
Processing triggers for man-db (2.10.2-1) ...
```

In [ ]:
```
!rm -rf /content/ninjacart_data
```

In [ ]:
```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [ ]:
```
path = r"/content/drive/MyDrive/Colab Notebooks"
folder_name = "ninjacart_data"
folder_path = os.path.join(path, folder_name)
zip_path = os.path.join(path, "ninjacart_data.zip")

cwd = os.getcwd()
extract_path = os.path.join(cwd, folder_name)
print(extract_path)

if not os.path.isdir(extract_path):
    print("Folder doesn't exist. Unzipping...")
    !unzip -q "{zip_path}" -d "{cwd}"
    print("Unzipped successfully.")
else:
    print("Folder already exists.")
```

```
/content/ninjacart_data
Folder doesn't exist. Unzipping...
Unzipped successfully.
```

In [ ]:
```
!cd /content/ninjacart_data   ; tree -L 2
```

```
.
├── test
│   ├── indian market
│   ├── onion
│   ├── potato
│   └── tomato
└── train
    ├── indian market
    ├── onion
    ├── potato
    └── tomato

10 directories, 0 files
```

# Samples from each of these classes and Data Distribution across samples!

In [ ]:

```python
def get_image_filenames(base_path, target_paths=['train'], extensions=[".jpg", "
    """
    Scans a base directory and collects sorted image filenames by category and t

    Args:
        base_path (str): Root directory (e.g., "/content/ninjacart_data")
        target_paths (list): List of subfolders to search inside (e.g., ['train'
        extensions (list): Valid image extensions

    Returns:
        dict: Structure like {'train': {'onion': [...], 'tomato': [...]}, ...}
    """
    result = {}

    for target in target_paths:
        target_dir = os.path.join(base_path, target)
        if not os.path.isdir(target_dir):
            print(f"Skipping missing directory: {target_dir}")
            continue

        category_dict = {}
        for category in os.listdir(target_dir):
            category_path = os.path.join(target_dir, category)
            if not os.path.isdir(category_path):
                continue

            image_files = sorted([
                f for f in os.listdir(category_path)
                if any(f.lower().endswith(ext) for ext in extensions)
            ])

            category_dict[category] = image_files

        result[target] = category_dict

    return result
```

```
def get_sub_folder_names(image_files_path):
  class_dirs = os.listdir(f'{image_files_path}')
  return class_dirs

# def get_categories(base_path):
#   class_dirs = os.listdir(base_path)
#   return class_dirs

def get_category_paths(base_path, traget_paths=['train']):
  get_category_paths = []
  categories = None
  sub_folder_paths = get_train_test_val_paths(base_path)
  for subfolder_path in sub_folder_paths:
    if subfolder_path.split('/')[-1]  in traget_paths :
      categories = os.listdir(subfolder_path)
      for category in categories:
        category_path = os.path.join(subfolder_path, category)
        get_category_paths.append(category_path)
  return get_category_paths


def get_train_test_val_paths(base_path):
  sub_folder_paths = []
  for i in get_sub_folder_names(base_path):
    sub_folder_path = os.path.join(base_path, i)
    sub_folder_paths.append(sub_folder_path)
  return sub_folder_paths

def get_files_count(base_path, traget_paths=['train']):
  final_dict = {}
  for i in traget_paths:
    catergory_files_count_dict={}
    for category_path in get_category_paths(base_path, i):
      files = glob.glob(f'{category_path}/*')
      catergory_files_count_dict.update({category_path.split('/')[-1] : len(file
    final_dict.update({i:catergory_files_count_dict})

  return final_dict

def get_sample_images(base_path, traget_paths=['train']):
  image_dict = {}
  for category_path in get_category_paths(base_path, traget_paths):
    image_files = glob.glob(f'{category_path}/*')
    image_path = random.choice(image_files)
    category = image_path.split('/')[2]
    image_dict.update({category: tf.keras.utils.load_img(image_path)})
  return image_dict

def get_noise_images(noise_path, noise_images):
  image_dict = {}
  for noise_image in noise_images:
    image_path = os.path.join(noise_path, noise_image)
    image_name = image_path.split('/')[-1]
    image_dict.update({image_name: tf.keras.utils.load_img(image_path)})
  return image_dict



def plot_images(image_dict, title):
  plt.figure(figsize=(15, 12))
```

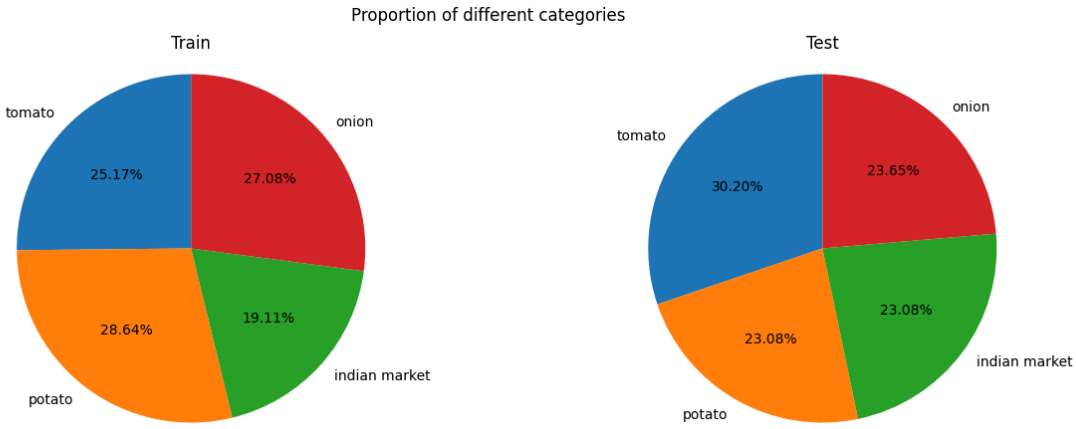
```
  plt.suptitle(title)
  for i, (cls,img) in enumerate(image_dict.items()):
    ax = plt.subplot(3, 4, i + 1)
    plt.imshow(img)
    plt.title(f'{cls}, {img.size}')
    plt.axis("off")
```

In [ ]:
```
base_data_path = 'ninjacart_data'
train_test_proportion = get_files_count(base_data_path, traget_paths=['train', '
pd.DataFrame(train_test_proportion)
```

Out[ ]:

| | train | test |
|---|---|---|
| **tomato** | 789 | 106 |
| **potato** | 898 | 81 |
| **indian market** | 599 | 81 |
| **onion** | 849 | 83 |

In [ ]:
```
train_proportion = train_test_proportion['train']
test_proportion = train_test_proportion['test']
plt.figure(figsize=(15, 5))
plt.suptitle('Proportion of different categories')
plt.subplot(1, 2, 1)
plt.title('Train')
plt.pie(x=train_proportion.values(), labels=train_proportion.keys(), autopct='%0
plt.axis('equal')
plt.subplot(1, 2, 2)
plt.title('Test')
plt.pie(x=test_proportion.values(), labels=test_proportion.keys(), autopct='%0.0
plt.axis('equal')
plt.show()
```




# Observation:

Train Data:

- Potato has the highest number of images at 898 (around 29% of the total). Onion is the second most frequent class with 849 images (around 27% of the total).
- Tomato is the third most frequent class with 789 images (around 25% of the total).

- Indian market has the lowest number of images at 599 (around 19% of the total).
- The training dataset is relatively balanced, with some slight variations in class distribution.

Test Data:

- Tomato has the highest number of images at 106 (around 30% of the total).
- Onion, Indian market, and Potato have a similar number of images (83, 81, and 81 respectively), each representing around 23-24% of the total.

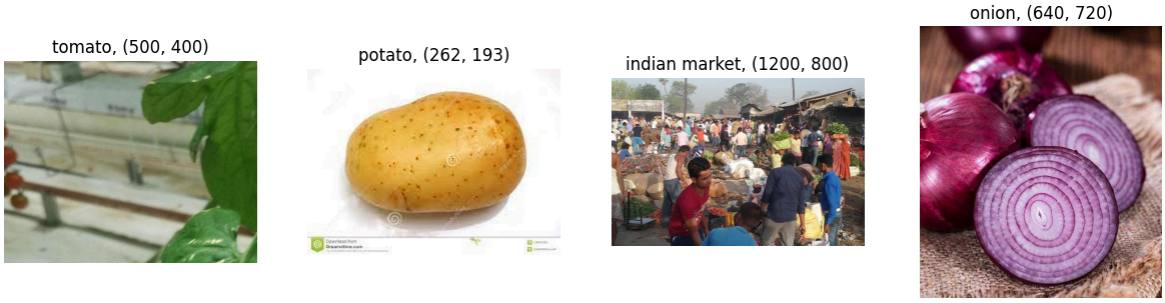
The test dataset is also relatively balanced, with 'tomato' having a slightly higher representation compared to the other classes.

In [ ]:
```
images = get_sample_images(base_data_path)
plot_images(images, title='One sample form each category from training data')
```

One sample form each category from training data


In [ ]:
```
images = get_sample_images(base_data_path, traget_paths=['test'])
plot_images(images, title='One sample form each category from Testing data')
```

One sample form each category from Testing data




# Noise Data Handling

In [ ]:
```
noise_images = ['market11407.png', 'market11408.png', 'market11403.png', 'market
train_indian_market_path = os.path.join(base_data_path, 'train', 'indian market'

noise_img_dict = get_noise_images(train_indian_market_path,noise_images)
plot_images(noise_img_dict, "Noise images from the 'indian market' folder from t
```

Noise images from the 'indian market' folder from the train dataset


In [ ]:
```
def remove_noise_images(base_data_path, noise_images_list):
  """Removes specified noise images from the 'indian market' folder in the train
  train_indian_market_path = os.path.join(base_data_path, 'train', 'indian marke
  removed_count = 0
  for image_name in noise_images_list:
    image_path = os.path.join(train_indian_market_path, image_name)
    if os.path.exists(image_path):
      os.remove(image_path)
      print(f"Removed: {image_path}")
      removed_count += 1
    else:
      print(f"Image not found: {image_path}")
  print(f"Finished removing {removed_count} noise images.")

# Define the base data path and the list of noise images
base_data_path = 'ninjacart_data'
noise_images = ['market11407.png', 'market11408.png', 'market11403.png', 'market

# Call the function to remove the noise images
remove_noise_images(base_data_path, noise_images)
```

```
Removed: ninjacart_data/train/indian market/market11407.png
Removed: ninjacart_data/train/indian market/market11408.png
Removed: ninjacart_data/train/indian market/market11403.png
Removed: ninjacart_data/train/indian market/market11404.png
Removed: ninjacart_data/train/indian market/market11405.png
Removed: ninjacart_data/train/indian market/market11406.png
Finished removing 6 noise images.
```

In [ ]:
```
remove_noise_images(base_data_path, noise_images)
```

```
Image not found: ninjacart_data/train/indian market/market11407.png
Image not found: ninjacart_data/train/indian market/market11408.png
Image not found: ninjacart_data/train/indian market/market11403.png
Image not found: ninjacart_data/train/indian market/market11404.png
Image not found: ninjacart_data/train/indian market/market11405.png
Image not found: ninjacart_data/train/indian market/market11406.png
Finished removing 0 noise images.
```

# Splitting the Train data into Train(80%) and Validation(20%) data sets

In [ ]:

```
import os
import glob
import shutil
from sklearn.model_selection import train_test_split

def create_directory_if_not_exists(directory_path):
    """Creates a directory if it doesn't exist."""
    if not os.path.exists(directory_path):
        os.makedirs(directory_path)
        print(f"Created directory: {directory_path}")
    else:
        print(f"Directory already exists: {directory_path}")

def split_data_for_category(category_path, validation_category_path, test_size=0
    """Splits image files for a given category into train and validation sets.""
    image_files = glob.glob(os.path.join(category_path, '*'))
    train_files, val_files = train_test_split(image_files, test_size=test_size,
    return train_files, val_files

def move_files(file_list, destination_path):
    """Moves a list of files to a specified destination."""
    for file_path in file_list:
        file_name = os.path.basename(file_path)
        destination = os.path.join(destination_path, file_name)
        shutil.move(file_path, destination)

# Define the base path for the data
base_data_path = 'ninjacart_data'
train_path = os.path.join(base_data_path, 'train')
validation_path = os.path.join(base_data_path, 'validation')

# Create the validation directory if it doesn't exist
create_directory_if_not_exists(validation_path)

# Get the list of categories (subfolders) in the training data
categories = os.listdir(train_path)

# Iterate through each category and split and move the data
for category in categories:
    category_train_path = os.path.join(train_path, category)
    category_validation_path = os.path.join(validation_path, category)

    # Create the category subfolder in the validation directory
    create_directory_if_not_exists(category_validation_path)

    # Split the image files for the current category
    train_files, val_files = split_data_for_category(category_train_path, catego

    # Move the validation files to the new validation directory
    print(f"Moving {len(val_files)} images from {category} to validation folder.
    move_files(val_files, category_validation_path)

print("Data splitting and validation folder creation complete.")
```

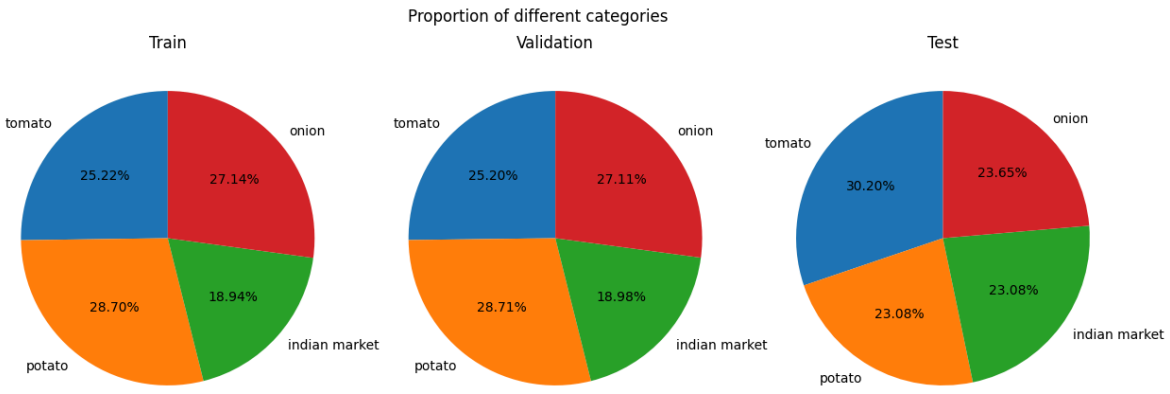
```
Created directory: ninjacart_data/validation
Created directory: ninjacart_data/validation/tomato
Moving 158 images from tomato to validation folder...
Created directory: ninjacart_data/validation/potato
Moving 180 images from potato to validation folder...
Created directory: ninjacart_data/validation/indian market
Moving 119 images from indian market to validation folder...
Created directory: ninjacart_data/validation/onion
Moving 170 images from onion to validation folder...
Data splitting and validation folder creation complete.
```

In [ ]:
```
train_test_val_proportions = get_files_count(base_data_path, traget_paths=['trai
pd.DataFrame(train_test_val_proportions)
```

Out[ ]:

| | train | validation | test |
|---|---|---|---|
| **tomato** | 631 | 158 | 106 |
| **potato** | 718 | 180 | 81 |
| **indian market** | 474 | 119 | 81 |
| **onion** | 679 | 170 | 83 |

In [ ]:
```
train_proportion = train_test_val_proportions['train']
test_proportion = train_test_val_proportions['test']
validation_proportion = train_test_val_proportions['validation']
plt.figure(figsize=(15, 5))
plt.suptitle('Proportion of different categories')
plt.subplot(1, 3, 1)
plt.title('Train')
plt.pie(x=train_proportion.values(), labels=train_proportion.keys(), autopct='%0
plt.axis('equal')
plt.subplot(1, 3, 2)
plt.title('Validation')
plt.pie(x=validation_proportion.values(), labels=validation_proportion.keys(), a
plt.axis('equal')
plt.subplot(1, 3, 3)
plt.title('Test')
plt.pie(x=test_proportion.values(), labels=test_proportion.keys(), autopct='%0.0
plt.axis('equal')
plt.show()
```




Train Data (after splitting for validation):

- Potato now has 718 images, making it the most frequent class.

- Onion has 679 images, the second most frequent.
- Tomato has 631 images, the third most frequent.
- Indian market has 479 images, the least frequent. The training dataset is still relatively balanced, with 'potato' and 'onion' having slightly more images than 'tomato' and 'indian market'.

Validation Data:

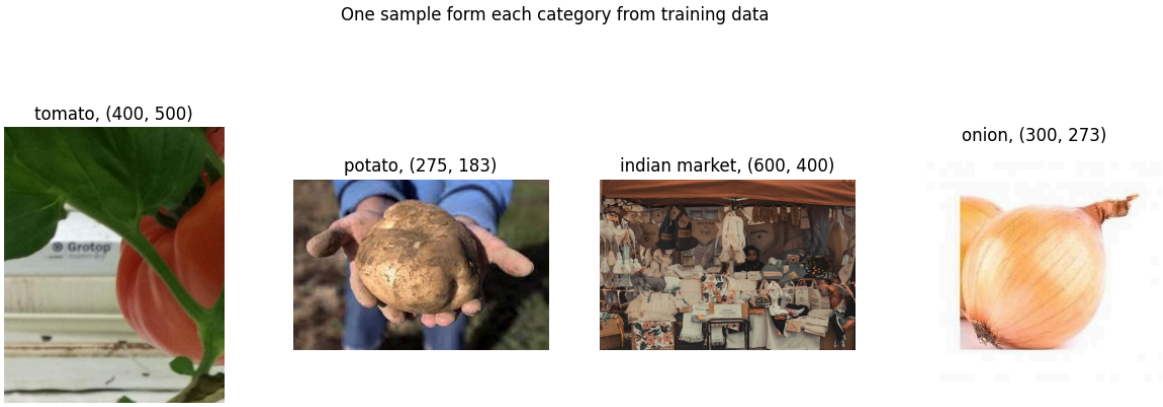
- Potato has the highest number of images at 180.
- Onion has 170 images.
- Tomato has 158 images.
- Indian market has the lowest number of images at 120. The validation dataset maintains a similar distribution to the new training data, which is expected after the split.

Test Data:

- Tomato has the highest number of images at 106.
- Onion, Indian market, and Potato have a similar number of images (83, 81, and 81 respectively). The test dataset is also relatively balanced, with 'tomato' having a slightly higher representation compared to the other classes.

Overall, the data distribution across the classes is fairly consistent among the updated train, validation, and test sets.

In [ ]:
```
images = get_sample_images(base_data_path, ['train'])
plot_images(images, title='One sample form each category from training data')
```




In [ ]:
```
images = get_sample_images(base_data_path, ['validation'])
plot_images(images, title='One sample form each category from validation data')
```

One sample form each category from validation data


In [ ]:
```
images = get_sample_images(base_data_path, ['test'])
plot_images(images, title='One sample form each category from test data')
```

One sample form each category from test data


In [ ]:
```
image_names_dict = get_image_filenames('ninjacart_data', target_paths=['train',
```

# Preprocessing

- Shape Preprocessing
- Value Preprocessing

## Shape Preprocessing

In [ ]:
```
def keras_load_data(base_path, target_paths=['train'], _label_mode='int', _batch
  keras_data_dict = {}
  for target_path in target_paths:
    _shuffle = True if target_path == 'train' else False

    print(f'Loading {target_path} data...')
    path = os.path.join(base_path, target_path)
    data = tf.keras.utils.image_dataset_from_directory(path, shuffle=_shuffle, i
    keras_data_dict.update({target_path: data})
  return keras_data_dict

keras_loaded_data = keras_load_data(base_data_path, target_paths=['train', 'vali
```

```
Loading train data...
Found 2502 files belonging to 4 classes.
Loading validation data...
Found 627 files belonging to 4 classes.
Loading test data...
Found 351 files belonging to 4 classes.
```

In [ ]:
```
keras_loaded_data
```

Out[ ]:
```
{'train': <_PrefetchDataset element_spec=(TensorSpec(shape=(128, 128, 3), dtype
=tf.float32, name=None), TensorSpec(shape=(), dtype=tf.int32, name=None))>,
 'validation': <_PrefetchDataset element_spec=(TensorSpec(shape=(128, 128, 3),
dtype=tf.float32, name=None), TensorSpec(shape=(), dtype=tf.int32, name=None))
>,
 'test': <_PrefetchDataset element_spec=(TensorSpec(shape=(128, 128, 3), dtype=
tf.float32, name=None), TensorSpec(shape=(), dtype=tf.int32, name=None))>}
```

## Value Preprocessing

In [ ]:
```
data_rescale = keras.Sequential(
    name = "data_rescale",
    layers = [
        layers.Rescaling(1./255)
    ]
)

# Perform Data Processing on the train, val, test dataset
train_ds = keras_loaded_data['train'].map(lambda x, y: (data_rescale(x), y))
validation_ds = keras_loaded_data['validation'].map(lambda x, y: (data_rescale(x
test_ds = keras_loaded_data['test'].map(lambda x, y: (data_rescale(x), y))
```

# Snaity check on Preprocessed data

In [ ]:
```
def store_processed_images(dest_dir_name, keras_loaded_data, target_paths=['trai
  """Stores processed images from a keras_loaded_data dictionary into a director
  for target_path in target_paths:
    print(f'Storing {target_path} data...')
    source_path = os.path.join(base_data_path, target_path)
    dest_path = os.path.join(dest_dir_name, target_path)

    # Get class names from the source directory
    class_names = sorted(os.listdir(source_path))

    create_directory_if_not_exists(dest_path)

    # Create category folders and counters
    category_counters = {}
    for category in class_names:
        category_path = os.path.join(dest_path, category)
        create_directory_if_not_exists(category_path)
        category_counters[category] = 1  # Initialize counter

    # Process each image individually
    # Iterate over the dataset which contains rescaled images
    for image, label in keras_loaded_data[target_path]:
        label = int(label.numpy())
        category = class_names[label]
        img_path = os.path.join(dest_path, category)

        # Create the file name: e.g., onion1.jpg
        filename = f"{category}{category_counters[category]}.jpg"
        category_counters[category] += 1  # Increment counter
```

```
        # Encode and save
        encoded_image = tf.cast(image, tf.uint8)
        jpeg_encoded = tf.io.encode_jpeg(encoded_image)
        tf.io.write_file(os.path.join(img_path, filename), jpeg_encoded)
```

In [ ]:
```
store_processed_images('ninjacart_data_processed', keras_loaded_data, target_pat
```

```
Storing train data...
Created directory: ninjacart_data_processed/train
Created directory: ninjacart_data_processed/train/indian market
Created directory: ninjacart_data_processed/train/onion
Created directory: ninjacart_data_processed/train/potato
Created directory: ninjacart_data_processed/train/tomato
Storing validation data...
Created directory: ninjacart_data_processed/validation
Created directory: ninjacart_data_processed/validation/indian market
Created directory: ninjacart_data_processed/validation/onion
Created directory: ninjacart_data_processed/validation/potato
Created directory: ninjacart_data_processed/validation/tomato
Storing test data...
Created directory: ninjacart_data_processed/test
Created directory: ninjacart_data_processed/test/indian market
Created directory: ninjacart_data_processed/test/onion
Created directory: ninjacart_data_processed/test/potato
Created directory: ninjacart_data_processed/test/tomato
```

In [ ]:
```
def make_archive_zip(scr_path, dest_path, target_paths=['train']):
  create_directory_if_not_exists(dest_path)
  for target_path in target_paths:
    source_path = os.path.join(scr_path, target_path)
    # destination_path = os.path.join(dest_dir, target_)
    shutil.make_archive(os.path.join(dest_path, target_path), 'zip', source_path
    # files.download(f'{os.path.join(dest_path, target_path)}.zip')
```

In [ ]:
```
make_archive_zip('ninjacart_data_processed', 'ninjacart_data_processed_zip_files
```

```
Created directory: ninjacart_data_processed_zip_files
```

In [ ]:
```
make_archive_zip('ninjacart_data', 'ninjacart_zip_files', target_paths=['train',
```

```
Created directory: ninjacart_zip_files
```

In [ ]:
```
base_data_path = 'ninjacart_data_processed'
images = get_sample_images(base_data_path, ['train'])
plot_images(images, title='One sample form each category from training data afte
```

One sample form each category from training data after preprocessing


In [ ]:
```
images = get_sample_images(base_data_path, ['test'])
plot_images(images, title='One sample form each category from test data after pr
```

One sample form each category from test data after preprocessing


In [ ]:
```
images = get_sample_images(base_data_path, ['validation'])
plot_images(images, title='One sample form each category from validation data af
```

One sample form each category from validation data after preprocessing

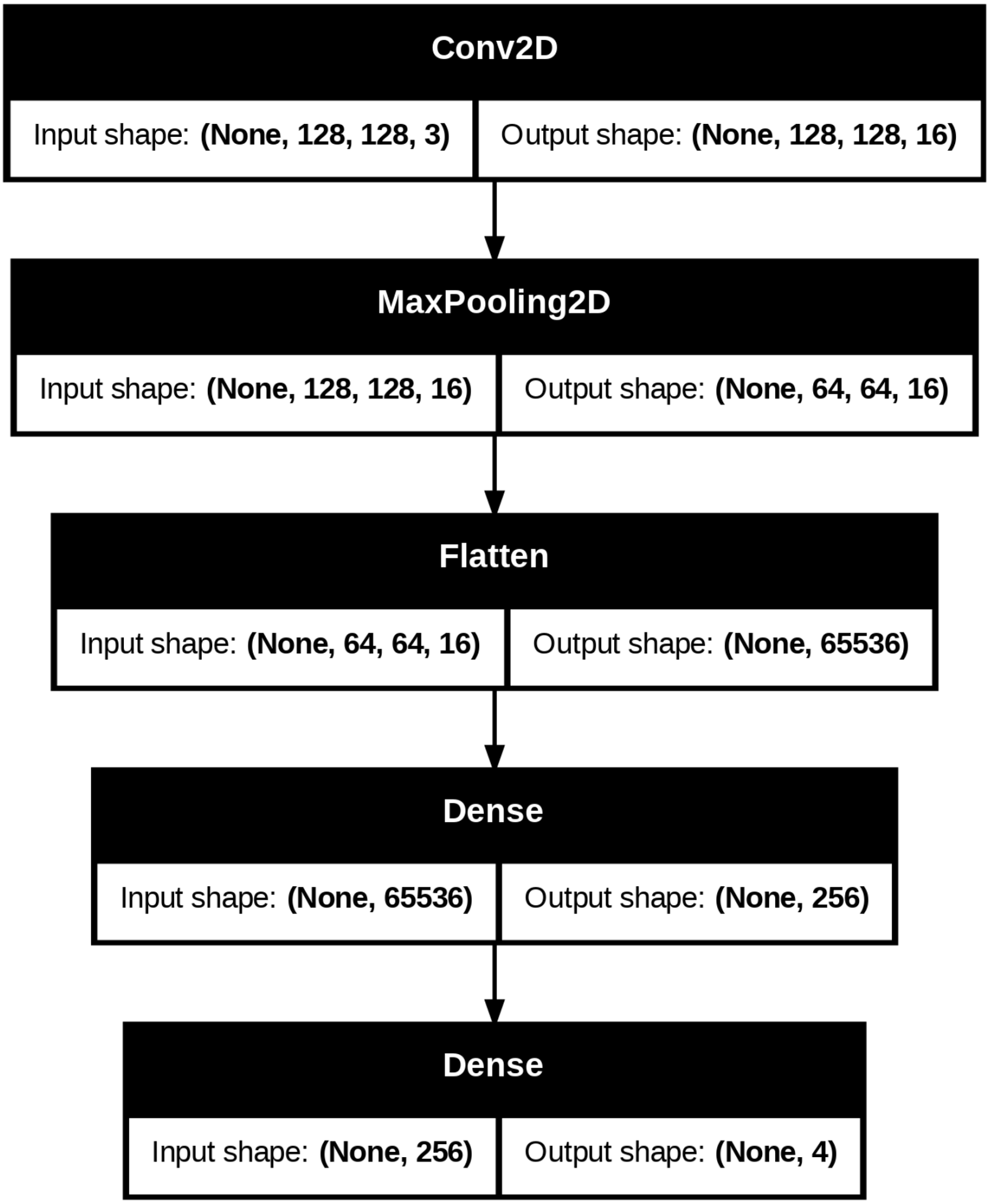

# BaseLine Model

In [ ]:
```
num_classes = 4
hidden_size = 256
height = 128
width = 128

model = keras.Sequential(
    name="model_cnn_without_hyperparmeter_tuning",
    layers=[
        Input(shape=(height, width, 3)),
        layers.Conv2D(filters=16, kernel_size=3, strides=1, padding="same", acti
        layers.MaxPooling2D(),
        layers.Flatten(),
        layers.Dense(units=hidden_size, activation='relu'),
        layers.Dense(units=num_classes, activation='softmax')
    ]
)
```

In [ ]:
```
create_directory_if_not_exists('ninjacart_models')
create_directory_if_not_exists('ninjacart_models/model1_BaseLineModel')
tf.keras.utils.plot_model(model, to_file="/content/ninjacart_models/model1_BaseL
```

```
Created directory: ninjacart_models
Created directory: ninjacart_models/model1_BaseLineModel
```

Out[ ]:




In [ ]:
```
model.summary()
```

**Model: "model_cnn_without_hyperparmeter_tuning"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 128, 128, 16) | 448 |
| max_pooling2d (MaxPooling2D) | (None, 64, 64, 16) | 0 |
| flatten (Flatten) | (None, 65536) | 0 |
| dense (Dense) | (None, 256) | 16,777,472 |
| dense_1 (Dense) | (None, 4) | 1,028 |

```
Total params: 16,778,948 (64.01 MB)
Trainable params: 16,778,948 (64.01 MB)
Non-trainable params: 0 (0.00 B)
```

## b. Compile the model with **cross-entropy** loss and **adam** optimizer:

In [ ]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

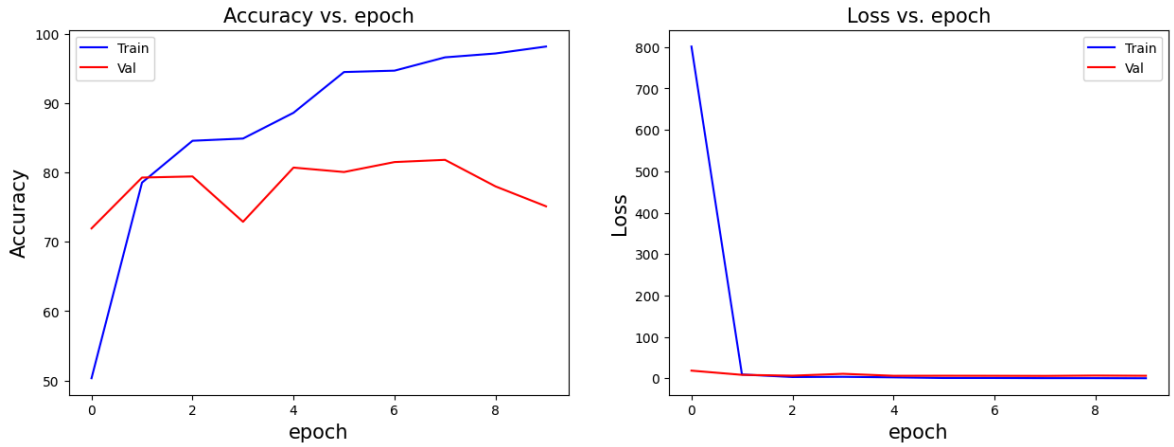
## Training and Evaluation with CNNs

In [ ]:
```
epochs = 10
batch_size = 30
train_ds_batched = keras_loaded_data['train'].batch(batch_size)
validation_ds_batched = keras_loaded_data['validation'].batch(batch_size)
test_ds_batched = keras_loaded_data['test'].batch(batch_size)

model_fit = model.fit(train_ds_batched, validation_data=validation_ds_batched, e
```

```
Epoch 1/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 13s 111ms/step - accuracy: 0.3656 - loss: 2111.0425 -
val_accuracy: 0.7193 - val_loss: 18.4912
Epoch 2/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.7775 - loss: 11.4944 - val_
accuracy: 0.7927 - val_loss: 8.3506
Epoch 3/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 6s 77ms/step - accuracy: 0.8469 - loss: 3.2771 - val_a
ccuracy: 0.7943 - val_loss: 6.3301
Epoch 4/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8398 - loss: 4.1331 - val_a
ccuracy: 0.7289 - val_loss: 10.8417
Epoch 5/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 106ms/step - accuracy: 0.8747 - loss: 2.6937 - val
_accuracy: 0.8070 - val_loss: 6.0605
Epoch 6/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 6s 74ms/step - accuracy: 0.9499 - loss: 0.6330 - val_a
ccuracy: 0.8006 - val_loss: 6.1888
Epoch 7/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 110ms/step - accuracy: 0.9457 - loss: 0.6710 - val_
accuracy: 0.8150 - val_loss: 6.0428
Epoch 8/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9711 - loss: 0.2189 - val_a
ccuracy: 0.8182 - val_loss: 5.8333
Epoch 9/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 77ms/step - accuracy: 0.9857 - loss: 0.1303 - val_
accuracy: 0.7799 - val_loss: 6.6017
Epoch 10/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 109ms/step - accuracy: 0.9783 - loss: 0.1390 - val_
accuracy: 0.7512 - val_loss: 6.1099
```

In [ ]:
```
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(15, 5))
ax = axes.ravel()

#accuracy graph
```

```
ax[0].plot(range(0,model_fit.params['epochs']), [acc * 100 for acc in model_fit.
ax[0].plot(range(0,model_fit.params['epochs']), [acc * 100 for acc in model_fit.
ax[0].set_title('Accuracy vs. epoch', fontsize=15)
ax[0].set_ylabel('Accuracy', fontsize=15)
ax[0].set_xlabel('epoch', fontsize=15)
ax[0].legend()

#loss graph
ax[1].plot(range(0,model_fit.params['epochs']), model_fit.history['loss'], label
ax[1].plot(range(0,model_fit.params['epochs']), model_fit.history['val_loss'], l
ax[1].set_title('Loss vs. epoch', fontsize=15)
ax[1].set_ylabel('Loss', fontsize=15)
ax[1].set_xlabel('epoch', fontsize=15)
ax[1].legend()

#display the graph
plt.show()
```




In [ ]:
```
model.save_weights("/content/ninjacart_models/model1_BaseLineModel/BaseLineModel
```

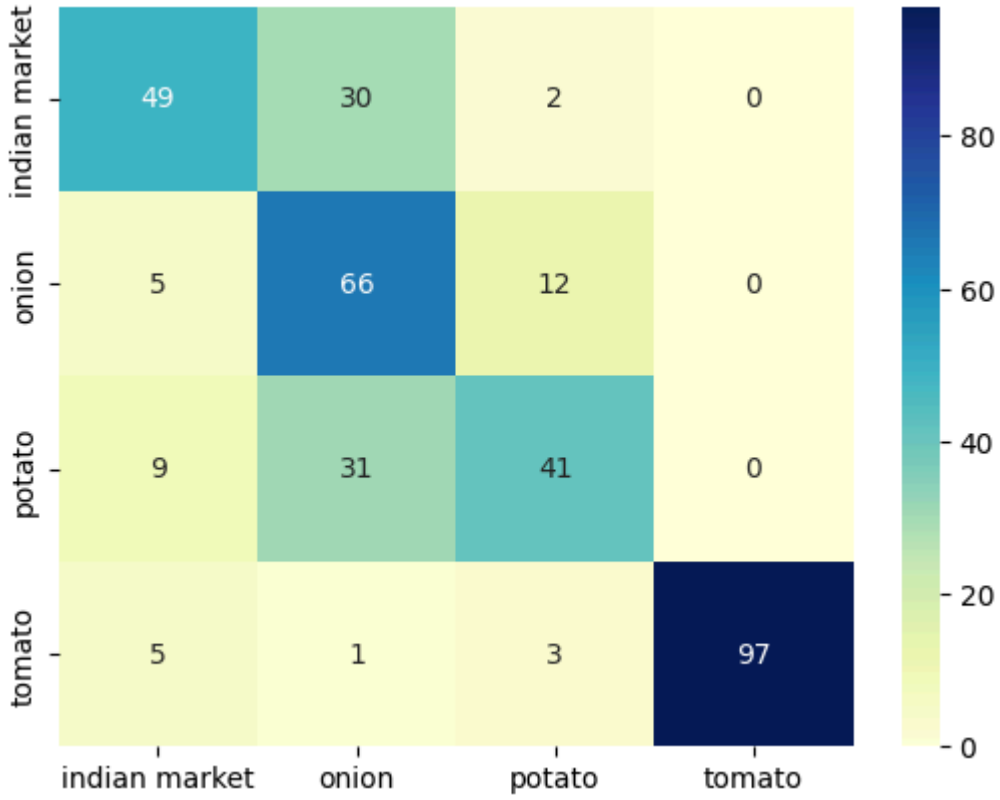
In [ ]:
```
# load model from pretrained checkpoints (optional)
model.load_weights("/content/ninjacart_models/model1_BaseLineModel/BaseLineModel

true_categories = tf.concat([y for x, y in test_ds_batched], axis=0)
images = tf.concat([x for x, y in test_ds_batched], axis=0)
y_pred = model.predict(test_ds_batched)
class_names = keras_loaded_data['test'].class_names
predicted_categories = tf.argmax(y_pred, axis=1)

test_acc = metrics.accuracy_score(true_categories, predicted_categories) * 100
print(f'\nTest Accuracy: {test_acc:.2f}%\n')
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 111ms/step

Test Accuracy: 72.08%
```

In [ ]:
```
def ConfusionMatrix(model, ds, label_list):
# Note: This logic doesn't work with shuffled datasets
    # run model prediction and obtain probabilities
    y_pred = model.predict(ds)
    # get list of predicted classes by taking argmax of the probabilities(y_pred
    predicted_categories = tf.argmax(y_pred, axis=1)
    # create list of all "y"s labels, by iterating over test dataset
    true_categories = tf.concat([y for x, y in ds], axis=0)
    # generate confusion matrix and plot it
```

```
    cm = metrics.confusion_matrix(true_categories,predicted_categories) # last b
    sns.heatmap(cm, annot=True, xticklabels=label_list, yticklabels=label_list,
    plt.show()

ConfusionMatrix(model, test_ds_batched, keras_loaded_data['test'].class_names)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 65ms/step
```



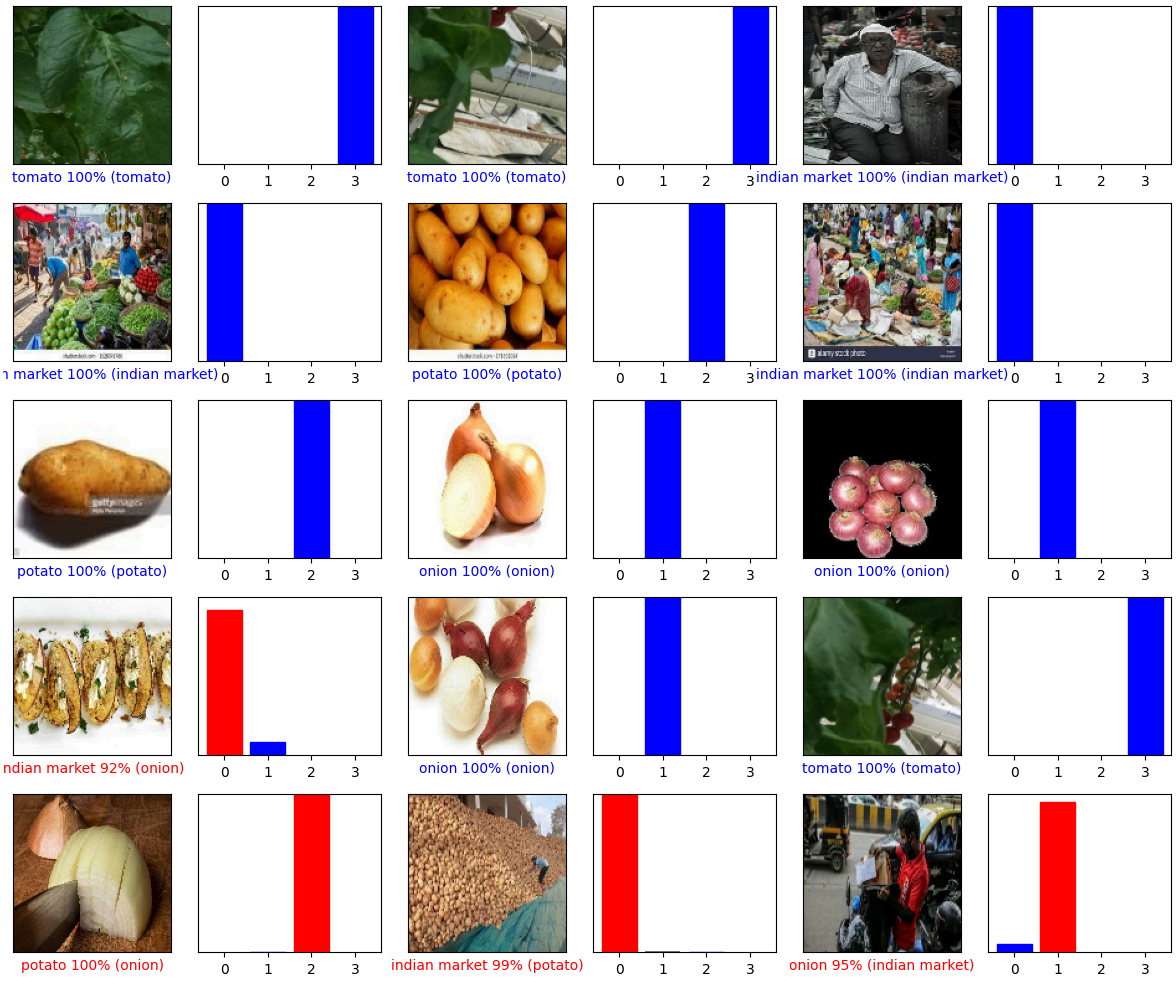

In [ ]:

```
def plot_image(pred_array, true_label, img):
  plt.grid(False)
  plt.xticks([])
  plt.yticks([])

  # Explicitly cast the image data to uint8 before plotting
  img_to_plot = tf.cast(img, tf.uint8)
  plt.imshow(img_to_plot, cmap=plt.cm.binary)

  predicted_label = np.argmax(pred_array)
  if predicted_label == true_label:
    color = 'blue'
  else:
    color = 'red'

  plt.xlabel("{} {:2.0f}% ({})".format(class_names[predicted_label],
                                100*np.max(pred_array),
                                class_names[true_label]),
                                color=color)

# function to plot barplot of class probabilities (pred_array)
def plot_value_array(pred_array, true_label):
  plt.grid(False)
  plt.xticks(range(4))
  plt.yticks([])
  thisplot = plt.bar(range(4), pred_array, color="#777777")
  plt.ylim([0, 1])
  predicted_label = np.argmax(pred_array)
```

```
  thisplot[predicted_label].set_color('red')
  thisplot[true_label].set_color('blue')
```

In [ ]:
```
true_categories = tf.concat([y for x, y in test_ds_batched], axis=0)
images = tf.concat([x for x, y in test_ds_batched], axis=0)
y_pred = model.predict(test_ds_batched)
class_names = keras_loaded_data['test'].class_names

# print(y_pred)
# print(true_categories)
# Randomly sample 15 test images and plot it with their predicted labels, and th
indices = random.sample(range(len(images)), 15)
# Color correct predictions in blue and incorrect predictions in red.
num_rows = 5
num_cols = 3
num_images = num_rows*num_cols
plt.figure(figsize=(2*2*num_cols, 2*num_rows))
for i,index in enumerate(indices):
  plt.subplot(num_rows, 2*num_cols, 2*i+1)
  plot_image(y_pred[index], true_categories[index], images[index])
  plt.subplot(num_rows, 2*num_cols, 2*i+2)
  plot_value_array(y_pred[index], true_categories[index])
plt.tight_layout()
plt.show()
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 115ms/step
```



# Model2 - Using Global Average pooling

In [ ]:
```
def arch_1(height=128, width=128):
    num_classes = 4
    hidden_size = 256

    model = keras.Sequential(
        name="model_cnn_1",
        layers=[
            Input(shape=(height, width, 3)),
            layers.Conv2D(filters=16, kernel_size=3, padding="same", activation=
            layers.MaxPooling2D(),
            layers.Conv2D(filters=32, kernel_size=3, padding="same", activation=
            layers.MaxPooling2D(),
            layers.Conv2D(filters=64, kernel_size=3, padding="same", activation=
            layers.MaxPooling2D(),
            layers.Conv2D(filters=128, kernel_size=3, padding="same", activation
            layers.MaxPooling2D(),
            layers.Conv2D(filters=256, kernel_size=3, padding="same", activation
            # layers.MaxPooling2D(),
            # layers.Flatten(),
            layers.GlobalAveragePooling2D(),
            layers.Dense(units=hidden_size, activation='relu'),
            layers.Dense(units=num_classes, activation='softmax')
        ]
    )
    return model
```

In [ ]:
```
model = arch_1()
model.summary()
```

**Model: "model_cnn_1"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 128, 128, 16) | 448 |
| max_pooling2d_1 (MaxPooling2D) | (None, 64, 64, 16) | 0 |
| conv2d_2 (Conv2D) | (None, 64, 64, 32) | 4,640 |
| max_pooling2d_2 (MaxPooling2D) | (None, 32, 32, 32) | 0 |
| conv2d_3 (Conv2D) | (None, 32, 32, 64) | 18,496 |
| max_pooling2d_3 (MaxPooling2D) | (None, 16, 16, 64) | 0 |
| conv2d_4 (Conv2D) | (None, 16, 16, 128) | 73,856 |
| max_pooling2d_4 (MaxPooling2D) | (None, 8, 8, 128) | 0 |
| conv2d_5 (Conv2D) | (None, 8, 8, 256) | 295,168 |
| global_average_pooling2d (GlobalAveragePooling2D) | (None, 256) | 0 |
| dense_2 (Dense) | (None, 256) | 65,792 |
| dense_3 (Dense) | (None, 4) | 1,028 |

**Total params:** 459,428 (1.75 MB)

**Trainable params:** 459,428 (1.75 MB)

**Non-trainable params:** 0 (0.00 B)

In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model2_with_GAP')
```

```
Created directory: /content/ninjacart_models/model2_with_GAP
```

In [ ]:
```
# model_fit = compile_train_v1(model, train_ds, val_ds)
base_data_path = '/content/ninjacart_data'
keras_loaded_data = keras_load_data(base_data_path, target_paths=['train', 'vali
```

```
Loading train data...
Found 2502 files belonging to 4 classes.
Loading validation data...
Found 627 files belonging to 4 classes.
Loading test data...
Found 351 files belonging to 4 classes.
```

In [ ]:
```
train_ds = keras_loaded_data['train'].map(lambda x, y: (data_rescale(x), y))
validation_ds = keras_loaded_data['validation'].map(lambda x, y: (data_rescale(x
test_ds = keras_loaded_data['test'].map(lambda x, y: (data_rescale(x), y))
```

In [ ]:
```
def compile_train_v1(model, train_ds, val_ds, _callback, epochs=10):
    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    model_fit = model.fit(train_ds, validation_data=val_ds, epochs=epochs, callb
    return model_fit
```

In [ ]:
```
ckpt_path="/content/ninjacart_models/model2_with_GAP/gap_model_10_epochs.weights
callbacks=[
        keras.callbacks.ModelCheckpoint(ckpt_path, save_weights_only=True, monit
    ]
```

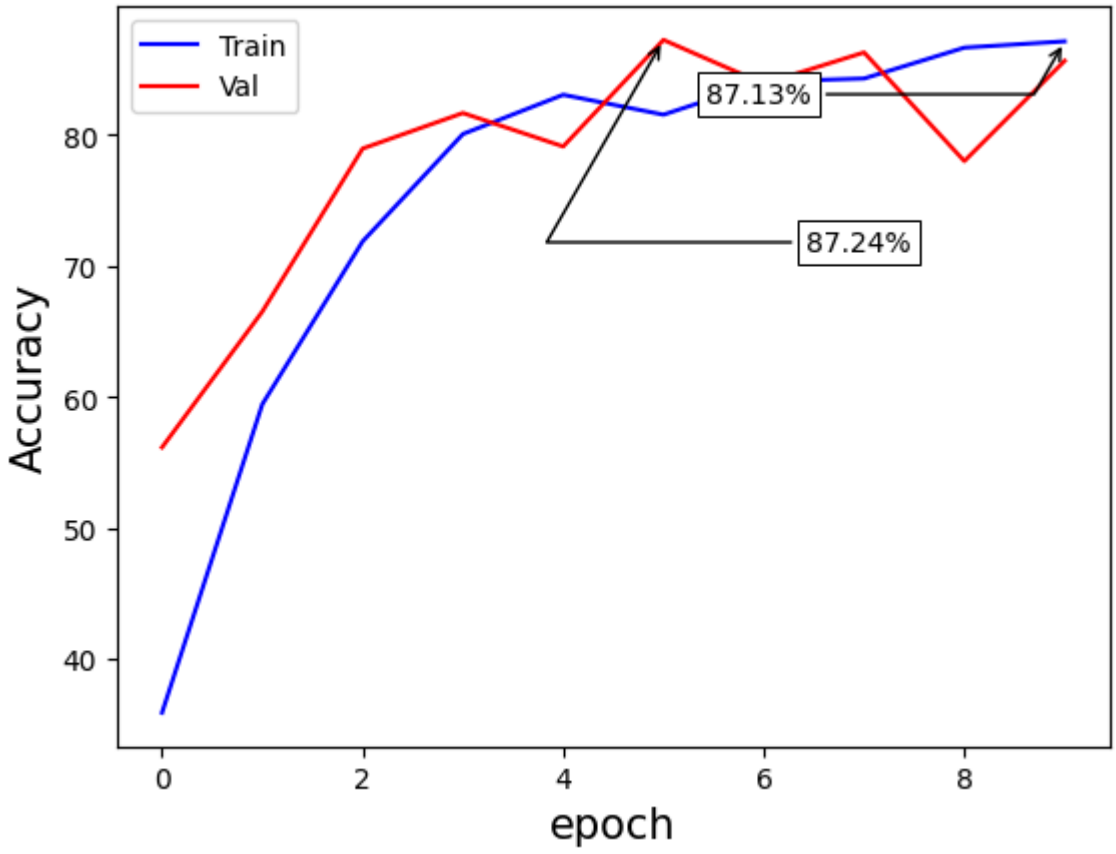
In [ ]:
```
epochs = 10
batch_size = 30
train_ds_batched = keras_loaded_data['train'].batch(batch_size)
validation_ds_batched = keras_loaded_data['validation'].batch(batch_size)
test_ds_batched = keras_loaded_data['test'].batch(batch_size)
model_fit = compile_train_v1(model, train_ds_batched, validation_ds_batched, cal
```

```
Epoch 1/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 16s 131ms/step - accuracy: 0.3064 - loss: 7.1938 - val
_accuracy: 0.5614 - val_loss: 0.9352
Epoch 2/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 78ms/step - accuracy: 0.5533 - loss: 0.9396 - val_a
ccuracy: 0.6651 - val_loss: 0.8803
Epoch 3/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.6997 - loss: 0.7457 - val_a
ccuracy: 0.7895 - val_loss: 0.5413
Epoch 4/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 86ms/step - accuracy: 0.8185 - loss: 0.4770 - val_a
ccuracy: 0.8166 - val_loss: 0.4605
Epoch 5/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 77ms/step - accuracy: 0.8206 - loss: 0.4283 - val_
accuracy: 0.7911 - val_loss: 0.5262
Epoch 6/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.8077 - loss: 0.4919 - val_a
ccuracy: 0.8724 - val_loss: 0.3469
Epoch 7/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 78ms/step - accuracy: 0.8450 - loss: 0.3964 - val_a
ccuracy: 0.8389 - val_loss: 0.4023
Epoch 8/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8337 - loss: 0.4038 - val_a
ccuracy: 0.8628 - val_loss: 0.3343
Epoch 9/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.8675 - loss: 0.3098 - val_a
ccuracy: 0.7799 - val_loss: 0.5083
Epoch 10/10
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 110ms/step - accuracy: 0.8445 - loss: 0.3846 - val_
accuracy: 0.8565 - val_loss: 0.3463
```

In [ ]:
```
# helper function to annotate maximum values in the plots
def annot_max(x,y, xytext=(0.94,0.96), ax=None, only_y=True):
    xmax = x[np.argmax(y)]
    ymax = max(y)
    if only_y:
        text = "{:.2f}%".format(ymax)
    else:
        text= "x={:.2f}, y={:.2f}%".format(xmax, ymax)
```

```
    if not ax:
        ax=plt.gca()
    bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
    arrowprops=dict(arrowstyle="->",connectionstyle="angle,angleA=0,angleB=60")
    kw = dict(xycoords='data',textcoords="axes fraction",
              arrowprops=arrowprops, bbox=bbox_props, ha="right", va="top")
    ax.annotate(text, xy=(xmax, ymax), xytext=xytext, **kw)

def plot_accuracy(model_fit):
    #accuracy graph
    x = range(0,len(model_fit.history['accuracy']))
    y_train = [acc * 100 for acc in model_fit.history['accuracy']]
    y_val = [acc * 100 for acc in model_fit.history['val_accuracy']]

    plt.plot(x, y_train, label='Train', color='b')
    annot_max(x, y_train, xytext=(0.7,0.9))
    plt.plot(x, y_val, label='Val', color='r')
    annot_max(x, y_val, xytext=(0.8,0.7))
    plt.ylabel('Accuracy', fontsize=15)
    plt.xlabel('epoch', fontsize=15)
    plt.legend()
    plt.show()
```

In [ ]:
```
plot_accuracy(model_fit)
```

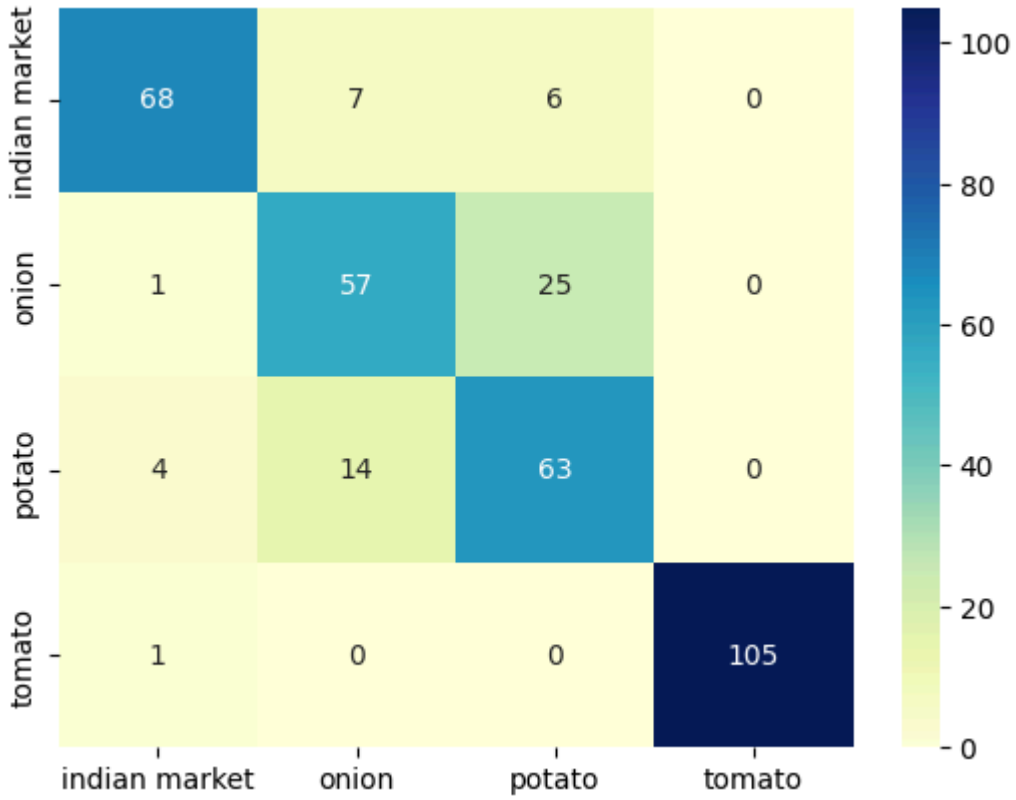



In [ ]:
```
def print_accuracy_stats(model, ds, class_names, weights_path):
    model.load_weights(weights_path)

    # Collect true labels by iterating through the dataset
    true_categories_list = []
    for _, labels in ds:
        # Check rank using tf.rank and compare as integers
        if tf.rank(labels).numpy() > 1:
```

```
            true_categories_list.append(tf.argmax(labels, axis=1).numpy())
        else:
            true_categories_list.append(labels.numpy())

    true_categories = np.concatenate(true_categories_list, axis=0)


    y_pred = model.predict(ds)
    predicted_categories = tf.argmax(y_pred, axis=1)

    test_acc = metrics.accuracy_score(true_categories, predicted_categories) * 1
    print(f'\nTest Accuracy: {test_acc:.2f}%\n')

# Note: This doesn't work with shuffled datasets - This note is still relevant f
def plot_confusion_matrix(model, ds, class_names, weights_path):
    model.load_weights(weights_path)

    # Collect true labels by iterating through the dataset
    true_categories_list = []
    for _, labels in ds:
        # Check rank using tf.rank and compare as integers
        if tf.rank(labels).numpy() > 1:
            true_categories_list.append(tf.argmax(labels, axis=1).numpy())
        else:
            true_categories_list.append(labels.numpy())

    true_categories = np.concatenate(true_categories_list, axis=0)

    y_pred = model.predict(ds)
    predicted_categories = tf.argmax(y_pred, axis=1)
    cm = metrics.confusion_matrix(true_categories,predicted_categories)
    sns.heatmap(cm, annot=True, xticklabels=class_names, yticklabels=class_names
    plt.show()
```

In [ ]:
```
print_accuracy_stats(model, test_ds_batched, class_names, ckpt_path)
plot_confusion_matrix(model, test_ds_batched, class_names, ckpt_path)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 130ms/step

Test Accuracy: 83.48%

12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 91ms/step
```



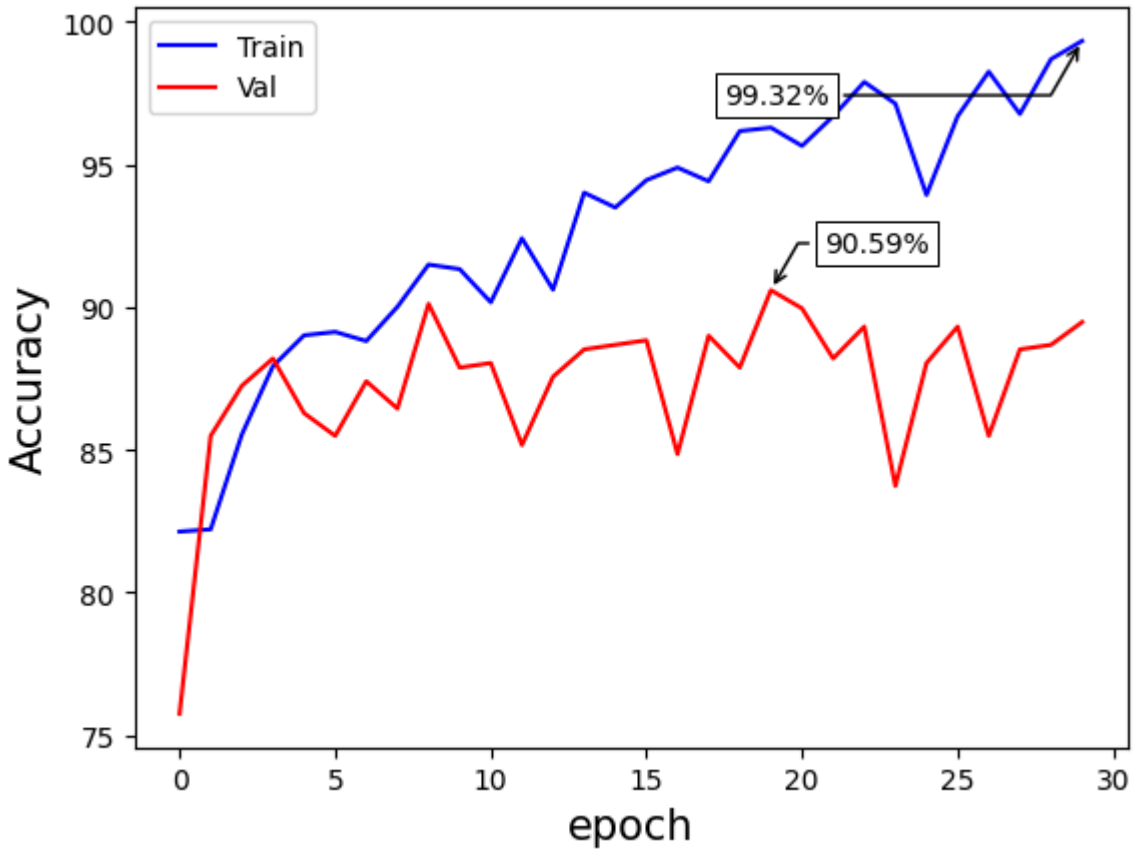

In [ ]:
```
ckpt_path="/content/ninjacart_models/model2_with_GAP/gap_model_30_epochs.weights
callbacks=[
        keras.callbacks.ModelCheckpoint(ckpt_path, save_weights_only=True, monit
    ]
```

In [ ]:
```
# model_fit = compile_train_v1(model, train_ds_batched, validation_ds_batched, e
model_fit = compile_train_v1(model, train_ds_batched, validation_ds_batched,  ca
```

```
Epoch 1/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 12s 101ms/step - accuracy: 0.7902 - loss: 0.5468 - val
_accuracy: 0.7576 - val_loss: 0.5714
Epoch 2/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 91ms/step - accuracy: 0.8280 - loss: 0.4315 - val_a
ccuracy: 0.8549 - val_loss: 0.3476
Epoch 3/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 101ms/step - accuracy: 0.8555 - loss: 0.3695 - val_
accuracy: 0.8724 - val_loss: 0.3577
Epoch 4/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.8828 - loss: 0.3003 - val_a
ccuracy: 0.8820 - val_loss: 0.3140
Epoch 5/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8790 - loss: 0.2836 - val_a
ccuracy: 0.8628 - val_loss: 0.3941
Epoch 6/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 78ms/step - accuracy: 0.8956 - loss: 0.2710 - val_a
ccuracy: 0.8549 - val_loss: 0.3926
Epoch 7/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8827 - loss: 0.2906 - val_a
ccuracy: 0.8740 - val_loss: 0.3982
Epoch 8/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 78ms/step - accuracy: 0.9042 - loss: 0.2550 - val_a
ccuracy: 0.8644 - val_loss: 0.3169
Epoch 9/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9094 - loss: 0.2134 - val_a
ccuracy: 0.9011 - val_loss: 0.3030
Epoch 10/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 77ms/step - accuracy: 0.9192 - loss: 0.2035 - val_a
ccuracy: 0.8788 - val_loss: 0.3984
Epoch 11/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8906 - loss: 0.2831 - val_a
ccuracy: 0.8804 - val_loss: 0.3886
Epoch 12/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 94ms/step - accuracy: 0.9310 - loss: 0.1905 - val_a
ccuracy: 0.8517 - val_loss: 0.3802
Epoch 13/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 83ms/step - accuracy: 0.9038 - loss: 0.2302 - val_a
ccuracy: 0.8756 - val_loss: 0.3763
Epoch 14/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 109ms/step - accuracy: 0.9268 - loss: 0.1815 - val_
accuracy: 0.8852 - val_loss: 0.3459
Epoch 15/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9419 - loss: 0.1509 - val_a
ccuracy: 0.8868 - val_loss: 0.3082
Epoch 16/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9462 - loss: 0.1356 - val_a
ccuracy: 0.8884 - val_loss: 0.3958
Epoch 17/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 78ms/step - accuracy: 0.9519 - loss: 0.1133 - val_a
ccuracy: 0.8485 - val_loss: 0.4256
Epoch 18/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9367 - loss: 0.1665 - val_a
ccuracy: 0.8900 - val_loss: 0.3541
Epoch 19/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.9567 - loss: 0.1024 - val_a
ccuracy: 0.8788 - val_loss: 0.4244
Epoch 20/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 81ms/step - accuracy: 0.9664 - loss: 0.0905 - val_a
ccuracy: 0.9059 - val_loss: 0.3596
```

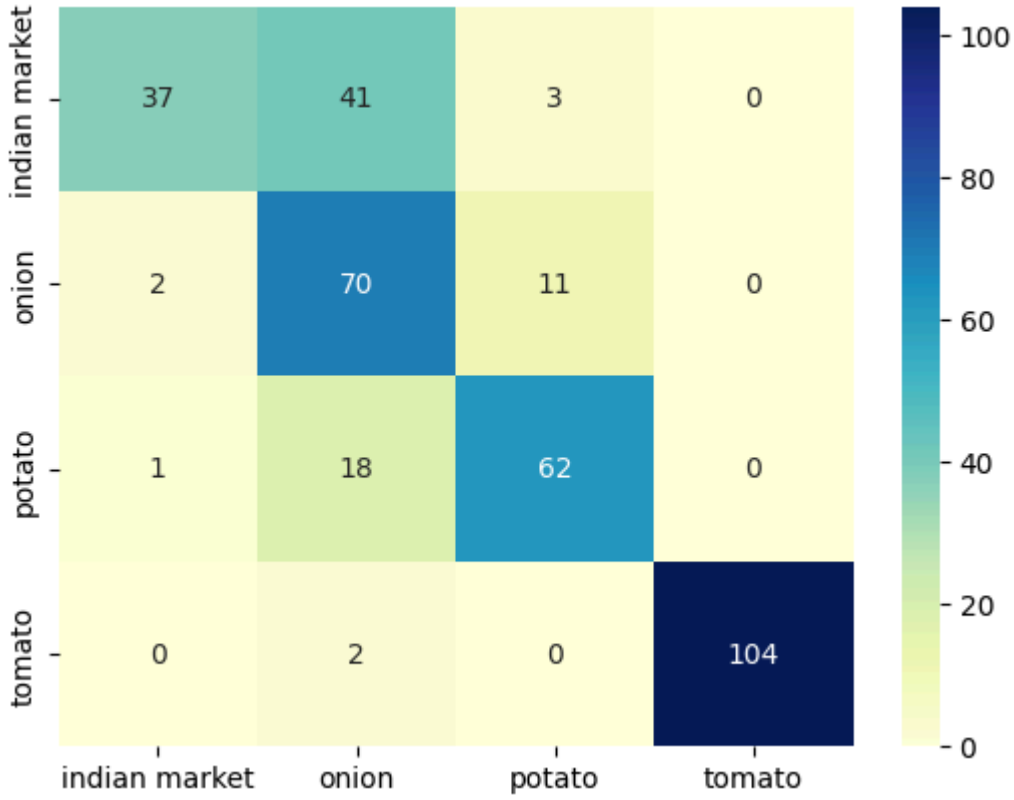
```
Epoch 21/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9515 - loss: 0.1391 - val_a
ccuracy: 0.8995 - val_loss: 0.4170
Epoch 22/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9771 - loss: 0.0608 - val_a
ccuracy: 0.8820 - val_loss: 0.3782
Epoch 23/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9723 - loss: 0.0752 - val_a
ccuracy: 0.8931 - val_loss: 0.4028
Epoch 24/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.9808 - loss: 0.0562 - val_a
ccuracy: 0.8373 - val_loss: 0.7279
Epoch 25/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 79ms/step - accuracy: 0.9191 - loss: 0.2369 - val_a
ccuracy: 0.8804 - val_loss: 0.3938
Epoch 26/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9757 - loss: 0.0714 - val_a
ccuracy: 0.8931 - val_loss: 0.4045
Epoch 27/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9809 - loss: 0.0687 - val_a
ccuracy: 0.8549 - val_loss: 0.7350
Epoch 28/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9682 - loss: 0.0872 - val_a
ccuracy: 0.8852 - val_loss: 0.4622
Epoch 29/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 108ms/step - accuracy: 0.9798 - loss: 0.0626 - val
_accuracy: 0.8868 - val_loss: 0.6122
Epoch 30/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9904 - loss: 0.0247 - val_a
ccuracy: 0.8947 - val_loss: 0.5931
```

In [ ]:
```
plot_accuracy(model_fit)
```



In [ ]:
```
print_accuracy_stats(model, test_ds_batched, class_names, ckpt_path)
plot_confusion_matrix(model, test_ds_batched, class_names, ckpt_path)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 109ms/step

Test Accuracy: 77.78%

12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 109ms/step
```




# Model3 - Using Dropout and Batch Normalization Layers

In [ ]:
```
def arch_2(height=128, width=128):
    num_classes = 4
    hidden_size = 256

    model = keras.Sequential(
        name="model_cnn_2",
        layers=[
            Input(shape=(height, width, 3)),
            layers.Conv2D(filters=16, kernel_size=3, padding="same"),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=32, kernel_size=3, padding="same"),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=64, kernel_size=3, padding="same"),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
```

```
            layers.Conv2D(filters=128, kernel_size=3, padding="same"),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=256, kernel_size=3, padding="same"),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            # layers.MaxPooling2D(),
            # layers.Flatten(),
            layers.GlobalAveragePooling2D(),
            layers.Dense(units=hidden_size),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.Dropout(0.5),
            layers.Dense(units=num_classes, activation='softmax')
        ]
    )
    return model
```

In [ ]:
```
model = arch_2()
model.summary()
```

**Model: "model_cnn_2"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_6 (Conv2D) | (None, 128, 128, 16) | 448 |
| activation (Activation) | (None, 128, 128, 16) | 0 |
| batch_normalization (BatchNormalization) | (None, 128, 128, 16) | 64 |
| max_pooling2d_5 (MaxPooling2D) | (None, 64, 64, 16) | 0 |
| conv2d_7 (Conv2D) | (None, 64, 64, 32) | 4,640 |
| activation_1 (Activation) | (None, 64, 64, 32) | 0 |
| batch_normalization_1 (BatchNormalization) | (None, 64, 64, 32) | 128 |
| max_pooling2d_6 (MaxPooling2D) | (None, 32, 32, 32) | 0 |
| conv2d_8 (Conv2D) | (None, 32, 32, 64) | 18,496 |
| activation_2 (Activation) | (None, 32, 32, 64) | 0 |
| batch_normalization_2 (BatchNormalization) | (None, 32, 32, 64) | 256 |
| max_pooling2d_7 (MaxPooling2D) | (None, 16, 16, 64) | 0 |
| conv2d_9 (Conv2D) | (None, 16, 16, 128) | 73,856 |
| activation_3 (Activation) | (None, 16, 16, 128) | 0 |
| batch_normalization_3 (BatchNormalization) | (None, 16, 16, 128) | 512 |
| max_pooling2d_8 (MaxPooling2D) | (None, 8, 8, 128) | 0 |
| conv2d_10 (Conv2D) | (None, 8, 8, 256) | 295,168 |
| activation_4 (Activation) | (None, 8, 8, 256) | 0 |
| batch_normalization_4 (BatchNormalization) | (None, 8, 8, 256) | 1,024 |
| global_average_pooling2d_1 (GlobalAveragePooling2D) | (None, 256) | 0 |
| dense_4 (Dense) | (None, 256) | 65,792 |
| activation_5 (Activation) | (None, 256) | 0 |
| batch_normalization_5 (BatchNormalization) | (None, 256) | 1,024 |
| dropout (Dropout) | (None, 256) | 0 |

| dense_5 (Dense) | (None, 4) | 1,028 |
|---|---|---|

**Total params:** 462,436 (1.76 MB)

**Trainable params:** 460,932 (1.76 MB)

**Non-trainable params:** 1,504 (5.88 KB)

In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model3_with_GAP_BN_DO'
ckpt_path="/content/ninjacart_models/model3_with_GAP_BN_DO/model3_with_GAP_BN_DO
callbacks=[
        keras.callbacks.ModelCheckpoint(ckpt_path, save_weights_only=True, monit
    ]
```

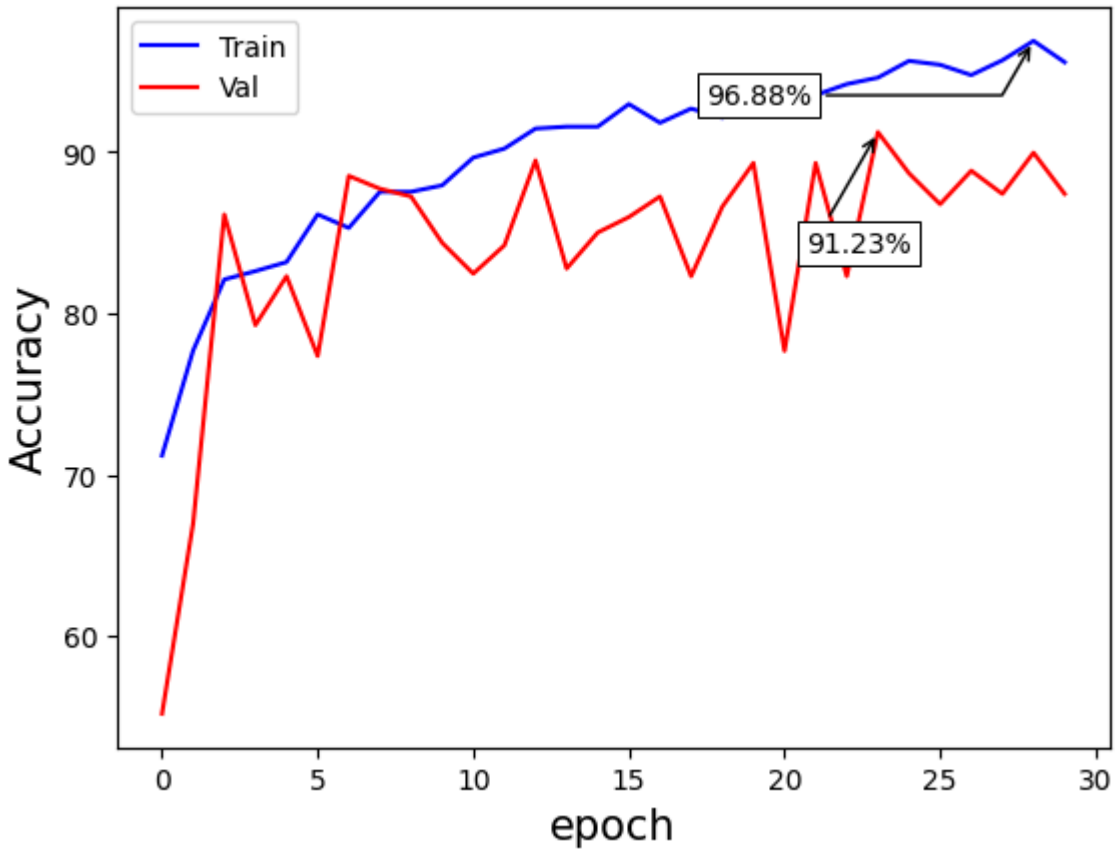
```
Created directory: /content/ninjacart_models/model3_with_GAP_BN_DO
```

In [ ]:
```
model_fit = compile_train_v1(model, train_ds_batched, validation_ds_batched, cal
```

```
Epoch 1/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 20s 150ms/step - accuracy: 0.6507 - loss: 1.0929 - val
_accuracy: 0.5518 - val_loss: 1.2198
Epoch 2/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.7661 - loss: 0.6533 - val_a
ccuracy: 0.6699 - val_loss: 1.0389
Epoch 3/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.8057 - loss: 0.5186 - val_a
ccuracy: 0.8612 - val_loss: 0.4023
Epoch 4/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.8229 - loss: 0.4633 - val_a
ccuracy: 0.7927 - val_loss: 0.5806
Epoch 5/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.8094 - loss: 0.4932 - val_a
ccuracy: 0.8230 - val_loss: 0.4313
Epoch 6/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8460 - loss: 0.3985 - val_a
ccuracy: 0.7735 - val_loss: 0.7846
Epoch 7/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.8386 - loss: 0.3958 - val_a
ccuracy: 0.8852 - val_loss: 0.2918
Epoch 8/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 83ms/step - accuracy: 0.8824 - loss: 0.3383 - val_a
ccuracy: 0.8772 - val_loss: 0.3710
Epoch 9/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.8642 - loss: 0.3824 - val_a
ccuracy: 0.8724 - val_loss: 0.3984
Epoch 10/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 78ms/step - accuracy: 0.8717 - loss: 0.3053 - val_a
ccuracy: 0.8437 - val_loss: 0.3744
Epoch 11/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.8925 - loss: 0.2877 - val_a
ccuracy: 0.8246 - val_loss: 0.5320
Epoch 12/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.8938 - loss: 0.2657 - val_a
ccuracy: 0.8421 - val_loss: 0.4545
Epoch 13/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9018 - loss: 0.2628 - val_a
ccuracy: 0.8947 - val_loss: 0.3347
Epoch 14/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9276 - loss: 0.2057 - val_a
ccuracy: 0.8278 - val_loss: 0.4184
Epoch 15/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9081 - loss: 0.2518 - val_a
ccuracy: 0.8501 - val_loss: 0.3844
Epoch 16/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 90ms/step - accuracy: 0.9260 - loss: 0.2272 - val_a
ccuracy: 0.8596 - val_loss: 0.4256
Epoch 17/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 80ms/step - accuracy: 0.9094 - loss: 0.2412 - val_a
ccuracy: 0.8724 - val_loss: 0.3552
Epoch 18/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9356 - loss: 0.1895 - val_a
ccuracy: 0.8230 - val_loss: 0.6499
Epoch 19/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9206 - loss: 0.2080 - val_a
ccuracy: 0.8660 - val_loss: 0.4578
Epoch 20/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9264 - loss: 0.1801 - val_a
ccuracy: 0.8931 - val_loss: 0.3578
```

```
Epoch 21/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 92ms/step - accuracy: 0.9322 - loss: 0.1737 - val_a
ccuracy: 0.7767 - val_loss: 0.5710
Epoch 22/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 15s 153ms/step - accuracy: 0.9345 - loss: 0.1684 - val
_accuracy: 0.8931 - val_loss: 0.2970
Epoch 23/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 16s 98ms/step - accuracy: 0.9503 - loss: 0.1410 - val_
accuracy: 0.8230 - val_loss: 0.6571
Epoch 24/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.9480 - loss: 0.1369 - val_a
ccuracy: 0.9123 - val_loss: 0.2605
Epoch 25/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9502 - loss: 0.1227 - val_a
ccuracy: 0.8868 - val_loss: 0.3576
Epoch 26/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.9598 - loss: 0.1099 - val_a
ccuracy: 0.8676 - val_loss: 0.4969
Epoch 27/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 79ms/step - accuracy: 0.9427 - loss: 0.1796 - val_a
ccuracy: 0.8884 - val_loss: 0.5453
Epoch 28/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 113ms/step - accuracy: 0.9491 - loss: 0.1484 - val
_accuracy: 0.8740 - val_loss: 0.4570
Epoch 29/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 88ms/step - accuracy: 0.9685 - loss: 0.1061 - val_a
ccuracy: 0.8995 - val_loss: 0.4009
Epoch 30/30
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 80ms/step - accuracy: 0.9573 - loss: 0.1091 - val_
accuracy: 0.8740 - val_loss: 0.3829
```

In [ ]:
```
plot_accuracy(model_fit)
```



In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model4_with_GAP_BN_DO_
ckpt_path = '/content/ninjacart_models/model4_with_GAP_BN_DO_CB/model4_with_GAP_
callbacks = [
        keras.callbacks.ReduceLROnPlateau(
            monitor="val_loss", factor=0.3, patience=5, min_lr=0.00001
        ),
        keras.callbacks.ModelCheckpoint(ckpt_path, save_weights_only=True, monit
        keras.callbacks.EarlyStopping(
            monitor="val_loss", patience=10, min_delta=0.001, mode='min'
        )]
```

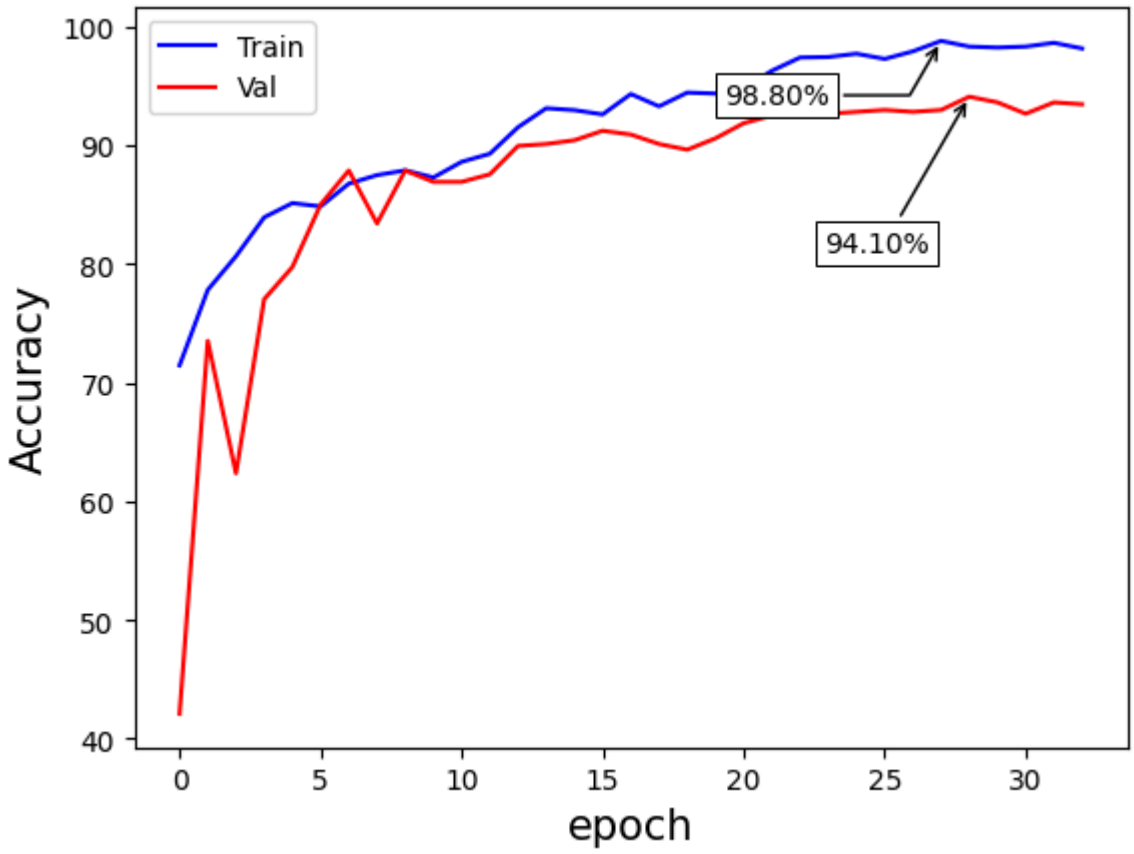
```
Created directory: /content/ninjacart_models/model4_with_GAP_BN_DO_CB
```

In [ ]:
```
model = arch_2()
model_fit = compile_train_v1(model, train_ds_batched, validation_ds_batched, cal
```

```
Epoch 1/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 20s 148ms/step - accuracy: 0.6546 - loss: 1.0235 - val
_accuracy: 0.4211 - val_loss: 1.6457 - learning_rate: 0.0010
Epoch 2/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 12s 84ms/step - accuracy: 0.7689 - loss: 0.7054 - val_
accuracy: 0.7352 - val_loss: 0.6958 - learning_rate: 0.0010
Epoch 3/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 97ms/step - accuracy: 0.7913 - loss: 0.5327 - val_
accuracy: 0.6236 - val_loss: 1.0406 - learning_rate: 0.0010
Epoch 4/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.8357 - loss: 0.4497 - val_a
ccuracy: 0.7703 - val_loss: 0.8803 - learning_rate: 0.0010
Epoch 5/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 85ms/step - accuracy: 0.8478 - loss: 0.3982 - val_a
ccuracy: 0.7974 - val_loss: 0.6586 - learning_rate: 0.0010
Epoch 6/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 112ms/step - accuracy: 0.8367 - loss: 0.4439 - val_
accuracy: 0.8501 - val_loss: 0.3897 - learning_rate: 0.0010
Epoch 7/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.8771 - loss: 0.3185 - val_a
ccuracy: 0.8788 - val_loss: 0.2904 - learning_rate: 0.0010
Epoch 8/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8783 - loss: 0.3194 - val_a
ccuracy: 0.8341 - val_loss: 0.4680 - learning_rate: 0.0010
Epoch 9/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.8830 - loss: 0.3105 - val_a
ccuracy: 0.8788 - val_loss: 0.2962 - learning_rate: 0.0010
Epoch 10/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8654 - loss: 0.3260 - val_a
ccuracy: 0.8692 - val_loss: 0.4599 - learning_rate: 0.0010
Epoch 11/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 91ms/step - accuracy: 0.8762 - loss: 0.3073 - val_a
ccuracy: 0.8692 - val_loss: 0.3927 - learning_rate: 0.0010
Epoch 12/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 80ms/step - accuracy: 0.8945 - loss: 0.2942 - val_a
ccuracy: 0.8756 - val_loss: 0.3991 - learning_rate: 0.0010
Epoch 13/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9093 - loss: 0.2433 - val_a
ccuracy: 0.8995 - val_loss: 0.2802 - learning_rate: 3.0000e-04
Epoch 14/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.9249 - loss: 0.2029 - val_a
ccuracy: 0.9011 - val_loss: 0.2565 - learning_rate: 3.0000e-04
Epoch 15/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 80ms/step - accuracy: 0.9221 - loss: 0.2035 - val_
accuracy: 0.9043 - val_loss: 0.2977 - learning_rate: 3.0000e-04
Epoch 16/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9175 - loss: 0.2046 - val_a
ccuracy: 0.9123 - val_loss: 0.2378 - learning_rate: 3.0000e-04
Epoch 17/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9517 - loss: 0.1522 - val_a
ccuracy: 0.9091 - val_loss: 0.2763 - learning_rate: 3.0000e-04
Epoch 18/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9307 - loss: 0.1715 - val_a
ccuracy: 0.9011 - val_loss: 0.2528 - learning_rate: 3.0000e-04
Epoch 19/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.9490 - loss: 0.1337 - val_a
ccuracy: 0.8963 - val_loss: 0.3023 - learning_rate: 3.0000e-04
Epoch 20/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 83ms/step - accuracy: 0.9389 - loss: 0.1866 - val_a
ccuracy: 0.9059 - val_loss: 0.3227 - learning_rate: 3.0000e-04
```

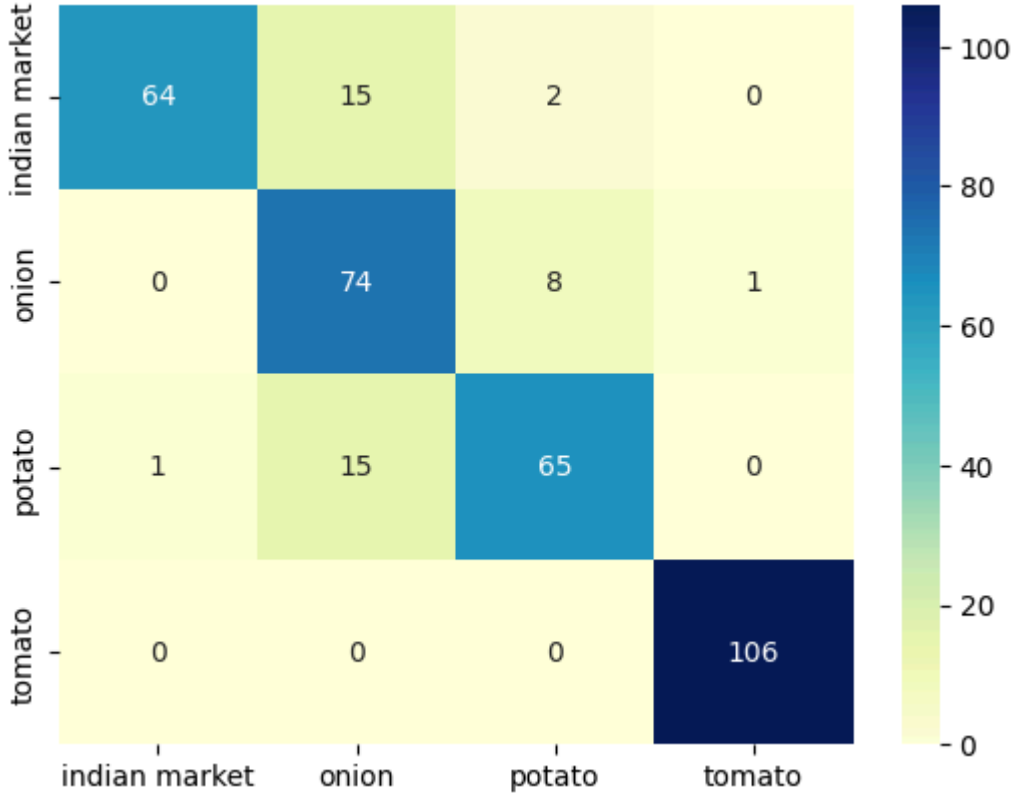
```
Epoch 21/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 99ms/step - accuracy: 0.9457 - loss: 0.1436 - val_
accuracy: 0.9187 - val_loss: 0.2626 - learning_rate: 3.0000e-04
Epoch 22/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 85ms/step - accuracy: 0.9604 - loss: 0.1075 - val_a
ccuracy: 0.9250 - val_loss: 0.2174 - learning_rate: 9.0000e-05
Epoch 23/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 94ms/step - accuracy: 0.9692 - loss: 0.0894 - val_a
ccuracy: 0.9314 - val_loss: 0.1972 - learning_rate: 9.0000e-05
Epoch 24/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 103ms/step - accuracy: 0.9780 - loss: 0.0664 - val_
accuracy: 0.9266 - val_loss: 0.2125 - learning_rate: 9.0000e-05
Epoch 25/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9762 - loss: 0.0690 - val_a
ccuracy: 0.9282 - val_loss: 0.2080 - learning_rate: 9.0000e-05
Epoch 26/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.9727 - loss: 0.0838 - val_a
ccuracy: 0.9298 - val_loss: 0.2088 - learning_rate: 9.0000e-05
Epoch 27/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9799 - loss: 0.0587 - val_a
ccuracy: 0.9282 - val_loss: 0.2380 - learning_rate: 9.0000e-05
Epoch 28/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9881 - loss: 0.0433 - val_a
ccuracy: 0.9298 - val_loss: 0.2251 - learning_rate: 9.0000e-05
Epoch 29/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9830 - loss: 0.0521 - val_a
ccuracy: 0.9410 - val_loss: 0.1993 - learning_rate: 2.7000e-05
Epoch 30/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9808 - loss: 0.0529 - val_a
ccuracy: 0.9362 - val_loss: 0.1995 - learning_rate: 2.7000e-05
Epoch 31/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 87ms/step - accuracy: 0.9810 - loss: 0.0450 - val_a
ccuracy: 0.9266 - val_loss: 0.2125 - learning_rate: 2.7000e-05
Epoch 32/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 79ms/step - accuracy: 0.9854 - loss: 0.0440 - val_
accuracy: 0.9362 - val_loss: 0.2238 - learning_rate: 2.7000e-05
Epoch 33/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9819 - loss: 0.0450 - val_a
ccuracy: 0.9346 - val_loss: 0.2128 - learning_rate: 2.7000e-05
```

In [ ]:
```
plot_accuracy(model_fit)
```




In [ ]:
```
print_accuracy_stats(model, test_ds_batched, class_names, ckpt_path)
plot_confusion_matrix(model, test_ds_batched, class_names, ckpt_path)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 113ms/step

Test Accuracy: 88.03%

12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 67ms/step
```



In [ ]:
```python
def arch_3(height=128, width=128):
    num_classes = 4
    hidden_size = 256

    model = keras.Sequential(
        name="model_cnn_3",
        layers=[
            layers.Conv2D(filters=16, kernel_size=3, padding="same", input_shape
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=32, kernel_size=3, padding="same",
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=64, kernel_size=3, padding="same",
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=128, kernel_size=3, padding="same",
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=256, kernel_size=3, padding="same",
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            # layers.MaxPooling2D(),
            # layers.Flatten(),
            layers.GlobalAveragePooling2D(),
            layers.Dense(units=hidden_size, kernel_regularizer=regularizers.l2(1
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.Dropout(0.5),
            layers.Dense(units=num_classes, activation='softmax')
        ]
    )
    return model
```

In [ ]:
```python
model = arch_3()
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.
py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
r. When using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

**Model: "model_cnn_3"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_16 (Conv2D) | (None, 128, 128, 16) | 448 |
| activation_12 (Activation) | (None, 128, 128, 16) | 0 |
| batch_normalization_12 (BatchNormalization) | (None, 128, 128, 16) | 64 |
| max_pooling2d_13 (MaxPooling2D) | (None, 64, 64, 16) | 0 |
| conv2d_17 (Conv2D) | (None, 64, 64, 32) | 4,640 |
| activation_13 (Activation) | (None, 64, 64, 32) | 0 |
| batch_normalization_13 (BatchNormalization) | (None, 64, 64, 32) | 128 |
| max_pooling2d_14 (MaxPooling2D) | (None, 32, 32, 32) | 0 |
| conv2d_18 (Conv2D) | (None, 32, 32, 64) | 18,496 |
| activation_14 (Activation) | (None, 32, 32, 64) | 0 |
| batch_normalization_14 (BatchNormalization) | (None, 32, 32, 64) | 256 |
| max_pooling2d_15 (MaxPooling2D) | (None, 16, 16, 64) | 0 |
| conv2d_19 (Conv2D) | (None, 16, 16, 128) | 73,856 |
| activation_15 (Activation) | (None, 16, 16, 128) | 0 |
| batch_normalization_15 (BatchNormalization) | (None, 16, 16, 128) | 512 |
| max_pooling2d_16 (MaxPooling2D) | (None, 8, 8, 128) | 0 |
| conv2d_20 (Conv2D) | (None, 8, 8, 256) | 295,168 |
| activation_16 (Activation) | (None, 8, 8, 256) | 0 |
| batch_normalization_16 (BatchNormalization) | (None, 8, 8, 256) | 1,024 |
| global_average_pooling2d_3 (GlobalAveragePooling2D) | (None, 256) | 0 |
| dense_8 (Dense) | (None, 256) | 65,792 |
| activation_17 (Activation) | (None, 256) | 0 |
| batch_normalization_17 (BatchNormalization) | (None, 256) | 1,024 |
| dropout_2 (Dropout) | (None, 256) | 0 |

| dense_9 (Dense) | (None, 4) | 1,028 |
|---|---|---|

**Total params:** 462,436 (1.76 MB)

**Trainable params:** 460,932 (1.76 MB)

**Non-trainable params:** 1,504 (5.88 KB)

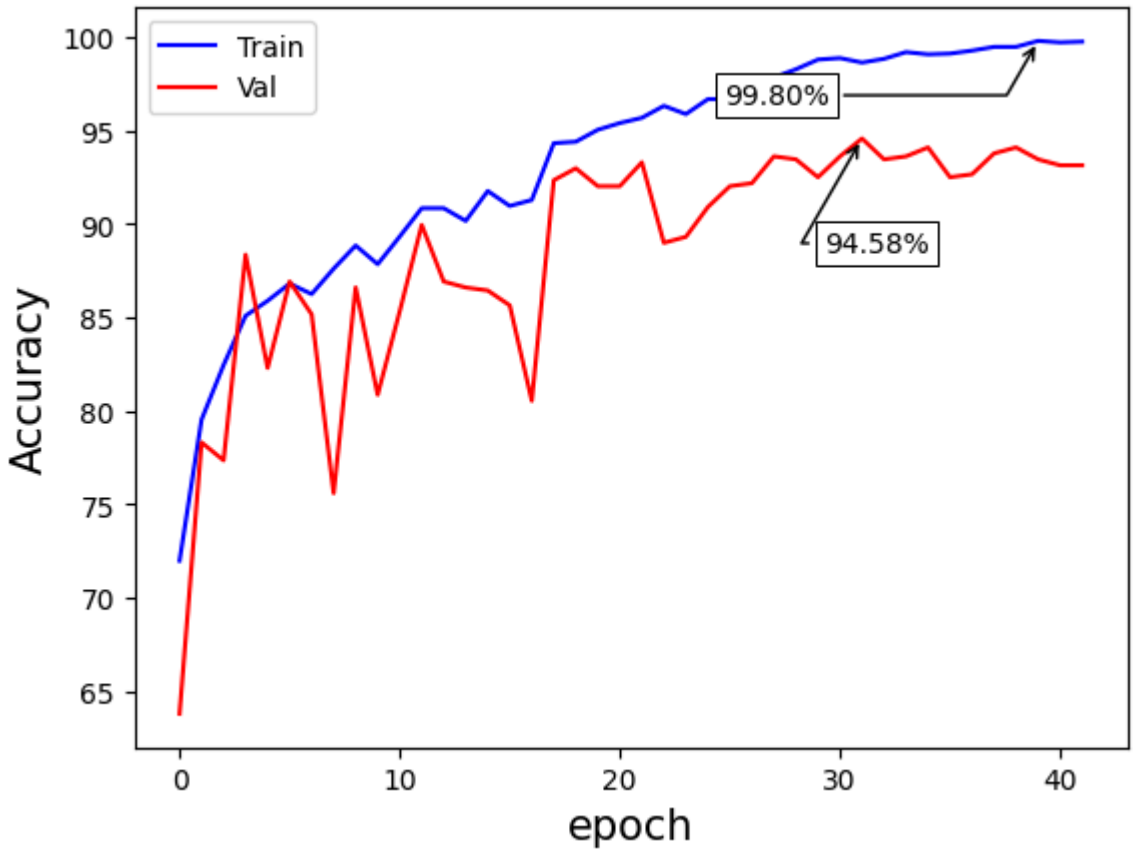
In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model5_with_GAP_BN_DO_
ckpt_path = '/content/ninjacart_models/model5_with_GAP_BN_DO_CB_LR/model5_with_G
callbacks = [
        keras.callbacks.ReduceLROnPlateau(
            monitor="val_loss", factor=0.3, patience=5, min_lr=0.00001
        ),
        keras.callbacks.ModelCheckpoint(ckpt_path, save_weights_only=True, monit
        keras.callbacks.EarlyStopping(
            monitor="val_loss", patience=10, min_delta=0.001, mode='min'
        )]
model_fit = compile_train_v1(model, train_ds_batched, validation_ds_batched, cal
```

```
Created directory: /content/ninjacart_models/model5_with_GAP_BN_DO_CB_LR
Epoch 1/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 23s 161ms/step - accuracy: 0.6537 - loss: 1.6525 - val
_accuracy: 0.6380 - val_loss: 1.6854 - learning_rate: 0.0010
Epoch 2/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.7872 - loss: 1.1510 - val_a
ccuracy: 0.7831 - val_loss: 1.3219 - learning_rate: 0.0010
Epoch 3/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.8129 - loss: 1.0824 - val_a
ccuracy: 0.7735 - val_loss: 1.2467 - learning_rate: 0.0010
Epoch 4/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.8313 - loss: 1.0211 - val_a
ccuracy: 0.8836 - val_loss: 0.8421 - learning_rate: 0.0010
Epoch 5/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.8448 - loss: 0.9219 - val_a
ccuracy: 0.8230 - val_loss: 1.1713 - learning_rate: 0.0010
Epoch 6/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8664 - loss: 0.8048 - val_a
ccuracy: 0.8692 - val_loss: 0.8172 - learning_rate: 0.0010
Epoch 7/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.8644 - loss: 0.8291 - val_a
ccuracy: 0.8517 - val_loss: 0.9049 - learning_rate: 0.0010
Epoch 8/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.8750 - loss: 0.7582 - val_a
ccuracy: 0.7560 - val_loss: 1.2067 - learning_rate: 0.0010
Epoch 9/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.8830 - loss: 0.7068 - val_a
ccuracy: 0.8660 - val_loss: 0.7985 - learning_rate: 0.0010
Epoch 10/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 83ms/step - accuracy: 0.8693 - loss: 0.7028 - val_a
ccuracy: 0.8086 - val_loss: 0.8129 - learning_rate: 0.0010
Epoch 11/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.8928 - loss: 0.6310 - val_a
ccuracy: 0.8533 - val_loss: 0.7042 - learning_rate: 0.0010
Epoch 12/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9026 - loss: 0.5855 - val_a
ccuracy: 0.8995 - val_loss: 0.5608 - learning_rate: 0.0010
Epoch 13/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9110 - loss: 0.5537 - val_a
ccuracy: 0.8692 - val_loss: 0.7774 - learning_rate: 0.0010
Epoch 14/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9010 - loss: 0.5425 - val_a
ccuracy: 0.8660 - val_loss: 0.6432 - learning_rate: 0.0010
Epoch 15/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9100 - loss: 0.5163 - val_a
ccuracy: 0.8644 - val_loss: 0.6661 - learning_rate: 0.0010
Epoch 16/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.9210 - loss: 0.4655 - val_a
ccuracy: 0.8565 - val_loss: 0.5637 - learning_rate: 0.0010
Epoch 17/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 84ms/step - accuracy: 0.9051 - loss: 0.5011 - val_a
ccuracy: 0.8054 - val_loss: 0.8058 - learning_rate: 0.0010
Epoch 18/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 109ms/step - accuracy: 0.9369 - loss: 0.4201 - val_
accuracy: 0.9234 - val_loss: 0.4437 - learning_rate: 3.0000e-04
Epoch 19/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9350 - loss: 0.3728 - val_a
ccuracy: 0.9298 - val_loss: 0.4168 - learning_rate: 3.0000e-04
Epoch 20/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9502 - loss: 0.3490 - val_a
```

```
ccuracy: 0.9203 - val_loss: 0.4485 - learning_rate: 3.0000e-04
Epoch 21/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9515 - loss: 0.3416 - val_a
ccuracy: 0.9203 - val_loss: 0.4509 - learning_rate: 3.0000e-04
Epoch 22/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9605 - loss: 0.3122 - val_a
ccuracy: 0.9330 - val_loss: 0.3934 - learning_rate: 3.0000e-04
Epoch 23/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 86ms/step - accuracy: 0.9635 - loss: 0.3083 - val_a
ccuracy: 0.8900 - val_loss: 0.5436 - learning_rate: 3.0000e-04
Epoch 24/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 79ms/step - accuracy: 0.9601 - loss: 0.2977 - val_
accuracy: 0.8931 - val_loss: 0.4722 - learning_rate: 3.0000e-04
Epoch 25/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9714 - loss: 0.2747 - val_a
ccuracy: 0.9091 - val_loss: 0.5256 - learning_rate: 3.0000e-04
Epoch 26/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9654 - loss: 0.2719 - val_a
ccuracy: 0.9203 - val_loss: 0.4288 - learning_rate: 3.0000e-04
Epoch 27/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.9797 - loss: 0.2400 - val_a
ccuracy: 0.9219 - val_loss: 0.4566 - learning_rate: 3.0000e-04
Epoch 28/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 85ms/step - accuracy: 0.9779 - loss: 0.2311 - val_a
ccuracy: 0.9362 - val_loss: 0.4018 - learning_rate: 9.0000e-05
Epoch 29/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 91ms/step - accuracy: 0.9861 - loss: 0.2193 - val_a
ccuracy: 0.9346 - val_loss: 0.3686 - learning_rate: 9.0000e-05
Epoch 30/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.9876 - loss: 0.2118 - val_a
ccuracy: 0.9250 - val_loss: 0.4008 - learning_rate: 9.0000e-05
Epoch 31/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 80ms/step - accuracy: 0.9908 - loss: 0.2017 - val_a
ccuracy: 0.9362 - val_loss: 0.3642 - learning_rate: 9.0000e-05
Epoch 32/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9910 - loss: 0.1995 - val_a
ccuracy: 0.9458 - val_loss: 0.3479 - learning_rate: 9.0000e-05
Epoch 33/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9926 - loss: 0.1952 - val_a
ccuracy: 0.9346 - val_loss: 0.3743 - learning_rate: 9.0000e-05
Epoch 34/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 80ms/step - accuracy: 0.9902 - loss: 0.1939 - val_
accuracy: 0.9362 - val_loss: 0.3631 - learning_rate: 9.0000e-05
Epoch 35/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 82ms/step - accuracy: 0.9895 - loss: 0.1943 - val_
accuracy: 0.9410 - val_loss: 0.3774 - learning_rate: 9.0000e-05
Epoch 36/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9935 - loss: 0.1818 - val_a
ccuracy: 0.9250 - val_loss: 0.4271 - learning_rate: 9.0000e-05
Epoch 37/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.9926 - loss: 0.1842 - val_a
ccuracy: 0.9266 - val_loss: 0.3950 - learning_rate: 9.0000e-05
Epoch 38/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9970 - loss: 0.1730 - val_a
ccuracy: 0.9378 - val_loss: 0.3673 - learning_rate: 2.7000e-05
Epoch 39/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9942 - loss: 0.1788 - val_a
ccuracy: 0.9410 - val_loss: 0.3755 - learning_rate: 2.7000e-05
Epoch 40/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9986 - loss: 0.1673 - val_a
```

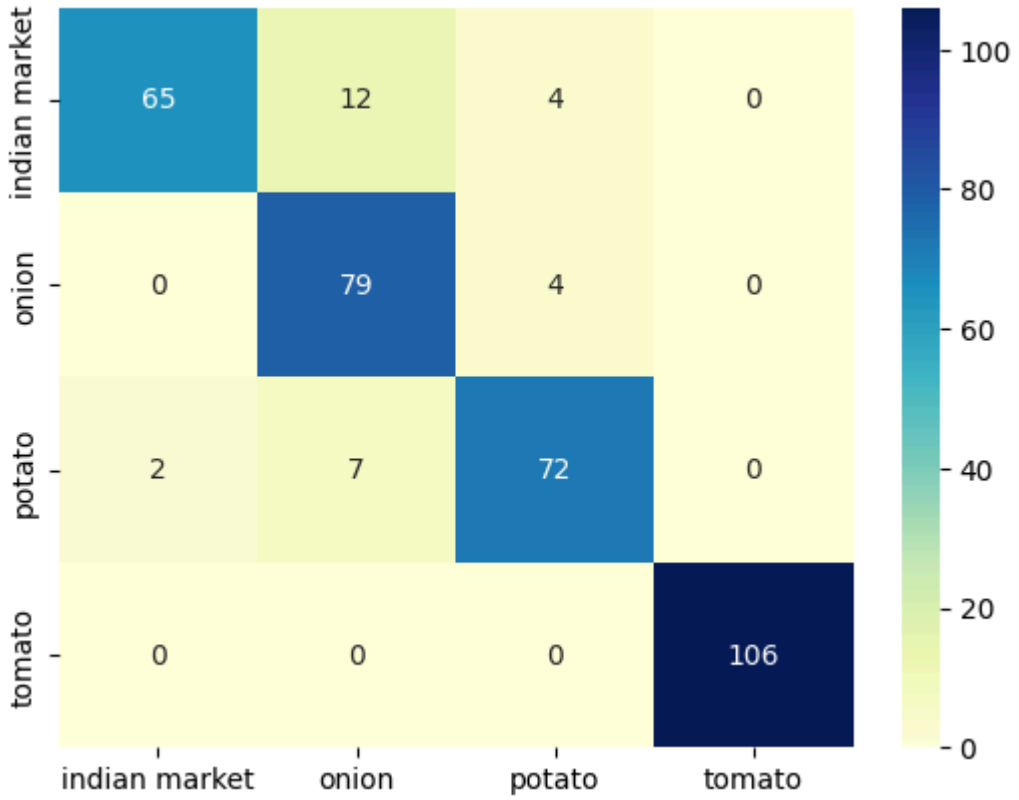
```
ccuracy: 0.9346 - val_loss: 0.3809 - learning_rate: 2.7000e-05
Epoch 41/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.9970 - loss: 0.1662 - val_a
ccuracy: 0.9314 - val_loss: 0.3748 - learning_rate: 2.7000e-05
Epoch 42/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 80ms/step - accuracy: 0.9986 - loss: 0.1646 - val_a
ccuracy: 0.9314 - val_loss: 0.3848 - learning_rate: 2.7000e-05
```

In [ ]:
```
plot_accuracy(model_fit)
```




In [ ]:
```
print_accuracy_stats(model, test_ds_batched, class_names, ckpt_path)
plot_confusion_matrix(model, test_ds_batched, class_names, ckpt_path)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 165ms/step

Test Accuracy: 91.74%

12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 84ms/step
```




In [ ]:

```
def arch_3(height=128, width=128):
    num_classes = 4
    hidden_size = 256

    model = keras.Sequential(
        name="model_cnn_3",
        layers=[
            layers.Conv2D(filters=16, kernel_size=3, padding="same", input_shape
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=32, kernel_size=3, padding="same",
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=64, kernel_size=3, padding="same",
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=128, kernel_size=3, padding="same",
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.MaxPooling2D(),
            layers.Conv2D(filters=256, kernel_size=3, padding="same",
                          kernel_regularizer=regularizers.l2(1e-3)),
            layers.Activation("relu"),
            layers.BatchNormalization(),
            # layers.MaxPooling2D(),
            # layers.Flatten(),
            layers.GlobalAveragePooling2D(),
```

```
            layers.Dense(units=hidden_size, kernel_regularizer=regularizers.l2(1
            layers.Activation("relu"),
            layers.BatchNormalization(),
            layers.Dropout(0.5),
            layers.Dense(units=num_classes, activation='softmax')
        ]
    )
    return model
```

In [ ]:
```
model = arch_3()
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.
py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
r. When using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

**Model: "model_cnn_3"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_21 (Conv2D) | (None, 128, 128, 16) | 448 |
| activation_18 (Activation) | (None, 128, 128, 16) | 0 |
| batch_normalization_18 (BatchNormalization) | (None, 128, 128, 16) | 64 |
| max_pooling2d_17 (MaxPooling2D) | (None, 64, 64, 16) | 0 |
| conv2d_22 (Conv2D) | (None, 64, 64, 32) | 4,640 |
| activation_19 (Activation) | (None, 64, 64, 32) | 0 |
| batch_normalization_19 (BatchNormalization) | (None, 64, 64, 32) | 128 |
| max_pooling2d_18 (MaxPooling2D) | (None, 32, 32, 32) | 0 |
| conv2d_23 (Conv2D) | (None, 32, 32, 64) | 18,496 |
| activation_20 (Activation) | (None, 32, 32, 64) | 0 |
| batch_normalization_20 (BatchNormalization) | (None, 32, 32, 64) | 256 |
| max_pooling2d_19 (MaxPooling2D) | (None, 16, 16, 64) | 0 |
| conv2d_24 (Conv2D) | (None, 16, 16, 128) | 73,856 |
| activation_21 (Activation) | (None, 16, 16, 128) | 0 |
| batch_normalization_21 (BatchNormalization) | (None, 16, 16, 128) | 512 |
| max_pooling2d_20 (MaxPooling2D) | (None, 8, 8, 128) | 0 |
| conv2d_25 (Conv2D) | (None, 8, 8, 256) | 295,168 |
| activation_22 (Activation) | (None, 8, 8, 256) | 0 |
| batch_normalization_22 (BatchNormalization) | (None, 8, 8, 256) | 1,024 |
| global_average_pooling2d_4 (GlobalAveragePooling2D) | (None, 256) | 0 |
| dense_10 (Dense) | (None, 256) | 65,792 |
| activation_23 (Activation) | (None, 256) | 0 |
| batch_normalization_23 (BatchNormalization) | (None, 256) | 1,024 |
| dropout_3 (Dropout) | (None, 256) | 0 |

| dense_11 (Dense) | (None, 4) | 1,028 |
|---|---|---|

**Total params:** 462,436 (1.76 MB)

**Trainable params:** 460,932 (1.76 MB)

**Non-trainable params:** 1,504 (5.88 KB)

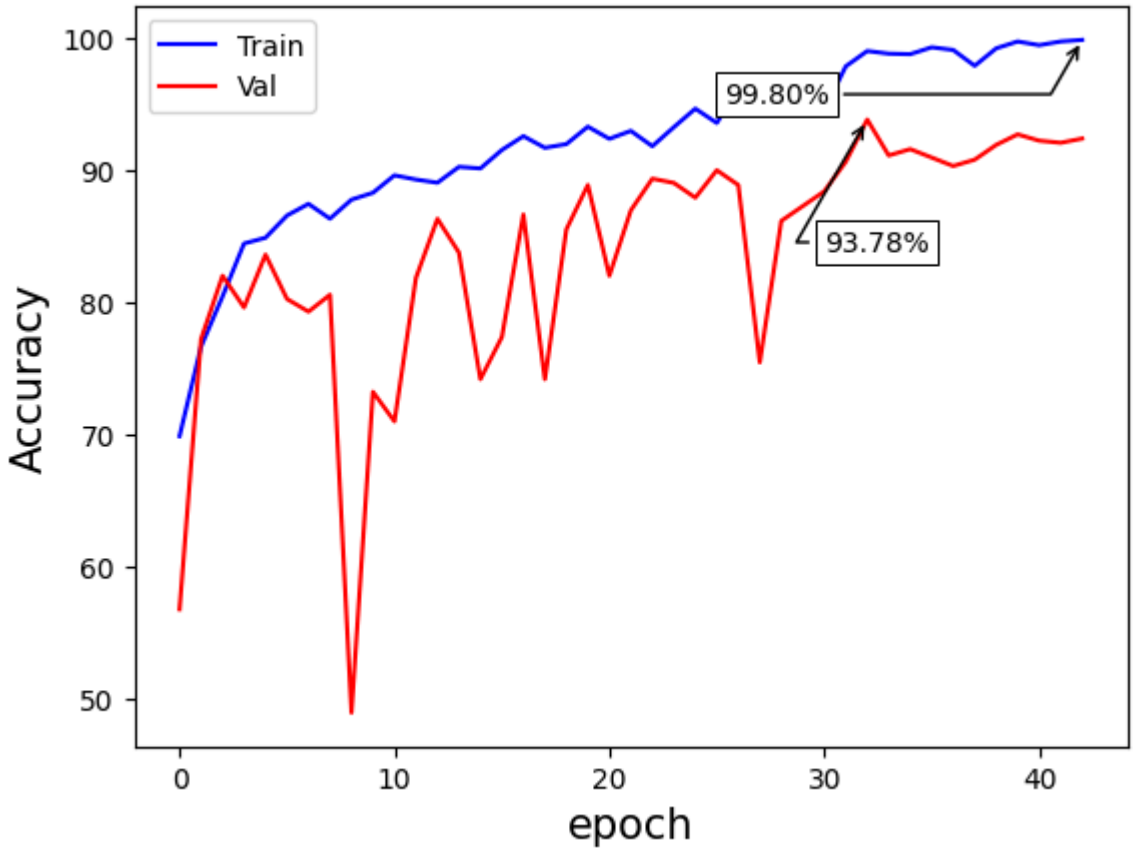
In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model6_with_GAP_BN_DO_
ckpt_path = '/content/ninjacart_models/model6_with_GAP_BN_DO_CB_LR_L2/model6_wit
callbacks = [
        keras.callbacks.ReduceLROnPlateau(
            monitor="val_loss", factor=0.3, patience=5, min_lr=0.00001
        ),
        keras.callbacks.ModelCheckpoint(ckpt_path, save_weights_only=True, monit
        keras.callbacks.EarlyStopping(
            monitor="val_loss", patience=10, min_delta=0.001, mode='min'
        )]
model_fit = compile_train_v1(model, train_ds_batched, validation_ds_batched, cal
```

```
Created directory: /content/ninjacart_models/model6_with_GAP_BN_DO_CB_LR_L2
Epoch 1/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 20s 151ms/step - accuracy: 0.6297 - loss: 1.6929 - val
_accuracy: 0.5678 - val_loss: 1.7282 - learning_rate: 0.0010
Epoch 2/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 100ms/step - accuracy: 0.7579 - loss: 1.2804 - val_
accuracy: 0.7719 - val_loss: 1.1630 - learning_rate: 0.0010
Epoch 3/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.8041 - loss: 1.1072 - val_a
ccuracy: 0.8198 - val_loss: 1.0329 - learning_rate: 0.0010
Epoch 4/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.8433 - loss: 0.9580 - val_a
ccuracy: 0.7959 - val_loss: 1.1170 - learning_rate: 0.0010
Epoch 5/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 85ms/step - accuracy: 0.8463 - loss: 0.9204 - val_a
ccuracy: 0.8357 - val_loss: 1.0488 - learning_rate: 0.0010
Epoch 6/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 81ms/step - accuracy: 0.8711 - loss: 0.8317 - val_
accuracy: 0.8022 - val_loss: 1.0198 - learning_rate: 0.0010
Epoch 7/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.8689 - loss: 0.8218 - val_a
ccuracy: 0.7927 - val_loss: 1.1416 - learning_rate: 0.0010
Epoch 8/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.8559 - loss: 0.8323 - val_a
ccuracy: 0.8054 - val_loss: 0.9580 - learning_rate: 0.0010
Epoch 9/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 101ms/step - accuracy: 0.8792 - loss: 0.7382 - val_
accuracy: 0.4896 - val_loss: 2.1495 - learning_rate: 0.0010
Epoch 10/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 93ms/step - accuracy: 0.8806 - loss: 0.6793 - val_a
ccuracy: 0.7321 - val_loss: 1.2706 - learning_rate: 0.0010
Epoch 11/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 97ms/step - accuracy: 0.8966 - loss: 0.6414 - val_
accuracy: 0.7097 - val_loss: 1.4492 - learning_rate: 0.0010
Epoch 12/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.8671 - loss: 0.6652 - val_a
ccuracy: 0.8182 - val_loss: 0.8225 - learning_rate: 0.0010
Epoch 13/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.8891 - loss: 0.6094 - val_a
ccuracy: 0.8628 - val_loss: 0.7391 - learning_rate: 0.0010
Epoch 14/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 81ms/step - accuracy: 0.9029 - loss: 0.5711 - val_a
ccuracy: 0.8373 - val_loss: 0.7184 - learning_rate: 0.0010
Epoch 15/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9057 - loss: 0.5347 - val_a
ccuracy: 0.7416 - val_loss: 0.8825 - learning_rate: 0.0010
Epoch 16/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.9065 - loss: 0.5151 - val_a
ccuracy: 0.7735 - val_loss: 0.8146 - learning_rate: 0.0010
Epoch 17/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.9224 - loss: 0.4532 - val_a
ccuracy: 0.8660 - val_loss: 0.6297 - learning_rate: 0.0010
Epoch 18/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.9084 - loss: 0.4649 - val_a
ccuracy: 0.7416 - val_loss: 1.1390 - learning_rate: 0.0010
Epoch 19/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 82ms/step - accuracy: 0.9109 - loss: 0.4821 - val_a
ccuracy: 0.8549 - val_loss: 0.7070 - learning_rate: 0.0010
Epoch 20/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 109ms/step - accuracy: 0.9409 - loss: 0.4057 - val_
```

```
accuracy: 0.8884 - val_loss: 0.5481 - learning_rate: 0.0010
Epoch 21/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9170 - loss: 0.4361 - val_a
ccuracy: 0.8198 - val_loss: 0.7245 - learning_rate: 0.0010
Epoch 22/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9331 - loss: 0.4084 - val_a
ccuracy: 0.8692 - val_loss: 0.5348 - learning_rate: 0.0010
Epoch 23/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9058 - loss: 0.4778 - val_a
ccuracy: 0.8931 - val_loss: 0.4944 - learning_rate: 0.0010
Epoch 24/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.9408 - loss: 0.3577 - val_a
ccuracy: 0.8900 - val_loss: 0.4988 - learning_rate: 0.0010
Epoch 25/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 84ms/step - accuracy: 0.9487 - loss: 0.3365 - val_a
ccuracy: 0.8788 - val_loss: 0.5545 - learning_rate: 0.0010
Epoch 26/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 92ms/step - accuracy: 0.9294 - loss: 0.3667 - val_a
ccuracy: 0.8995 - val_loss: 0.4603 - learning_rate: 0.0010
Epoch 27/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.9684 - loss: 0.2783 - val_a
ccuracy: 0.8884 - val_loss: 0.5072 - learning_rate: 0.0010
Epoch 28/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9457 - loss: 0.3128 - val_a
ccuracy: 0.7544 - val_loss: 0.8014 - learning_rate: 0.0010
Epoch 29/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 110ms/step - accuracy: 0.9562 - loss: 0.2888 - val_
accuracy: 0.8612 - val_loss: 0.6769 - learning_rate: 0.0010
Epoch 30/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9525 - loss: 0.2969 - val_a
ccuracy: 0.8724 - val_loss: 0.5400 - learning_rate: 0.0010
Epoch 31/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9509 - loss: 0.3005 - val_a
ccuracy: 0.8836 - val_loss: 0.5809 - learning_rate: 0.0010
Epoch 32/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9712 - loss: 0.2377 - val_a
ccuracy: 0.9059 - val_loss: 0.4371 - learning_rate: 3.0000e-04
Epoch 33/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 81ms/step - accuracy: 0.9929 - loss: 0.1937 - val_
accuracy: 0.9378 - val_loss: 0.3583 - learning_rate: 3.0000e-04
Epoch 34/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.9916 - loss: 0.1830 - val_a
ccuracy: 0.9107 - val_loss: 0.4522 - learning_rate: 3.0000e-04
Epoch 35/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 96ms/step - accuracy: 0.9880 - loss: 0.1873 - val_
accuracy: 0.9155 - val_loss: 0.4408 - learning_rate: 3.0000e-04
Epoch 36/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.9926 - loss: 0.1735 - val_a
ccuracy: 0.9091 - val_loss: 0.4858 - learning_rate: 3.0000e-04
Epoch 37/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 97ms/step - accuracy: 0.9928 - loss: 0.1713 - val_a
ccuracy: 0.9027 - val_loss: 0.5327 - learning_rate: 3.0000e-04
Epoch 38/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 81ms/step - accuracy: 0.9697 - loss: 0.2258 - val_a
ccuracy: 0.9075 - val_loss: 0.4925 - learning_rate: 3.0000e-04
Epoch 39/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.9800 - loss: 0.1942 - val_a
ccuracy: 0.9187 - val_loss: 0.4286 - learning_rate: 9.0000e-05
Epoch 40/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9966 - loss: 0.1477 - val_a
```

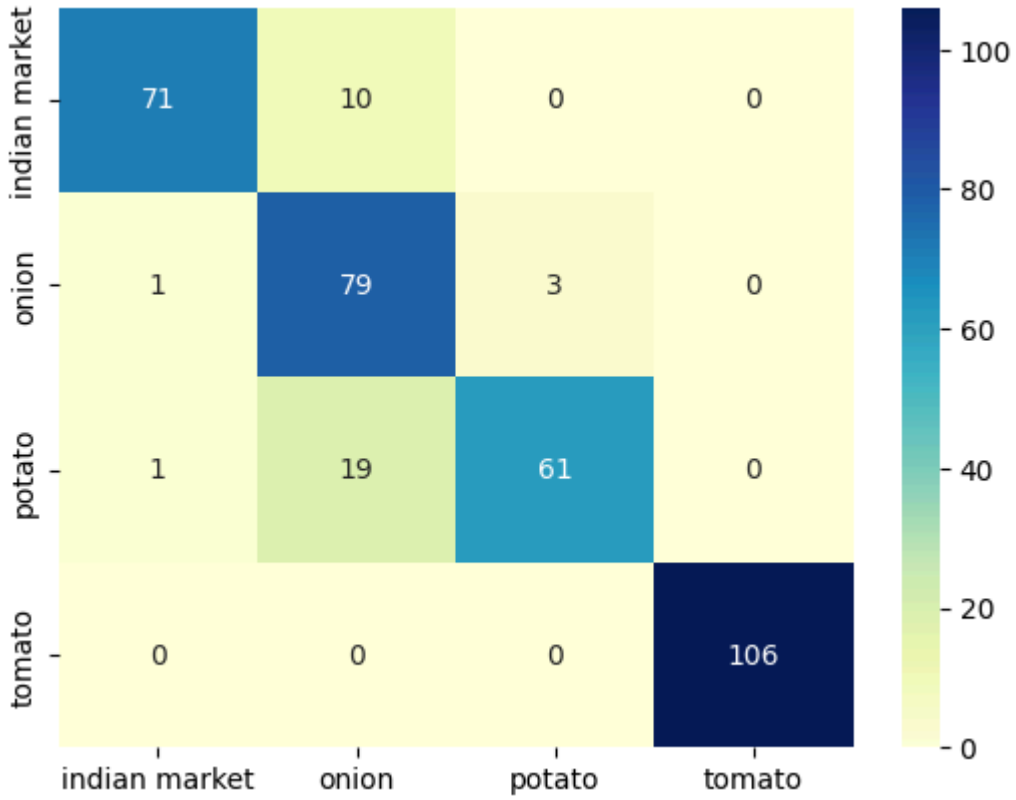
```
ccuracy: 0.9266 - val_loss: 0.4120 - learning_rate: 9.0000e-05
Epoch 41/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 98ms/step - accuracy: 0.9944 - loss: 0.1541 - val_a
ccuracy: 0.9219 - val_loss: 0.4128 - learning_rate: 9.0000e-05
Epoch 42/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9958 - loss: 0.1462 - val_a
ccuracy: 0.9203 - val_loss: 0.4367 - learning_rate: 9.0000e-05
Epoch 43/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 79ms/step - accuracy: 0.9975 - loss: 0.1564 - val_
accuracy: 0.9234 - val_loss: 0.4461 - learning_rate: 9.0000e-05
```

In [ ]:
```
plot_accuracy(model_fit)
```




In [ ]:
```
print_accuracy_stats(model, test_ds_batched, class_names, ckpt_path)
plot_confusion_matrix(model, test_ds_batched, class_names, ckpt_path)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 3s 208ms/step

Test Accuracy: 90.31%

12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 63ms/step
```




In [ ]:

```
def preprocess_v2(train_data, val_data, test_data, target_height=128, target_wid

    # Data Processing Stage with resizing and rescaling operations #same as befo
    data_preprocess = keras.Sequential(
        name="data_preprocess",
        layers=[
            layers.Resizing(target_height, target_width),
            layers.Rescaling(1.0/255),
        ]
    )

    # Data Processing Stage with resizing and rescaling operations
    data_augmentation = keras.Sequential(
        name="data_augmentation",
        layers=[
            layers.Resizing(156, 156), # First resize to 156,156
            layers.RandomCrop(target_height, target_width), # Then randomly crop
            # Layers.RandomBrightness(0.2), # Modify brightness by 0.2 factor
            layers.Rescaling(1.0/255), # Finally rescale
        ]
    )

    # Perform Data Processing on the train, val, test dataset
    train_ds = train_data.map(
        lambda x, y: (data_augmentation(x), y), num_parallel_calls=tf.data.AUTOT
    ).prefetch(tf.data.AUTOTUNE)
    val_ds = val_data.map(
        lambda x, y: (data_preprocess(x), y), num_parallel_calls=tf.data.AUTOTUN
    ).prefetch(tf.data.AUTOTUNE)
    test_ds = test_data.map(
        lambda x, y: (data_preprocess(x), y), num_parallel_calls=tf.data.AUTOTUN
    ).prefetch(tf.data.AUTOTUNE)

    return train_ds, val_ds, test_ds
```

In [ ]:
```
def keras_load_data_v2(base_path, target_paths=['train'], _label_mode='int', _ba
  keras_data_dict = {}
  for target_path in target_paths:
    _shuffle = True if target_path == 'train' else False

    print(f'Loading {target_path} data...')
    path = os.path.join(base_path, target_path)
    data = tf.keras.utils.image_dataset_from_directory(path, shuffle=_shuffle, i
    keras_data_dict.update({target_path: data})
  return keras_data_dict

# keras_loaded_data = keras_load_data(base_data_path, target_paths=['train', 'va
```

In [ ]:
```
base_data_path = 'ninjacart_data'
keras_loaded_data = keras_load_data(base_data_path, target_paths=['train', 'vali
```

```
Loading train data...
Found 2502 files belonging to 4 classes.
Loading validation data...
Found 627 files belonging to 4 classes.
Loading test data...
Found 351 files belonging to 4 classes.
```

In [ ]:
```
train_ds, val_ds, test_ds = preprocess_v2(keras_loaded_data['train'], keras_load
```

In [ ]:
```
model = arch_3()
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.
py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a laye
r. When using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

**Model: "model_cnn_3"**

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_26 (Conv2D) | (None, 128, 128, 16) | 448 |
| activation_24 (Activation) | (None, 128, 128, 16) | 0 |
| batch_normalization_24 (BatchNormalization) | (None, 128, 128, 16) | 64 |
| max_pooling2d_21 (MaxPooling2D) | (None, 64, 64, 16) | 0 |
| conv2d_27 (Conv2D) | (None, 64, 64, 32) | 4,640 |
| activation_25 (Activation) | (None, 64, 64, 32) | 0 |
| batch_normalization_25 (BatchNormalization) | (None, 64, 64, 32) | 128 |
| max_pooling2d_22 (MaxPooling2D) | (None, 32, 32, 32) | 0 |
| conv2d_28 (Conv2D) | (None, 32, 32, 64) | 18,496 |
| activation_26 (Activation) | (None, 32, 32, 64) | 0 |
| batch_normalization_26 (BatchNormalization) | (None, 32, 32, 64) | 256 |
| max_pooling2d_23 (MaxPooling2D) | (None, 16, 16, 64) | 0 |
| conv2d_29 (Conv2D) | (None, 16, 16, 128) | 73,856 |
| activation_27 (Activation) | (None, 16, 16, 128) | 0 |
| batch_normalization_27 (BatchNormalization) | (None, 16, 16, 128) | 512 |
| max_pooling2d_24 (MaxPooling2D) | (None, 8, 8, 128) | 0 |
| conv2d_30 (Conv2D) | (None, 8, 8, 256) | 295,168 |
| activation_28 (Activation) | (None, 8, 8, 256) | 0 |
| batch_normalization_28 (BatchNormalization) | (None, 8, 8, 256) | 1,024 |
| global_average_pooling2d_5 (GlobalAveragePooling2D) | (None, 256) | 0 |
| dense_12 (Dense) | (None, 256) | 65,792 |
| activation_29 (Activation) | (None, 256) | 0 |
| batch_normalization_29 (BatchNormalization) | (None, 256) | 1,024 |
| dropout_4 (Dropout) | (None, 256) | 0 |

| dense_13 (Dense) | (None, 4) | 1,028 |
|---|---|---|

**Total params:** 462,436 (1.76 MB)

**Trainable params:** 460,932 (1.76 MB)

**Non-trainable params:** 1,504 (5.88 KB)

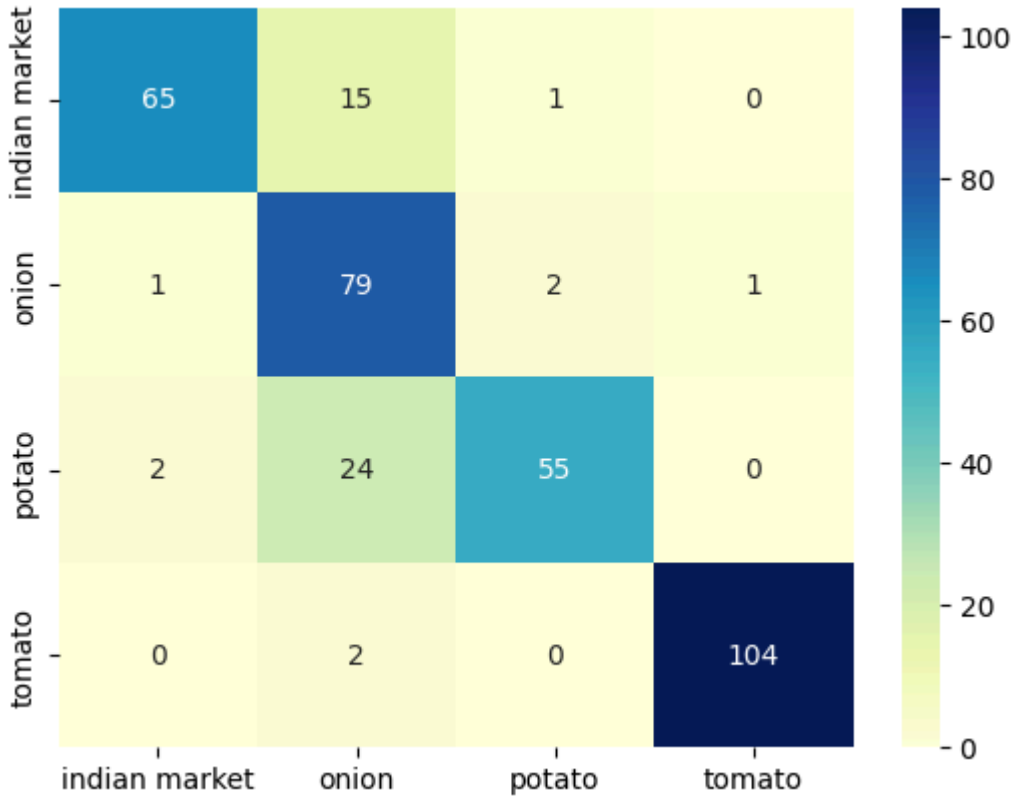
In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model7_agumentation')
ckpt_path = '/content/ninjacart_models/model7_agumentation/model7_agumentation.w
callbacks = [
        keras.callbacks.ReduceLROnPlateau(
            monitor="val_loss", factor=0.3, patience=5, min_lr=0.00001
        ),
        keras.callbacks.ModelCheckpoint(ckpt_path, save_weights_only=True, monit
        keras.callbacks.EarlyStopping(
            monitor="val_loss", patience=10, min_delta=0.001, mode='min'
        )]
model_fit = compile_train_v1(model, train_ds, val_ds, callbacks, epochs=100)
```

```
Created directory: /content/ninjacart_models/model7_agumentation
Epoch 1/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 20s 129ms/step - accuracy: 0.6591 - loss: 1.6233 - val
_accuracy: 0.2871 - val_loss: 2.6854 - learning_rate: 0.0010
Epoch 2/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 12s 81ms/step - accuracy: 0.7841 - loss: 1.2075 - val_
accuracy: 0.4019 - val_loss: 2.3617 - learning_rate: 0.0010
Epoch 3/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 99ms/step - accuracy: 0.8022 - loss: 1.0842 - val_a
ccuracy: 0.4530 - val_loss: 2.5060 - learning_rate: 0.0010
Epoch 4/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 94ms/step - accuracy: 0.8167 - loss: 0.9574 - val_a
ccuracy: 0.4721 - val_loss: 2.2748 - learning_rate: 0.0010
Epoch 5/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 82ms/step - accuracy: 0.8382 - loss: 0.9240 - val_a
ccuracy: 0.6890 - val_loss: 1.3911 - learning_rate: 0.0010
Epoch 6/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 12s 100ms/step - accuracy: 0.8240 - loss: 0.9358 - val
_accuracy: 0.7687 - val_loss: 0.9811 - learning_rate: 0.0010
Epoch 7/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 96ms/step - accuracy: 0.8616 - loss: 0.8356 - val_a
ccuracy: 0.8278 - val_loss: 0.8526 - learning_rate: 0.0010
Epoch 8/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 80ms/step - accuracy: 0.8783 - loss: 0.7607 - val_a
ccuracy: 0.8373 - val_loss: 0.9362 - learning_rate: 0.0010
Epoch 9/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 80ms/step - accuracy: 0.8569 - loss: 0.7717 - val_
accuracy: 0.7943 - val_loss: 0.8395 - learning_rate: 0.0010
Epoch 10/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 12s 97ms/step - accuracy: 0.8591 - loss: 0.7577 - val_
accuracy: 0.7687 - val_loss: 0.8842 - learning_rate: 0.0010
Epoch 11/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 98ms/step - accuracy: 0.8836 - loss: 0.6901 - val_
accuracy: 0.7257 - val_loss: 1.2509 - learning_rate: 0.0010
Epoch 12/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 78ms/step - accuracy: 0.8667 - loss: 0.6769 - val_a
ccuracy: 0.4242 - val_loss: 2.2943 - learning_rate: 0.0010
Epoch 13/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 101ms/step - accuracy: 0.8785 - loss: 0.6280 - val_
accuracy: 0.8389 - val_loss: 0.7043 - learning_rate: 0.0010
Epoch 14/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 95ms/step - accuracy: 0.8703 - loss: 0.6118 - val_a
ccuracy: 0.8118 - val_loss: 0.7136 - learning_rate: 0.0010
Epoch 15/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 85ms/step - accuracy: 0.8799 - loss: 0.5811 - val_a
ccuracy: 0.7145 - val_loss: 1.1678 - learning_rate: 0.0010
Epoch 16/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 100ms/step - accuracy: 0.8851 - loss: 0.5513 - val
_accuracy: 0.7624 - val_loss: 0.8755 - learning_rate: 0.0010
Epoch 17/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 104ms/step - accuracy: 0.8838 - loss: 0.5497 - val
_accuracy: 0.8070 - val_loss: 0.7161 - learning_rate: 0.0010
Epoch 18/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9018 - loss: 0.4901 - val_a
ccuracy: 0.7097 - val_loss: 0.9437 - learning_rate: 0.0010
Epoch 19/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 100ms/step - accuracy: 0.9053 - loss: 0.4646 - val_
accuracy: 0.7719 - val_loss: 0.7965 - learning_rate: 3.0000e-04
Epoch 20/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 97ms/step - accuracy: 0.9241 - loss: 0.4063 - val_
```

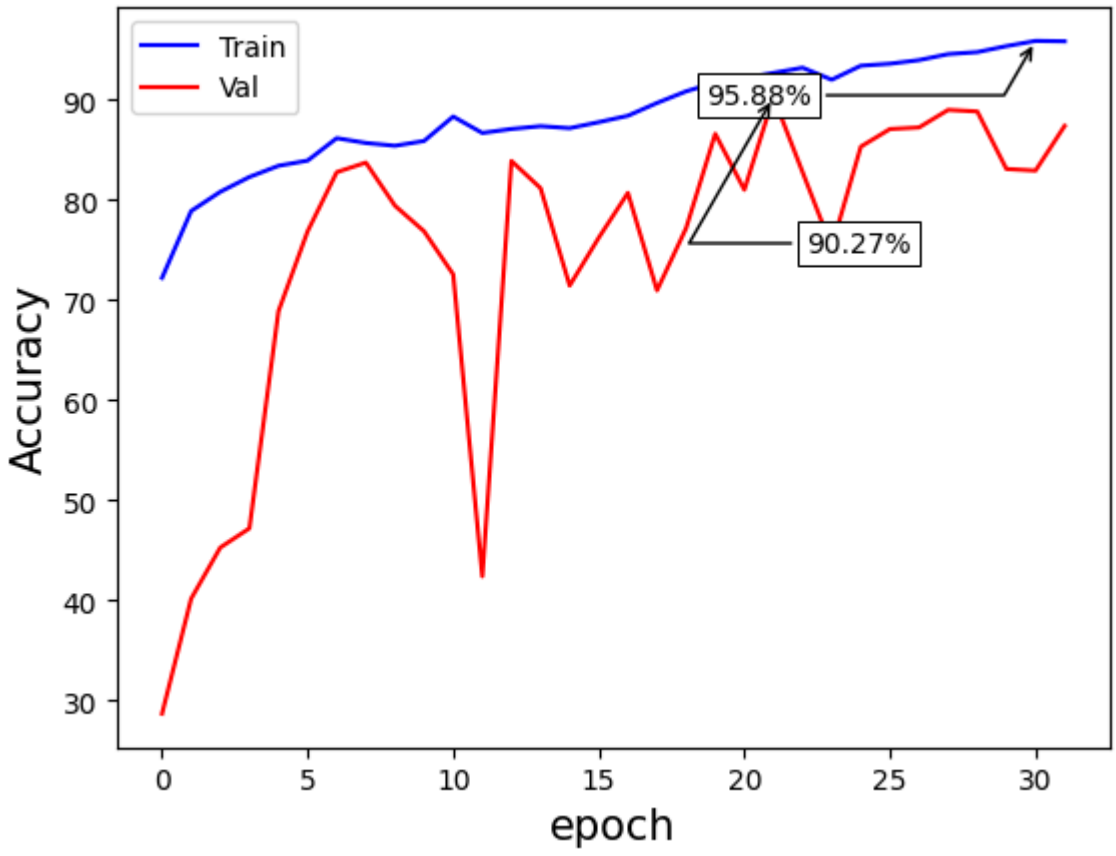
```
accuracy: 0.8660 - val_loss: 0.5157 - learning_rate: 3.0000e-04
Epoch 21/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 83ms/step - accuracy: 0.9291 - loss: 0.3889 - val_a
ccuracy: 0.8102 - val_loss: 0.6698 - learning_rate: 3.0000e-04
Epoch 22/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 101ms/step - accuracy: 0.9221 - loss: 0.3971 - val_
accuracy: 0.9027 - val_loss: 0.4391 - learning_rate: 3.0000e-04
Epoch 23/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 80ms/step - accuracy: 0.9313 - loss: 0.3724 - val_a
ccuracy: 0.8278 - val_loss: 0.6514 - learning_rate: 3.0000e-04
Epoch 24/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 80ms/step - accuracy: 0.9205 - loss: 0.3787 - val_
accuracy: 0.7528 - val_loss: 0.8363 - learning_rate: 3.0000e-04
Epoch 25/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 91ms/step - accuracy: 0.9321 - loss: 0.3690 - val_
accuracy: 0.8533 - val_loss: 0.5563 - learning_rate: 3.0000e-04
Epoch 26/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 9s 103ms/step - accuracy: 0.9354 - loss: 0.3479 - val_
accuracy: 0.8708 - val_loss: 0.4998 - learning_rate: 3.0000e-04
Epoch 27/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 79ms/step - accuracy: 0.9386 - loss: 0.3260 - val_a
ccuracy: 0.8724 - val_loss: 0.5075 - learning_rate: 3.0000e-04
Epoch 28/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 94ms/step - accuracy: 0.9442 - loss: 0.3286 - val_
accuracy: 0.8900 - val_loss: 0.4901 - learning_rate: 9.0000e-05
Epoch 29/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 11s 100ms/step - accuracy: 0.9443 - loss: 0.2921 - val
_accuracy: 0.8884 - val_loss: 0.4681 - learning_rate: 9.0000e-05
Epoch 30/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 7s 80ms/step - accuracy: 0.9624 - loss: 0.2640 - val_a
ccuracy: 0.8309 - val_loss: 0.5919 - learning_rate: 9.0000e-05
Epoch 31/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 8s 100ms/step - accuracy: 0.9628 - loss: 0.2685 - val_
accuracy: 0.8293 - val_loss: 0.6415 - learning_rate: 9.0000e-05
Epoch 32/100
84/84 ━━━━━━━━━━━━━━━━━━━━ 10s 94ms/step - accuracy: 0.9563 - loss: 0.2707 - val_
accuracy: 0.8740 - val_loss: 0.5283 - learning_rate: 9.0000e-05
```

In [ ]:
```
print_accuracy_stats(model, test_ds, class_names, ckpt_path)
plot_confusion_matrix(model, test_ds, class_names, ckpt_path)
```

```
12/12 ━━━━━━━━━━━━━━━━━━━━ 2s 94ms/step

Test Accuracy: 86.32%

12/12 ━━━━━━━━━━━━━━━━━━━━ 1s 66ms/step
```




In [ ]:
```
plot_accuracy(model_fit)
```




# VGG16

In [ ]:
```
def plot_predictions_with_bars(images, true_labels, pred_probs, class_names):
    pred_classes = np.argmax(pred_probs, axis=1)
```

```
    true_classes = np.argmax(true_labels, axis=1)

    # Filter for correctly predicted images
    correct_indices = [i for i, (true, pred) in enumerate(zip(true_classes, pred

    # If there are fewer than 5 correct predictions, plot all of them, otherwise
    num_images_to_plot = min(5, len(correct_indices))
    if num_images_to_plot == 0:
        print("No correctly predicted images in this batch to plot.")
        return

    # Randomly sample indices from the correctly predicted ones
    indices_to_plot = random.sample(correct_indices, num_images_to_plot)

    fig, axs = plt.subplots(num_images_to_plot, 2, figsize=(8, num_images_to_plo
    if num_images_to_plot == 1:
        axs = np.expand_dims(axs, axis=0)

    for i, original_index in enumerate(indices_to_plot):
        img = images[original_index]
        true_idx = true_classes[original_index]
        pred_idx = pred_classes[original_index]
        confidence = pred_probs[original_index][pred_idx] * 100
        pred_label = class_names[pred_idx]
        true_label = class_names[true_idx]
        color = 'blue' # Since we are only plotting correct predictions

        axs[i, 0].imshow(img)
        axs[i, 0].axis('off')
        axs[i, 0].set_title(f"{pred_label} {confidence:.0f}% ({true_label})", co

        axs[i, 1].barh(range(len(class_names)), pred_probs[original_index],
                       color=['blue' if j == pred_idx else 'gray' for j in range
        axs[i, 1].set_xlim([0, 1])
        axs[i, 1].set_yticks(range(len(class_names)))
        axs[i, 1].set_yticklabels(class_names, rotation=45)
        axs[i, 1].invert_yaxis()

    plt.subplots_adjust(hspace=0.5, wspace=0.4)
    plt.show()
```

In [ ]:
```
def plot_training_history(history):
    # Debug print to inspect keys
    print("History keys:", history.history.keys())

    acc = history.history.get('accuracy', [])
    val_acc = history.history.get('val_accuracy', [])
    loss = history.history.get('loss', [])
    val_loss = history.history.get('val_loss', [])
    epochs = list(range(len(acc)))

    plt.figure(figsize=(14, 5))

    # Accuracy Plot
    plt.subplot(1, 2, 1)
    plt.plot(epochs, acc, 'b-', label='Training Accuracy')
    plt.plot(epochs, val_acc, 'r-', label='Validation Accuracy')
    plt.title('Training vs Validation Accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
```

```
    plt.xticks(epochs)
    plt.legend()

    # Loss Plot
    plt.subplot(1, 2, 2)
    plt.plot(epochs, loss, 'b-', label='Training Loss')
    plt.plot(epochs, val_loss, 'r-', label='Validation Loss')
    plt.title('Training vs Validation Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.xticks(epochs)
    plt.legend()

    plt.tight_layout()
    plt.show()
```

In [ ]:
```
def plot_confusion_matrix(model, data_generator, class_names):
    y_true = data_generator.classes
    y_pred_probs = model.predict(data_generator)
    y_pred = np.argmax(y_pred_probs, axis=1)

    cm = confusion_matrix(y_true, y_pred)

    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=class_names,
                yticklabels=class_names)
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.show()

    print("\nClassification Report:\n")
    print(classification_report(y_true, y_pred, target_names=class_names))
```

## Before fine-tuning

In [ ]:
```
# 1. Define directories
train_dir = "/content/ninjacart_data/train"
val_dir = "/content/ninjacart_data/validation"
test_dir = "/content/ninjacart_data/test"

# 2. Image Data Generators
train_datagen = ImageDataGenerator(rescale=1./255, horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir, target_size=(224, 224), batch_size=32, class_mode='categorical'
)

val_generator = val_datagen.flow_from_directory(
    val_dir, target_size=(224, 224), batch_size=32, class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir, target_size=(224, 224), batch_size=32, class_mode='categorical', s
)
```

```python
# 3. Load base VGG16 model
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224,

# Freeze all layers first
for layer in base_model.layers:
    layer.trainable = False

# 4. Add custom classification head
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dense(512, activation='relu')(x)
x = tf.keras.layers.Dropout(0.5)(x)
predictions = tf.keras.layers.Dense(train_generator.num_classes, activation='sof

model = Model(inputs=base_model.input, outputs=predictions)

# 5. Compile the model
model.compile(optimizer=Adam(learning_rate=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# 6. Train head only
history = model.fit(train_generator, validation_data=val_generator, epochs=10)
```

```
Found 2502 images belonging to 4 classes.
Found 627 images belonging to 4 classes.
Found 351 images belonging to 4 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-application
s/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_datas
et_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__ini
t__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multipr
ocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 43s 390ms/step - accuracy: 0.3548 - loss: 1.3781 - val
_accuracy: 0.7767 - val_loss: 0.9356
Epoch 2/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 219ms/step - accuracy: 0.6406 - loss: 0.9452 - val
_accuracy: 0.8182 - val_loss: 0.7171
Epoch 3/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 22s 238ms/step - accuracy: 0.7335 - loss: 0.7616 - val
_accuracy: 0.8246 - val_loss: 0.5903
Epoch 4/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 19s 242ms/step - accuracy: 0.7947 - loss: 0.6030 - val
_accuracy: 0.8517 - val_loss: 0.5124
Epoch 5/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 215ms/step - accuracy: 0.8182 - loss: 0.5376 - val
_accuracy: 0.8581 - val_loss: 0.4674
Epoch 6/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 18s 226ms/step - accuracy: 0.8215 - loss: 0.4927 - val
_accuracy: 0.8581 - val_loss: 0.4282
Epoch 7/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 18s 220ms/step - accuracy: 0.8407 - loss: 0.4504 - val
_accuracy: 0.8628 - val_loss: 0.4037
Epoch 8/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 18s 223ms/step - accuracy: 0.8666 - loss: 0.4109 - val
_accuracy: 0.8756 - val_loss: 0.3832
Epoch 9/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 18s 224ms/step - accuracy: 0.8739 - loss: 0.3872 - val
_accuracy: 0.8852 - val_loss: 0.3657
Epoch 10/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 217ms/step - accuracy: 0.8672 - loss: 0.3582 - val
_accuracy: 0.8852 - val_loss: 0.3490
```

In [ ]:
```
model.summary()
```

```
Model: "functional_10"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_10 (InputLayer) | (None, 224, 224, 3) | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1,792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36,928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73,856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147,584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295,168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1,180,160 |
| block4_conv2 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| block4_conv3 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| block4_pool (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv2 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv3 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| global_average_pooling2d_6 (GlobalAveragePooling2D) | (None, 512) | 0 |
| dense_14 (Dense) | (None, 512) | 262,656 |
| dropout_5 (Dropout) | (None, 512) | 0 |
| dense_15 (Dense) | (None, 4) | 2,052 |

**Total params:** 15,508,814 (59.16 MB)

**Trainable params:** 264,708 (1.01 MB)

**Non-trainable params:** 14,714,688 (56.13 MB)

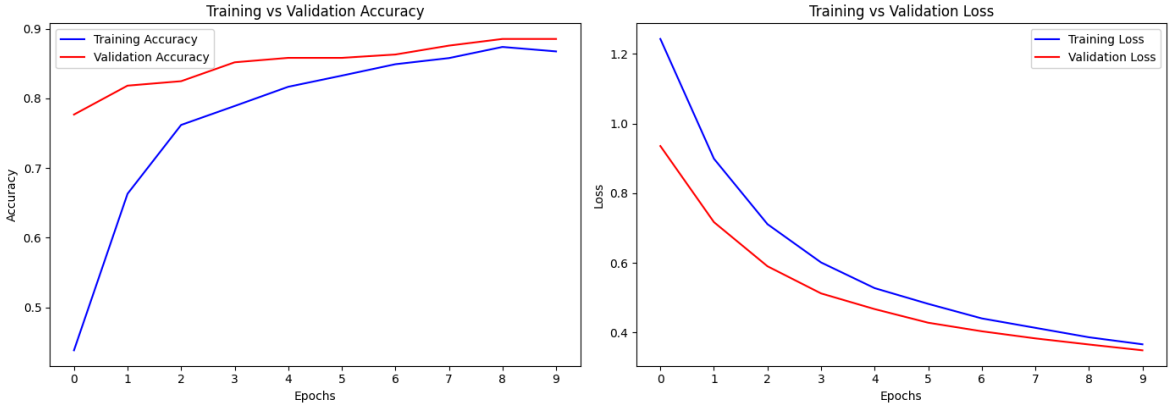
**Optimizer params:** 529,418 (2.02 MB)

In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model8_VGG_GAP_DP_Fine
model.save("/content/ninjacart_models/model8_VGG_GAP_DP_FineTune/model8_VGG_GAP_
```

```
Created directory: /content/ninjacart_models/model8_VGG_GAP_DP_FineTune
```

In [ ]:
```
plot_training_history(history)
```

```
History keys: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

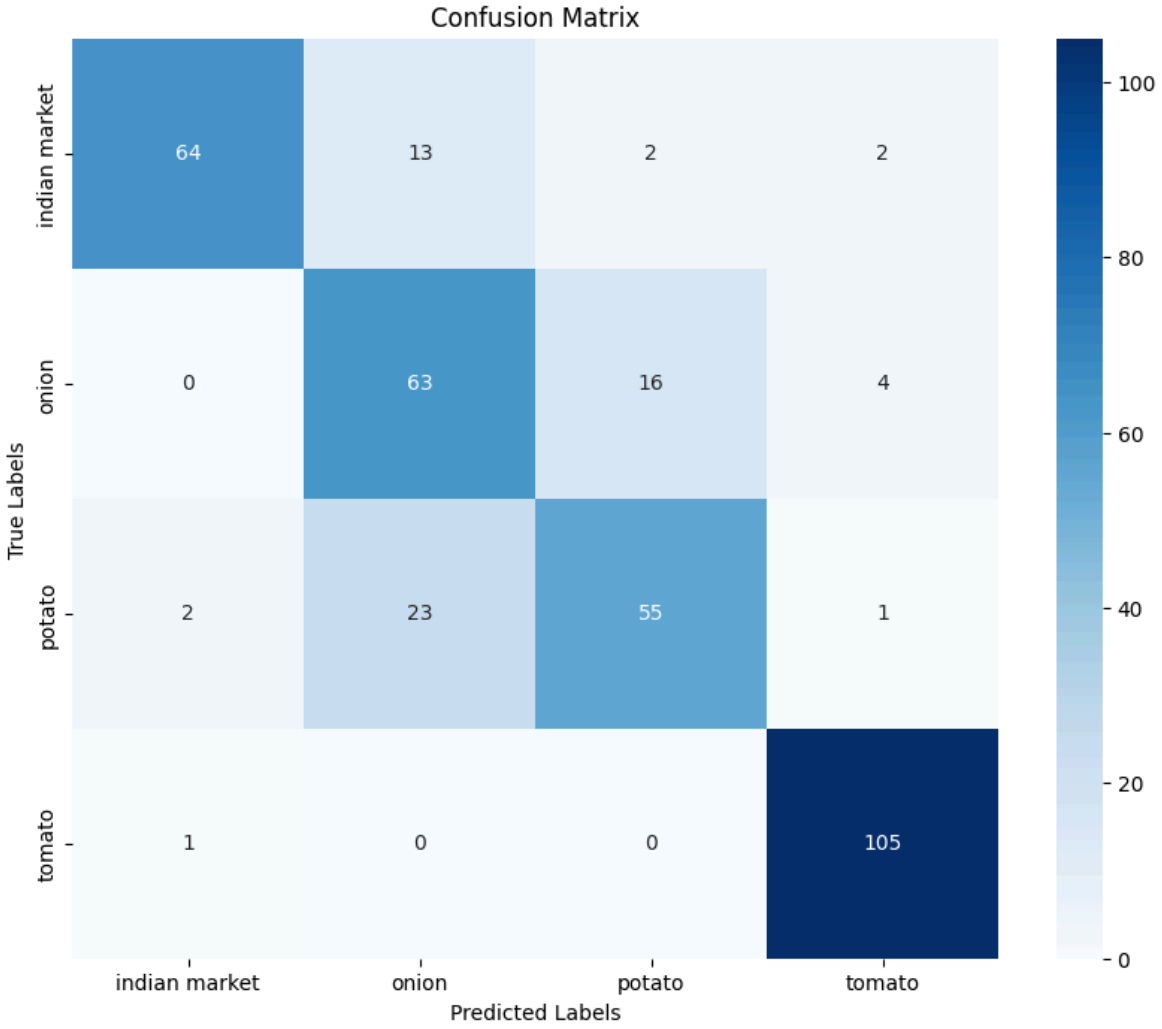



In [ ]:
```
inception_model_finetune = tf.keras.models.load_model("/content/ninjacart_models
loss, acc = inception_model_finetune.evaluate(test_generator, verbose=2)
print('Before FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
11/11 - 13s - 1s/step - accuracy: 0.8177 - loss: 0.4606
Before FineTuning
Restored model, accuracy: 81.77%
```

In [ ]:
```
class_names = list(test_generator.class_indices.keys())
plot_confusion_matrix(model, test_generator, class_names)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 4s 342ms/step
```




Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 0.96 | 0.79 | 0.86 | 81 |
| onion | 0.64 | 0.76 | 0.69 | 83 |
| potato | 0.75 | 0.68 | 0.71 | 81 |
| tomato | 0.94 | 0.99 | 0.96 | 106 |
| accuracy | | | 0.82 | 351 |
| macro avg | 0.82 | 0.80 | 0.81 | 351 |
| weighted avg | 0.83 | 0.82 | 0.82 | 351 |

# After fine-tuning

In [ ]:
```
# 7. Unfreeze last few layers of base model for fine-tuning
for layer in base_model.layers[-4:]:
    layer.trainable = True

# 8. Recompile with lower LR
model.compile(optimizer=Adam(learning_rate=1e-5),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# 9. Continue training (fine-tune)
history_finetune = model.fit(train_generator, validation_data=val_generator, epo
```

```
Epoch 1/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 28s 293ms/step - accuracy: 0.8762 - loss: 0.3222 - val
_accuracy: 0.9059 - val_loss: 0.2376
Epoch 2/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 19s 239ms/step - accuracy: 0.9167 - loss: 0.2213 - val
_accuracy: 0.9266 - val_loss: 0.1941
Epoch 3/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 19s 239ms/step - accuracy: 0.9344 - loss: 0.1673 - val
_accuracy: 0.9250 - val_loss: 0.1914
Epoch 4/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 20s 237ms/step - accuracy: 0.9582 - loss: 0.1365 - val
_accuracy: 0.9458 - val_loss: 0.1654
Epoch 5/5
79/79 ━━━━━━━━━━━━━━━━━━━━ 21s 270ms/step - accuracy: 0.9658 - loss: 0.0977 - val
_accuracy: 0.9458 - val_loss: 0.1527
```

In [ ]:
```
model.summary()
```

```
Model: "functional_10"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_10 (InputLayer) | (None, 224, 224, 3) | 0 |
| block1_conv1 (Conv2D) | (None, 224, 224, 64) | 1,792 |
| block1_conv2 (Conv2D) | (None, 224, 224, 64) | 36,928 |
| block1_pool (MaxPooling2D) | (None, 112, 112, 64) | 0 |
| block2_conv1 (Conv2D) | (None, 112, 112, 128) | 73,856 |
| block2_conv2 (Conv2D) | (None, 112, 112, 128) | 147,584 |
| block2_pool (MaxPooling2D) | (None, 56, 56, 128) | 0 |
| block3_conv1 (Conv2D) | (None, 56, 56, 256) | 295,168 |
| block3_conv2 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| block3_conv3 (Conv2D) | (None, 56, 56, 256) | 590,080 |
| block3_pool (MaxPooling2D) | (None, 28, 28, 256) | 0 |
| block4_conv1 (Conv2D) | (None, 28, 28, 512) | 1,180,160 |
| block4_conv2 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| block4_conv3 (Conv2D) | (None, 28, 28, 512) | 2,359,808 |
| block4_pool (MaxPooling2D) | (None, 14, 14, 512) | 0 |
| block5_conv1 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv2 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_conv3 (Conv2D) | (None, 14, 14, 512) | 2,359,808 |
| block5_pool (MaxPooling2D) | (None, 7, 7, 512) | 0 |
| global_average_pooling2d_6 (GlobalAveragePooling2D) | (None, 512) | 0 |
| dense_14 (Dense) | (None, 512) | 262,656 |
| dropout_5 (Dropout) | (None, 512) | 0 |
| dense_15 (Dense) | (None, 4) | 2,052 |

**Total params:** 29,667,662 (113.17 MB)

**Trainable params:** 7,344,132 (28.02 MB)

**Non-trainable params:** 7,635,264 (29.13 MB)

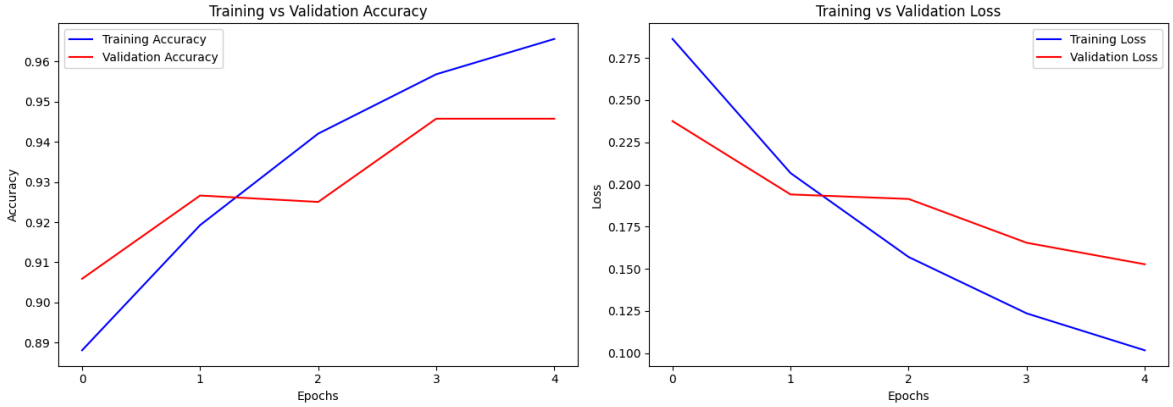
**Optimizer params:** 14,688,266 (56.03 MB)

In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model8_VGG_GAP_DP_Fine
model.save("/content/ninjacart_models/model8_VGG_GAP_DP_FineTune/model8_VGG_GAP_
```

```
Directory already exists: /content/ninjacart_models/model8_VGG_GAP_DP_FineTune
```

In [ ]:
```
plot_training_history(history_finetune)
```

```
History keys: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```

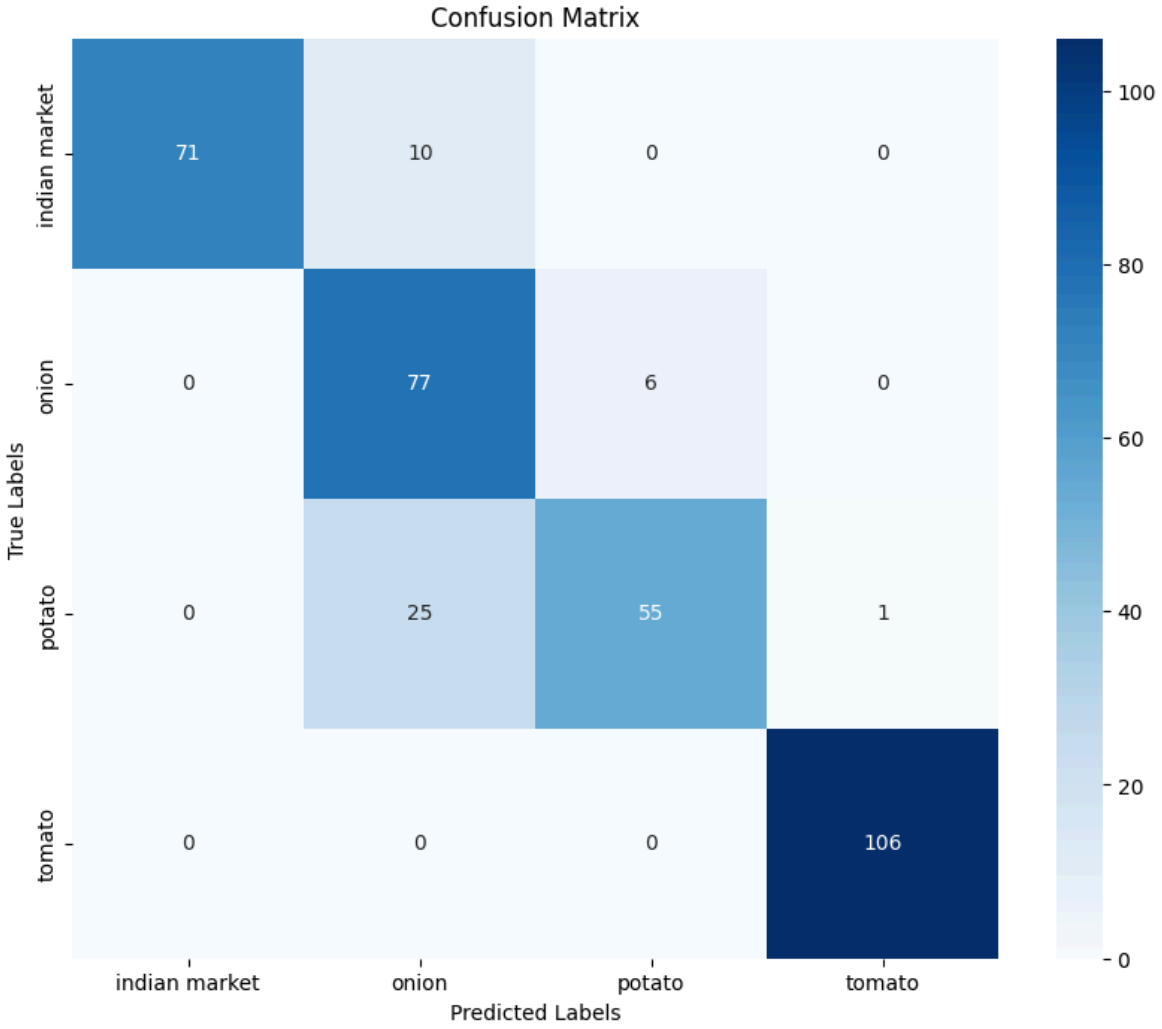



In [ ]:
```
inception_model_finetune = tf.keras.models.load_model("/content/ninjacart_models
loss, acc = inception_model_finetune.evaluate(test_generator, verbose=2)
print('After FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
11/11 - 5s - 479ms/step - accuracy: 0.8803 - loss: 0.3373
After FineTuning
Restored model, accuracy: 88.03%
```

In [ ]:
```
class_names = list(test_generator.class_indices.keys())
plot_confusion_matrix(model, test_generator, class_names)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 5s 389ms/step
```



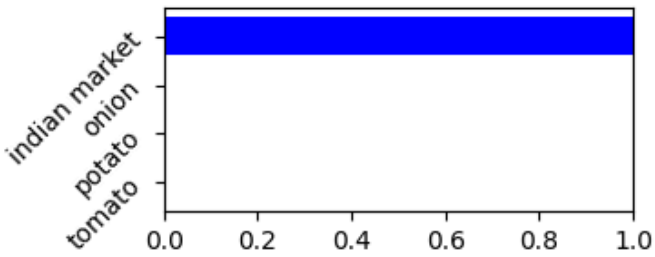

Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 1.00 | 0.88 | 0.93 | 81 |
| onion | 0.69 | 0.93 | 0.79 | 83 |
| potato | 0.90 | 0.68 | 0.77 | 81 |
| tomato | 0.99 | 1.00 | 1.00 | 106 |
| accuracy | | | 0.88 | 351 |
| macro avg | 0.89 | 0.87 | 0.87 | 351 |
| weighted avg | 0.90 | 0.88 | 0.88 | 351 |

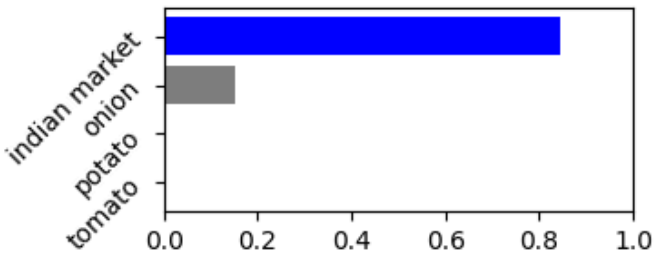
In [ ]:
```
test_generator.reset()
images, labels = next(test_generator)
pred_probs = model.predict(images)
plot_predictions_with_bars(images[:5], labels[:5], pred_probs[:5], class_names)
```

**1/1** ━━━━━━━━━━━━━━━━━━━━ **1s** 1s/step

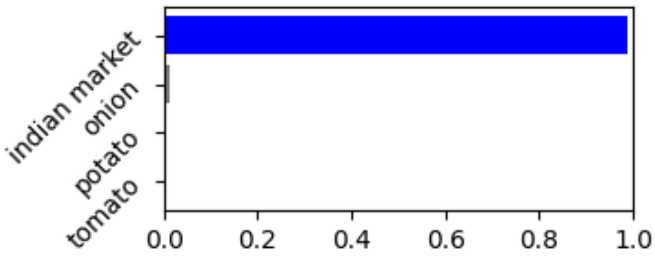
indian market 100% (indian market)



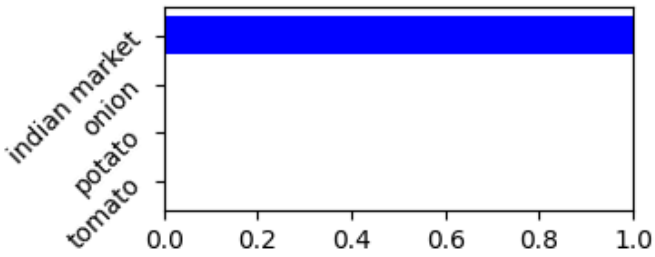

indian market 85% (indian market)




indian market 99% (indian market)




indian market 100% (indian market)




# InceptionV3 model

In [ ]:

```
datagen = ImageDataGenerator(preprocessing_function=inception_v3.preprocess_inpu

train_data = datagen.flow_from_directory(
    '/content/ninjacart_data/train',
    target_size=(299, 299),
    batch_size=32,
    class_mode='categorical'
)

val_data = datagen.flow_from_directory(
    '/content/ninjacart_data/validation',
    target_size=(299, 299),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

test_data = datagen.flow_from_directory(
```

```
    '/content/ninjacart_data/test',
    target_size=(299, 299),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 2502 images belonging to 4 classes.
Found 627 images belonging to 4 classes.
Found 351 images belonging to 4 classes.
```

# Without FineTuning

In [ ]:

```
num_classes = 4
# Load base InceptionV3 model without top classification layer
base_model = InceptionV3(weights='imagenet', include_top=False, input_shape=(299

# Freeze all layers first
for layer in base_model.layers:
    layer.trainable = False

# Add custom top layers
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.5)(x)                    # Optional dropout
x = tf.keras.layers.Dense(1024, activation='relu')(x)  # Fully connected layer
predictions = tf.keras.layers.Dense(num_classes, activation='softmax')(x)

# Final model
inception_model = Model(inputs=base_model.input, outputs=predictions)

# Compile the model
inception_model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_c

# Train only the top layers
before_ft_inc_hist = inception_model.fit(train_data, epochs=10, validation_data=
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-application
s/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_datas
et_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__ini
t__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multipr
ocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 72s 542ms/step - accuracy: 0.8215 - loss: 0.5547 - val
_accuracy: 0.9458 - val_loss: 0.1409
Epoch 2/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 22s 274ms/step - accuracy: 0.9487 - loss: 0.1236 - val
_accuracy: 0.9681 - val_loss: 0.0904
Epoch 3/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 41s 271ms/step - accuracy: 0.9622 - loss: 0.0980 - val
_accuracy: 0.9585 - val_loss: 0.1406
Epoch 4/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 20s 259ms/step - accuracy: 0.9701 - loss: 0.0806 - val
_accuracy: 0.9793 - val_loss: 0.0686
Epoch 5/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 21s 261ms/step - accuracy: 0.9763 - loss: 0.0622 - val
_accuracy: 0.9649 - val_loss: 0.1100
Epoch 6/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 20s 259ms/step - accuracy: 0.9663 - loss: 0.1068 - val
_accuracy: 0.9793 - val_loss: 0.0650
Epoch 7/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 21s 269ms/step - accuracy: 0.9752 - loss: 0.0593 - val
_accuracy: 0.9793 - val_loss: 0.0754
Epoch 8/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 20s 257ms/step - accuracy: 0.9726 - loss: 0.0816 - val
_accuracy: 0.9681 - val_loss: 0.0930
Epoch 9/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 22s 278ms/step - accuracy: 0.9736 - loss: 0.0870 - val
_accuracy: 0.9745 - val_loss: 0.0675
Epoch 10/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 21s 260ms/step - accuracy: 0.9813 - loss: 0.0550 - val
_accuracy: 0.9713 - val_loss: 0.0905
```

In [ ]:
```
inception_model.summary()
```

```
Model: "functional_11"
```

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_layer_11 (InputLayer) | (None, 299, 299, 3) | 0 | - |
| conv2d_31 (Conv2D) | (None, 149, 149, 32) | 864 | input_layer_11[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 149, 149, 32) | 96 | conv2d_31[0][0] |
| activation_30 (Activation) | (None, 149, 149, 32) | 0 | batch_normalizat… |
| conv2d_32 (Conv2D) | (None, 147, 147, 32) | 9,216 | activation_30[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 147, 147, 32) | 96 | conv2d_32[0][0] |
| activation_31 (Activation) | (None, 147, 147, 32) | 0 | batch_normalizat… |
| conv2d_33 (Conv2D) | (None, 147, 147, 64) | 18,432 | activation_31[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 147, 147, 64) | 192 | conv2d_33[0][0] |
| activation_32 (Activation) | (None, 147, 147, 64) | 0 | batch_normalizat… |
| max_pooling2d_25 (MaxPooling2D) | (None, 73, 73, 64) | 0 | activation_32[0]… |
| conv2d_34 (Conv2D) | (None, 73, 73, 80) | 5,120 | max_pooling2d_25… |
| batch_normalizatio… (BatchNormalizatio… | (None, 73, 73, 80) | 240 | conv2d_34[0][0] |
| activation_33 (Activation) | (None, 73, 73, 80) | 0 | batch_normalizat… |
| conv2d_35 (Conv2D) | (None, 71, 71, 192) | 138,240 | activation_33[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 71, 71, 192) | 576 | conv2d_35[0][0] |
| activation_34 (Activation) | (None, 71, 71, 192) | 0 | batch_normalizat… |
| max_pooling2d_26 (MaxPooling2D) | (None, 35, 35, 192) | 0 | activation_34[0]… |
| conv2d_39 (Conv2D) | (None, 35, 35, 64) | 12,288 | max_pooling2d_26… |

| | | | |
|---|---|---|---|
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_39[0][0] |
| activation_38 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| conv2d_37 (Conv2D) | (None, 35, 35, 48) | 9,216 | max_pooling2d_26… |
| conv2d_40 (Conv2D) | (None, 35, 35, 96) | 55,296 | activation_38[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 48) | 144 | conv2d_37[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_40[0][0] |
| activation_36 (Activation) | (None, 35, 35, 48) | 0 | batch_normalizat… |
| activation_39 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| average_pooling2d (AveragePooling2D) | (None, 35, 35, 192) | 0 | max_pooling2d_26… |
| conv2d_36 (Conv2D) | (None, 35, 35, 64) | 12,288 | max_pooling2d_26… |
| conv2d_38 (Conv2D) | (None, 35, 35, 64) | 76,800 | activation_36[0]… |
| conv2d_41 (Conv2D) | (None, 35, 35, 96) | 82,944 | activation_39[0]… |
| conv2d_42 (Conv2D) | (None, 35, 35, 32) | 6,144 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_36[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_38[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_41[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 32) | 96 | conv2d_42[0][0] |
| activation_35 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_37 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_40 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| activation_41 (Activation) | (None, 35, 35, 32) | 0 | batch_normalizat… |
| mixed0 (Concatenate) | (None, 35, 35, 256) | 0 | activation_35[0]…<br>activation_37[0]…<br>activation_40[0]…<br>activation_41[0]… |
| conv2d_46 (Conv2D) | (None, 35, 35, 64) | 16,384 | mixed0[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_46[0][0] |
| activation_45 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| conv2d_44 (Conv2D) | (None, 35, 35, 48) | 12,288 | mixed0[0][0] |
| conv2d_47 (Conv2D) | (None, 35, 35, 96) | 55,296 | activation_45[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 48) | 144 | conv2d_44[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_47[0][0] |
| activation_43 (Activation) | (None, 35, 35, 48) | 0 | batch_normalizat… |
| activation_46 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| average_pooling2d_1 (AveragePooling2D) | (None, 35, 35, 256) | 0 | mixed0[0][0] |
| conv2d_43 (Conv2D) | (None, 35, 35, 64) | 16,384 | mixed0[0][0] |
| conv2d_45 (Conv2D) | (None, 35, 35, 64) | 76,800 | activation_43[0]… |
| conv2d_48 (Conv2D) | (None, 35, 35, 96) | 82,944 | activation_46[0]… |
| conv2d_49 (Conv2D) | (None, 35, 35, 64) | 16,384 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_43[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_45[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_48[0][0] |
| batch_normalizatio… | (None, 35, 35, | 192 | conv2d_49[0][0] |

| | | | |
|---|---|---|---|
| (BatchNormalizatio… | 64) | | |
| activation_42 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_44 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_47 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| activation_48 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| mixed1 (Concatenate) | (None, 35, 35, 288) | 0 | activation_42[0]…<br>activation_44[0]…<br>activation_47[0]…<br>activation_48[0]… |
| conv2d_53 (Conv2D) | (None, 35, 35, 64) | 18,432 | mixed1[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_53[0][0] |
| activation_52 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| conv2d_51 (Conv2D) | (None, 35, 35, 48) | 13,824 | mixed1[0][0] |
| conv2d_54 (Conv2D) | (None, 35, 35, 96) | 55,296 | activation_52[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 48) | 144 | conv2d_51[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_54[0][0] |
| activation_50 (Activation) | (None, 35, 35, 48) | 0 | batch_normalizat… |
| activation_53 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| average_pooling2d_2 (AveragePooling2D) | (None, 35, 35, 288) | 0 | mixed1[0][0] |
| conv2d_50 (Conv2D) | (None, 35, 35, 64) | 18,432 | mixed1[0][0] |
| conv2d_52 (Conv2D) | (None, 35, 35, 64) | 76,800 | activation_50[0]… |
| conv2d_55 (Conv2D) | (None, 35, 35, 96) | 82,944 | activation_53[0]… |
| conv2d_56 (Conv2D) | (None, 35, 35, 64) | 18,432 | average_pooling2… |

| | | | |
|---|---|---|---|
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_50[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_52[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_55[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_56[0][0] |
| activation_49 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_51 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_54 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| activation_55 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| mixed2 (Concatenate) | (None, 35, 35, 288) | 0 | activation_49[0]… activation_51[0]… activation_54[0]… activation_55[0]… |
| conv2d_58 (Conv2D) | (None, 35, 35, 64) | 18,432 | mixed2[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_58[0][0] |
| activation_57 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| conv2d_59 (Conv2D) | (None, 35, 35, 96) | 55,296 | activation_57[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_59[0][0] |
| activation_58 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| conv2d_57 (Conv2D) | (None, 17, 17, 384) | 995,328 | mixed2[0][0] |
| conv2d_60 (Conv2D) | (None, 17, 17, 96) | 82,944 | activation_58[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 384) | 1,152 | conv2d_57[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 96) | 288 | conv2d_60[0][0] |

| | | | |
|---|---|---|---|
| activation_56 (Activation) | (None, 17, 17, 384) | 0 | batch_normalizat… |
| activation_59 (Activation) | (None, 17, 17, 96) | 0 | batch_normalizat… |
| max_pooling2d_27 (MaxPooling2D) | (None, 17, 17, 288) | 0 | mixed2[0][0] |
| mixed3 (Concatenate) | (None, 17, 17, 768) | 0 | activation_56[0]…<br>activation_59[0]…<br>max_pooling2d_27… |
| conv2d_65 (Conv2D) | (None, 17, 17, 128) | 98,304 | mixed3[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_65[0][0] |
| activation_64 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| conv2d_66 (Conv2D) | (None, 17, 17, 128) | 114,688 | activation_64[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_66[0][0] |
| activation_65 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| conv2d_62 (Conv2D) | (None, 17, 17, 128) | 98,304 | mixed3[0][0] |
| conv2d_67 (Conv2D) | (None, 17, 17, 128) | 114,688 | activation_65[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_62[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_67[0][0] |
| activation_61 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| activation_66 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| conv2d_63 (Conv2D) | (None, 17, 17, 128) | 114,688 | activation_61[0]… |
| conv2d_68 (Conv2D) | (None, 17, 17, 128) | 114,688 | activation_66[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_63[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_68[0][0] |

| | | | |
|---|---|---|---|
| activation_62 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| activation_67 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| average_pooling2d_3 (AveragePooling2D) | (None, 17, 17, 768) | 0 | mixed3[0][0] |
| conv2d_61 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed3[0][0] |
| conv2d_64 (Conv2D) | (None, 17, 17, 192) | 172,032 | activation_62[0]… |
| conv2d_69 (Conv2D) | (None, 17, 17, 192) | 172,032 | activation_67[0]… |
| conv2d_70 (Conv2D) | (None, 17, 17, 192) | 147,456 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_61[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_64[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_69[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_70[0][0] |
| activation_60 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_63 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_68 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_69 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| mixed4 (Concatenate) | (None, 17, 17, 768) | 0 | activation_60[0]… activation_63[0]… activation_68[0]… activation_69[0]… |
| conv2d_75 (Conv2D) | (None, 17, 17, 160) | 122,880 | mixed4[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_75[0][0] |
| activation_74 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| conv2d_76 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_74[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_76[0][0] |
| activation_75 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_72 (Conv2D) | (None, 17, 17, 160) | 122,880 | mixed4[0][0] |
| conv2d_77 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_75[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_72[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_77[0][0] |
| activation_71 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| activation_76 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_73 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_71[0]… |
| conv2d_78 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_76[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_73[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_78[0][0] |
| activation_72 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| activation_77 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| average_pooling2d_4 (AveragePooling2D) | (None, 17, 17, 768) | 0 | mixed4[0][0] |
| conv2d_71 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed4[0][0] |
| conv2d_74 (Conv2D) | (None, 17, 17, 192) | 215,040 | activation_72[0]… |
| conv2d_79 (Conv2D) | (None, 17, 17, 192) | 215,040 | activation_77[0]… |
| conv2d_80 (Conv2D) | (None, 17, 17, 192) | 147,456 | average_pooling2… |

| | | | |
|---|---|---|---|
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_71[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_74[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_79[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_80[0][0] |
| activation_70 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_73 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_78 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_79 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| mixed5 (Concatenate) | (None, 17, 17, 768) | 0 | activation_70[0]… activation_73[0]… activation_78[0]… activation_79[0]… |
| conv2d_85 (Conv2D) | (None, 17, 17, 160) | 122,880 | mixed5[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_85[0][0] |
| activation_84 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_86 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_84[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_86[0][0] |
| activation_85 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_82 (Conv2D) | (None, 17, 17, 160) | 122,880 | mixed5[0][0] |
| conv2d_87 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_85[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_82[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_87[0][0] |
| activation_81 | (None, 17, 17, | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| (Activation) | 160) | | |
| activation_86 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_83 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_81[0]… |
| conv2d_88 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_86[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_83[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_88[0][0] |
| activation_82 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| activation_87 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| average_pooling2d_5 (AveragePooling2D) | (None, 17, 17, 768) | 0 | mixed5[0][0] |
| conv2d_81 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed5[0][0] |
| conv2d_84 (Conv2D) | (None, 17, 17, 192) | 215,040 | activation_82[0]… |
| conv2d_89 (Conv2D) | (None, 17, 17, 192) | 215,040 | activation_87[0]… |
| conv2d_90 (Conv2D) | (None, 17, 17, 192) | 147,456 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_81[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_84[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_89[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_90[0][0] |
| activation_80 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_83 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_88 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_89 | (None, 17, 17, | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| (Activation) | 192) | | |
| mixed6 (Concatenate) | (None, 17, 17, 768) | 0 | activation_80[0]…<br>activation_83[0]…<br>activation_88[0]…<br>activation_89[0]… |
| conv2d_95 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed6[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_95[0][0] |
| activation_94 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_96 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_94[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_96[0][0] |
| activation_95 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_92 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed6[0][0] |
| conv2d_97 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_95[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_92[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_97[0][0] |
| activation_91 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_96 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_93 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_91[0]… |
| conv2d_98 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_96[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_93[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_98[0][0] |
| activation_92 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_97 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| average_pooling2d_6 (AveragePooling2D) | (None, 17, 17, 768) | 0 | mixed6[0][0] |
| conv2d_91 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed6[0][0] |
| conv2d_94 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_92[0]… |
| conv2d_99 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_97[0]… |
| conv2d_100 (Conv2D) | (None, 17, 17, 192) | 147,456 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_91[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_94[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_99[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_100[0][0] |
| activation_90 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_93 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_98 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_99 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| mixed7 (Concatenate) | (None, 17, 17, 768) | 0 | activation_90[0]… activation_93[0]… activation_98[0]… activation_99[0]… |
| conv2d_103 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed7[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_103[0][0] |
| activation_102 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_104 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_102[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_104[0][0] |

| | | | |
|---|---|---|---|
| activation_103 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_101 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed7[0][0] |
| conv2d_105 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_103[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_101[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_105[0][0] |
| activation_100 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_104 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_102 (Conv2D) | (None, 8, 8, 320) | 552,960 | activation_100[0… |
| conv2d_106 (Conv2D) | (None, 8, 8, 192) | 331,776 | activation_104[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 320) | 960 | conv2d_102[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 192) | 576 | conv2d_106[0][0] |
| activation_101 (Activation) | (None, 8, 8, 320) | 0 | batch_normalizat… |
| activation_105 (Activation) | (None, 8, 8, 192) | 0 | batch_normalizat… |
| max_pooling2d_28 (MaxPooling2D) | (None, 8, 8, 768) | 0 | mixed7[0][0] |
| mixed8 (Concatenate) | (None, 8, 8, 1280) | 0 | activation_101[0… activation_105[0… max_pooling2d_28… |
| conv2d_111 (Conv2D) | (None, 8, 8, 448) | 573,440 | mixed8[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 448) | 1,344 | conv2d_111[0][0] |
| activation_110 (Activation) | (None, 8, 8, 448) | 0 | batch_normalizat… |
| conv2d_108 (Conv2D) | (None, 8, 8, 384) | 491,520 | mixed8[0][0] |
| conv2d_112 (Conv2D) | (None, 8, 8, 384) | 1,548,288 | activation_110[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_108[0][0] |
| batch_normalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_112[0][0] |

| | | | |
|---|---|---|---|
| (BatchNormalizatio… | | | |
| activation_107 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_111 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| conv2d_109 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_107[0… |
| conv2d_110 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_107[0… |
| conv2d_113 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_111[0… |
| conv2d_114 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_111[0… |
| average_pooling2d_7 (AveragePooling2D) | (None, 8, 8, 1280) | 0 | mixed8[0][0] |
| conv2d_107 (Conv2D) | (None, 8, 8, 320) | 409,600 | mixed8[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_109[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_110[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_113[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_114[0][0] |
| conv2d_115 (Conv2D) | (None, 8, 8, 192) | 245,760 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 320) | 960 | conv2d_107[0][0] |
| activation_108 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_109 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_112 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_113 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 192) | 576 | conv2d_115[0][0] |
| activation_106 (Activation) | (None, 8, 8, 320) | 0 | batch_normalizat… |
| mixed9_0 (Concatenate) | (None, 8, 8, 768) | 0 | activation_108[0… activation_109[0… |
| concatenate | (None, 8, 8, 768) | 0 | activation_112[0… |

| | | | |
|---|---|---|---|
| (Concatenate) | | | activation_113[0… |
| activation_114 (Activation) | (None, 8, 8, 192) | 0 | batch_normalizat… |
| mixed9 (Concatenate) | (None, 8, 8, 2048) | 0 | activation_106[0… mixed9_0[0][0], concatenate[0][0… activation_114[0… |
| conv2d_120 (Conv2D) | (None, 8, 8, 448) | 917,504 | mixed9[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 448) | 1,344 | conv2d_120[0][0] |
| activation_119 (Activation) | (None, 8, 8, 448) | 0 | batch_normalizat… |
| conv2d_117 (Conv2D) | (None, 8, 8, 384) | 786,432 | mixed9[0][0] |
| conv2d_121 (Conv2D) | (None, 8, 8, 384) | 1,548,288 | activation_119[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_117[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_121[0][0] |
| activation_116 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_120 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| conv2d_118 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_116[0… |
| conv2d_119 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_116[0… |
| conv2d_122 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_120[0… |
| conv2d_123 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_120[0… |
| average_pooling2d_8 (AveragePooling2D) | (None, 8, 8, 2048) | 0 | mixed9[0][0] |
| conv2d_116 (Conv2D) | (None, 8, 8, 320) | 655,360 | mixed9[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_118[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_119[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_122[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_123[0][0] |
| conv2d_124 (Conv2D) | (None, 8, 8, 192) | 393,216 | average_pooling2… |

| | | | |
|---|---|---|---|
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 320) | 960 | conv2d_116[0][0] |
| activation_117 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_118 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_121 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_122 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 192) | 576 | conv2d_124[0][0] |
| activation_115 (Activation) | (None, 8, 8, 320) | 0 | batch_normalizat… |
| mixed9_1 (Concatenate) | (None, 8, 8, 768) | 0 | activation_117[0… activation_118[0… |
| concatenate_1 (Concatenate) | (None, 8, 8, 768) | 0 | activation_121[0… activation_122[0… |
| activation_123 (Activation) | (None, 8, 8, 192) | 0 | batch_normalizat… |
| mixed10 (Concatenate) | (None, 8, 8, 2048) | 0 | activation_115[0… mixed9_1[0][0], concatenate_1[0]… activation_123[0… |
| global_average_poo… (GlobalAveragePool… | (None, 2048) | 0 | mixed10[0][0] |
| dropout_6 (Dropout) | (None, 2048) | 0 | global_average_p… |
| dense_16 (Dense) | (None, 1024) | 2,098,176 | dropout_6[0][0] |
| dense_17 (Dense) | (None, 4) | 4,100 | dense_16[0][0] |

**Total params:** 28,109,614 (107.23 MB)

**Trainable params:** 2,102,276 (8.02 MB)

**Non-trainable params:** 21,802,784 (83.17 MB)

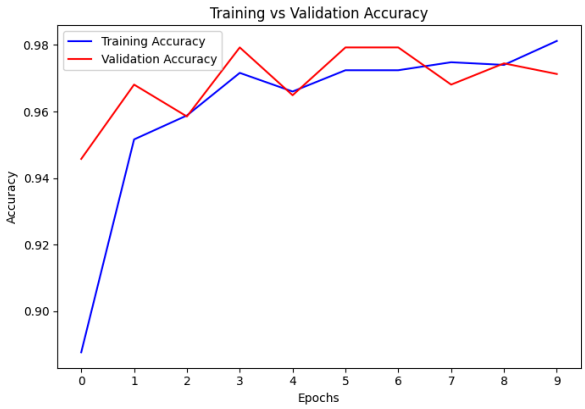
**Optimizer params:** 4,204,554 (16.04 MB)

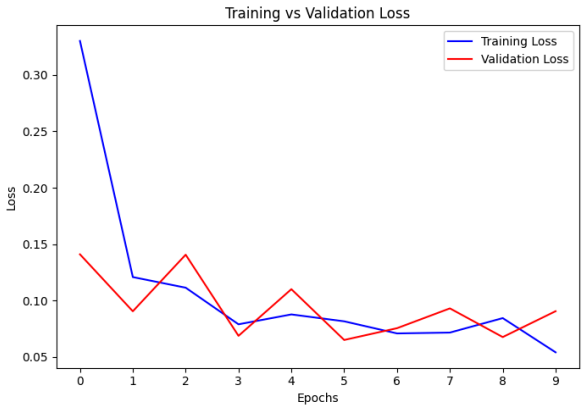
In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model9_Inception_GAP_D
inception_model.save("/content/ninjacart_models/model9_Inception_GAP_DP_FineTune
```

```
Created directory: /content/ninjacart_models/model9_Inception_GAP_DP_FineTune
```

In [ ]:
```
plot_training_history(before_ft_inc_hist)
```

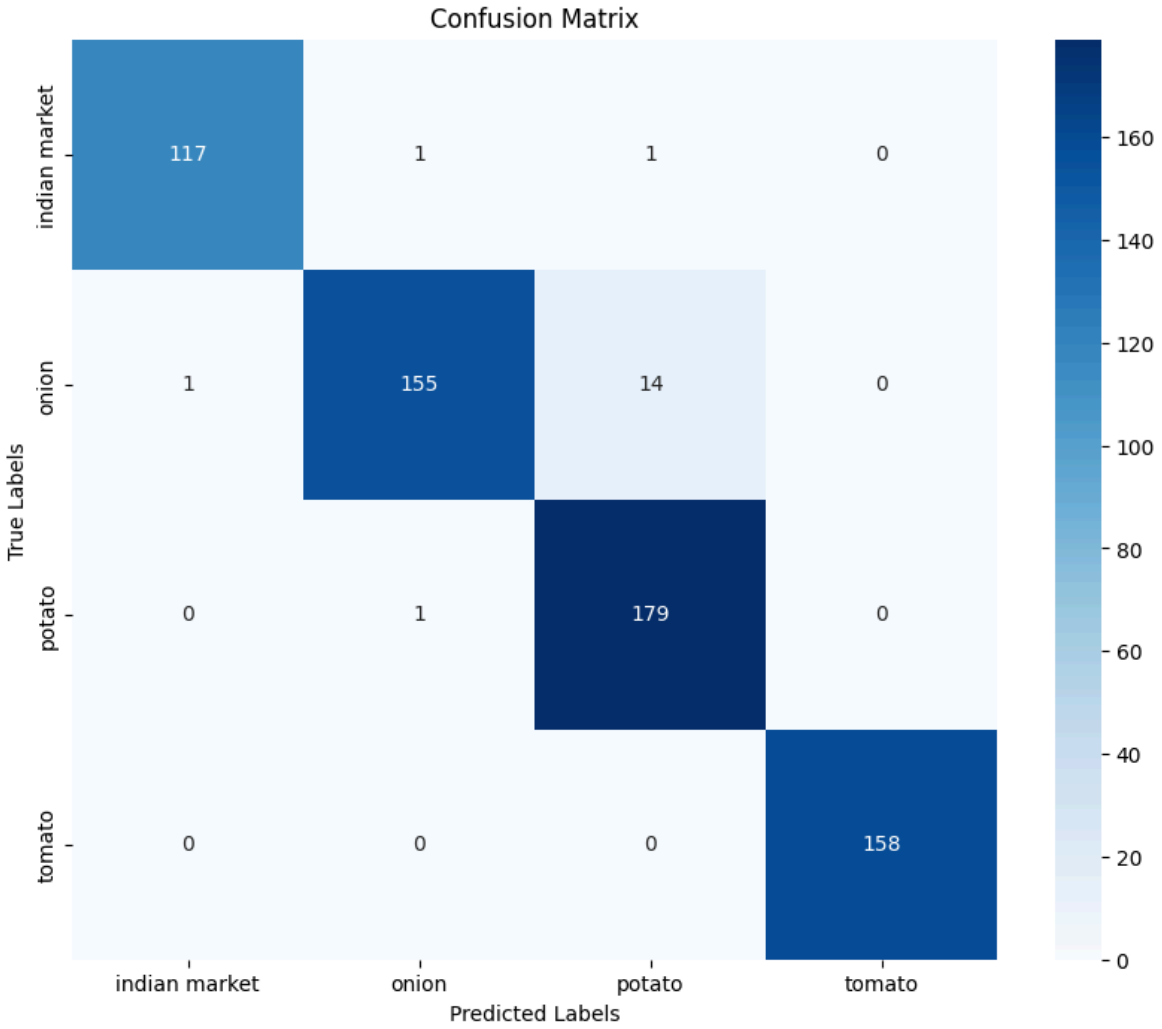
```
History keys: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```




In [ ]:
```
class_names = list(val_data.class_indices.keys())
plot_confusion_matrix(inception_model, val_data, class_names)
```

**20/20** ━━━━━━━━━━━━━━━━━━━━ **17s** 551ms/step



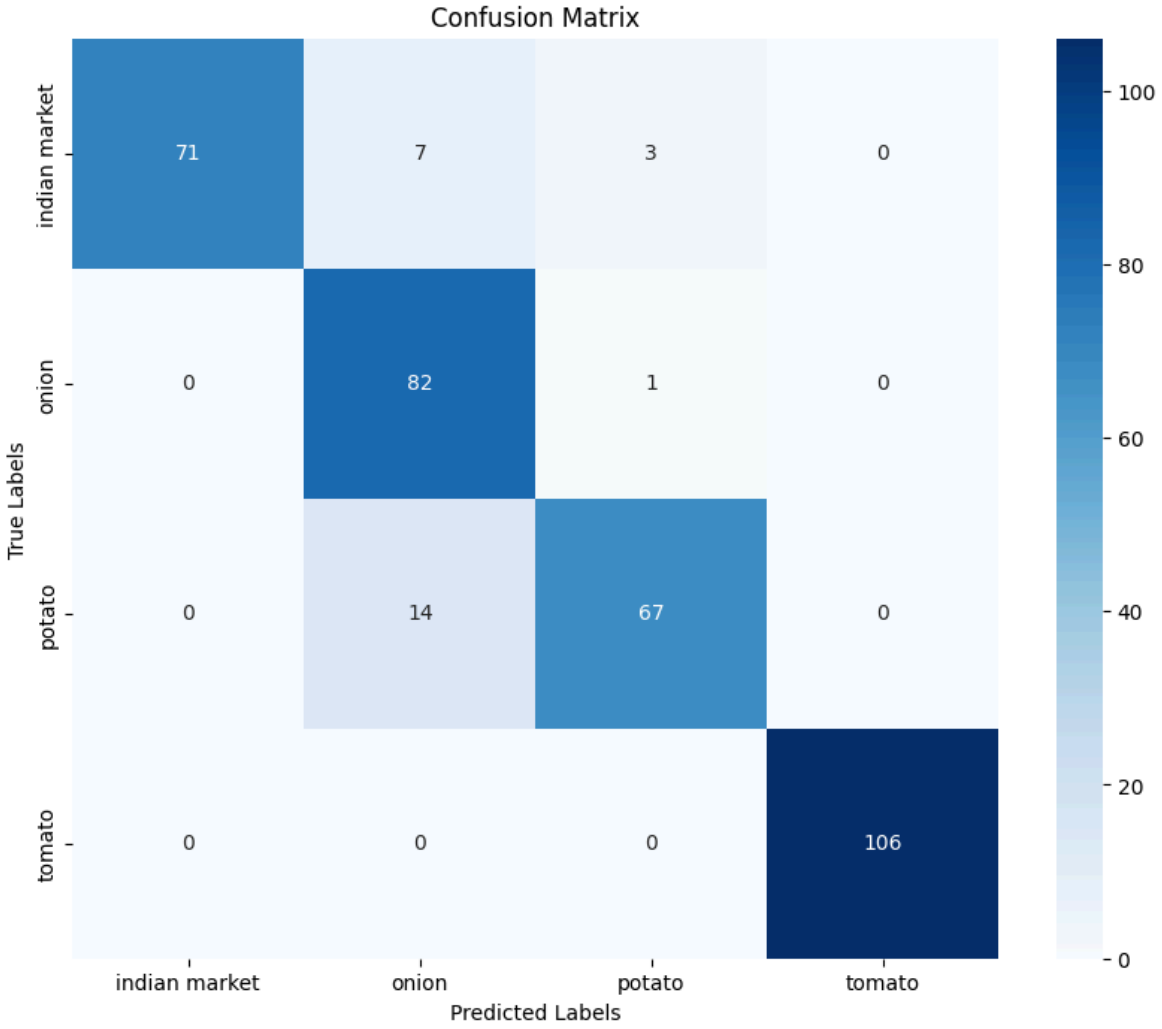
```
Classification Report:

               precision    recall  f1-score   support

indian market       0.99      0.98      0.99       119
        onion       0.99      0.91      0.95       170
       potato       0.92      0.99      0.96       180
       tomato       1.00      1.00      1.00       158

     accuracy                           0.97       627
    macro avg       0.98      0.97      0.97       627
 weighted avg       0.97      0.97      0.97       627
```

In [ ]:
```
inception_model_finetune = tf.keras.models.load_model("/content/ninjacart_models
loss, acc = inception_model_finetune.evaluate(test_data, verbose=2)
print('Before FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_datas
et_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__ini
t__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multipr
ocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
11/11 - 20s - 2s/step - accuracy: 0.9288 - loss: 0.2621
Before FineTuning
Restored model, accuracy: 92.88%
```

In [ ]:
```
class_names = list(test_data.class_indices.keys())
plot_confusion_matrix(inception_model, test_data, class_names)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 5s 516ms/step
```




Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 1.00 | 0.88 | 0.93 | 81 |
| onion | 0.80 | 0.99 | 0.88 | 83 |
| potato | 0.94 | 0.83 | 0.88 | 81 |
| tomato | 1.00 | 1.00 | 1.00 | 106 |
| accuracy | | | 0.93 | 351 |
| macro avg | 0.93 | 0.92 | 0.92 | 351 |
| weighted avg | 0.94 | 0.93 | 0.93 | 351 |

# FineTuning

In [ ]:
```
# Unfreeze last few layers for fine-tuning
for layer in base_model.layers[-50:]:  # You can change -50 to -30 or so
    layer.trainable = True

# Compile again with a lower learning rate
inception_model.compile(optimizer=Adam(learning_rate=1e-5), loss='categorical_cr

# Continue training (fine-tuning)
after_ft_inc_hist = inception_model.fit(train_data, epochs=10, validation_data=v
```

```
Epoch 1/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 63s 507ms/step - accuracy: 0.9282 - loss: 0.1738 - val
_accuracy: 0.9713 - val_loss: 0.0729
Epoch 2/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 21s 269ms/step - accuracy: 0.9714 - loss: 0.0755 - val
_accuracy: 0.9697 - val_loss: 0.0667
Epoch 3/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 21s 259ms/step - accuracy: 0.9757 - loss: 0.0686 - val
_accuracy: 0.9729 - val_loss: 0.0579
Epoch 4/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 21s 271ms/step - accuracy: 0.9720 - loss: 0.0711 - val
_accuracy: 0.9793 - val_loss: 0.0550
Epoch 5/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 21s 263ms/step - accuracy: 0.9881 - loss: 0.0325 - val
_accuracy: 0.9809 - val_loss: 0.0548
Epoch 6/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 22s 282ms/step - accuracy: 0.9899 - loss: 0.0286 - val
_accuracy: 0.9777 - val_loss: 0.0536
Epoch 7/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 19s 245ms/step - accuracy: 0.9900 - loss: 0.0297 - val
_accuracy: 0.9777 - val_loss: 0.0542
Epoch 8/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 22s 273ms/step - accuracy: 0.9940 - loss: 0.0176 - val
_accuracy: 0.9793 - val_loss: 0.0516
Epoch 9/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 20s 255ms/step - accuracy: 0.9937 - loss: 0.0199 - val
_accuracy: 0.9761 - val_loss: 0.0538
Epoch 10/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 22s 280ms/step - accuracy: 0.9914 - loss: 0.0216 - val
_accuracy: 0.9777 - val_loss: 0.0551
```

In [ ]:
```
inception_model.summary()
```

```
Model: "functional_11"
```

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_layer_11 (InputLayer) | (None, 299, 299, 3) | 0 | - |
| conv2d_31 (Conv2D) | (None, 149, 149, 32) | 864 | input_layer_11[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 149, 149, 32) | 96 | conv2d_31[0][0] |
| activation_30 (Activation) | (None, 149, 149, 32) | 0 | batch_normalizat… |
| conv2d_32 (Conv2D) | (None, 147, 147, 32) | 9,216 | activation_30[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 147, 147, 32) | 96 | conv2d_32[0][0] |
| activation_31 (Activation) | (None, 147, 147, 32) | 0 | batch_normalizat… |
| conv2d_33 (Conv2D) | (None, 147, 147, 64) | 18,432 | activation_31[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 147, 147, 64) | 192 | conv2d_33[0][0] |
| activation_32 (Activation) | (None, 147, 147, 64) | 0 | batch_normalizat… |
| max_pooling2d_25 (MaxPooling2D) | (None, 73, 73, 64) | 0 | activation_32[0]… |
| conv2d_34 (Conv2D) | (None, 73, 73, 80) | 5,120 | max_pooling2d_25… |
| batch_normalizatio… (BatchNormalizatio… | (None, 73, 73, 80) | 240 | conv2d_34[0][0] |
| activation_33 (Activation) | (None, 73, 73, 80) | 0 | batch_normalizat… |
| conv2d_35 (Conv2D) | (None, 71, 71, 192) | 138,240 | activation_33[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 71, 71, 192) | 576 | conv2d_35[0][0] |
| activation_34 (Activation) | (None, 71, 71, 192) | 0 | batch_normalizat… |
| max_pooling2d_26 (MaxPooling2D) | (None, 35, 35, 192) | 0 | activation_34[0]… |
| conv2d_39 (Conv2D) | (None, 35, 35, 64) | 12,288 | max_pooling2d_26… |

| | | | |
|---|---|---|---|
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_39[0][0] |
| activation_38 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| conv2d_37 (Conv2D) | (None, 35, 35, 48) | 9,216 | max_pooling2d_26… |
| conv2d_40 (Conv2D) | (None, 35, 35, 96) | 55,296 | activation_38[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 48) | 144 | conv2d_37[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_40[0][0] |
| activation_36 (Activation) | (None, 35, 35, 48) | 0 | batch_normalizat… |
| activation_39 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| average_pooling2d (AveragePooling2D) | (None, 35, 35, 192) | 0 | max_pooling2d_26… |
| conv2d_36 (Conv2D) | (None, 35, 35, 64) | 12,288 | max_pooling2d_26… |
| conv2d_38 (Conv2D) | (None, 35, 35, 64) | 76,800 | activation_36[0]… |
| conv2d_41 (Conv2D) | (None, 35, 35, 96) | 82,944 | activation_39[0]… |
| conv2d_42 (Conv2D) | (None, 35, 35, 32) | 6,144 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_36[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_38[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_41[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 32) | 96 | conv2d_42[0][0] |
| activation_35 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_37 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_40 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| activation_41 (Activation) | (None, 35, 35, 32) | 0 | batch_normalizat… |
| mixed0 (Concatenate) | (None, 35, 35, 256) | 0 | activation_35[0]…<br>activation_37[0]…<br>activation_40[0]…<br>activation_41[0]… |
| conv2d_46 (Conv2D) | (None, 35, 35, 64) | 16,384 | mixed0[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_46[0][0] |
| activation_45 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| conv2d_44 (Conv2D) | (None, 35, 35, 48) | 12,288 | mixed0[0][0] |
| conv2d_47 (Conv2D) | (None, 35, 35, 96) | 55,296 | activation_45[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 48) | 144 | conv2d_44[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_47[0][0] |
| activation_43 (Activation) | (None, 35, 35, 48) | 0 | batch_normalizat… |
| activation_46 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| average_pooling2d_1 (AveragePooling2D) | (None, 35, 35, 256) | 0 | mixed0[0][0] |
| conv2d_43 (Conv2D) | (None, 35, 35, 64) | 16,384 | mixed0[0][0] |
| conv2d_45 (Conv2D) | (None, 35, 35, 64) | 76,800 | activation_43[0]… |
| conv2d_48 (Conv2D) | (None, 35, 35, 96) | 82,944 | activation_46[0]… |
| conv2d_49 (Conv2D) | (None, 35, 35, 64) | 16,384 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_43[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_45[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_48[0][0] |
| batch_normalizatio… | (None, 35, 35, | 192 | conv2d_49[0][0] |

| | | | |
|---|---|---|---|
| (BatchNormalizatio… | 64) | | |
| activation_42 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_44 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_47 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| activation_48 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| mixed1 (Concatenate) | (None, 35, 35, 288) | 0 | activation_42[0]…<br>activation_44[0]…<br>activation_47[0]…<br>activation_48[0]… |
| conv2d_53 (Conv2D) | (None, 35, 35, 64) | 18,432 | mixed1[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_53[0][0] |
| activation_52 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| conv2d_51 (Conv2D) | (None, 35, 35, 48) | 13,824 | mixed1[0][0] |
| conv2d_54 (Conv2D) | (None, 35, 35, 96) | 55,296 | activation_52[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 48) | 144 | conv2d_51[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_54[0][0] |
| activation_50 (Activation) | (None, 35, 35, 48) | 0 | batch_normalizat… |
| activation_53 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| average_pooling2d_2 (AveragePooling2D) | (None, 35, 35, 288) | 0 | mixed1[0][0] |
| conv2d_50 (Conv2D) | (None, 35, 35, 64) | 18,432 | mixed1[0][0] |
| conv2d_52 (Conv2D) | (None, 35, 35, 64) | 76,800 | activation_50[0]… |
| conv2d_55 (Conv2D) | (None, 35, 35, 96) | 82,944 | activation_53[0]… |
| conv2d_56 (Conv2D) | (None, 35, 35, 64) | 18,432 | average_pooling2… |

| | | | |
|---|---|---|---|
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_50[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_52[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_55[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_56[0][0] |
| activation_49 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_51 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| activation_54 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| activation_55 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| mixed2 (Concatenate) | (None, 35, 35, 288) | 0 | activation_49[0]… activation_51[0]… activation_54[0]… activation_55[0]… |
| conv2d_58 (Conv2D) | (None, 35, 35, 64) | 18,432 | mixed2[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 64) | 192 | conv2d_58[0][0] |
| activation_57 (Activation) | (None, 35, 35, 64) | 0 | batch_normalizat… |
| conv2d_59 (Conv2D) | (None, 35, 35, 96) | 55,296 | activation_57[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 35, 35, 96) | 288 | conv2d_59[0][0] |
| activation_58 (Activation) | (None, 35, 35, 96) | 0 | batch_normalizat… |
| conv2d_57 (Conv2D) | (None, 17, 17, 384) | 995,328 | mixed2[0][0] |
| conv2d_60 (Conv2D) | (None, 17, 17, 96) | 82,944 | activation_58[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 384) | 1,152 | conv2d_57[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 96) | 288 | conv2d_60[0][0] |

| | | | |
|---|---|---|---|
| activation_56 (Activation) | (None, 17, 17, 384) | 0 | batch_normalizat… |
| activation_59 (Activation) | (None, 17, 17, 96) | 0 | batch_normalizat… |
| max_pooling2d_27 (MaxPooling2D) | (None, 17, 17, 288) | 0 | mixed2[0][0] |
| mixed3 (Concatenate) | (None, 17, 17, 768) | 0 | activation_56[0]…<br>activation_59[0]…<br>max_pooling2d_27… |
| conv2d_65 (Conv2D) | (None, 17, 17, 128) | 98,304 | mixed3[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_65[0][0] |
| activation_64 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| conv2d_66 (Conv2D) | (None, 17, 17, 128) | 114,688 | activation_64[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_66[0][0] |
| activation_65 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| conv2d_62 (Conv2D) | (None, 17, 17, 128) | 98,304 | mixed3[0][0] |
| conv2d_67 (Conv2D) | (None, 17, 17, 128) | 114,688 | activation_65[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_62[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_67[0][0] |
| activation_61 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| activation_66 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| conv2d_63 (Conv2D) | (None, 17, 17, 128) | 114,688 | activation_61[0]… |
| conv2d_68 (Conv2D) | (None, 17, 17, 128) | 114,688 | activation_66[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_63[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 128) | 384 | conv2d_68[0][0] |

| | | | |
|---|---|---|---|
| activation_62 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| activation_67 (Activation) | (None, 17, 17, 128) | 0 | batch_normalizat… |
| average_pooling2d_3 (AveragePooling2D) | (None, 17, 17, 768) | 0 | mixed3[0][0] |
| conv2d_61 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed3[0][0] |
| conv2d_64 (Conv2D) | (None, 17, 17, 192) | 172,032 | activation_62[0]… |
| conv2d_69 (Conv2D) | (None, 17, 17, 192) | 172,032 | activation_67[0]… |
| conv2d_70 (Conv2D) | (None, 17, 17, 192) | 147,456 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_61[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_64[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_69[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_70[0][0] |
| activation_60 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_63 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_68 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_69 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| mixed4 (Concatenate) | (None, 17, 17, 768) | 0 | activation_60[0]… activation_63[0]… activation_68[0]… activation_69[0]… |
| conv2d_75 (Conv2D) | (None, 17, 17, 160) | 122,880 | mixed4[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_75[0][0] |
| activation_74 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| conv2d_76 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_74[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_76[0][0] |
| activation_75 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_72 (Conv2D) | (None, 17, 17, 160) | 122,880 | mixed4[0][0] |
| conv2d_77 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_75[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_72[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_77[0][0] |
| activation_71 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| activation_76 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_73 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_71[0]… |
| conv2d_78 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_76[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_73[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_78[0][0] |
| activation_72 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| activation_77 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| average_pooling2d_4 (AveragePooling2D) | (None, 17, 17, 768) | 0 | mixed4[0][0] |
| conv2d_71 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed4[0][0] |
| conv2d_74 (Conv2D) | (None, 17, 17, 192) | 215,040 | activation_72[0]… |
| conv2d_79 (Conv2D) | (None, 17, 17, 192) | 215,040 | activation_77[0]… |
| conv2d_80 (Conv2D) | (None, 17, 17, 192) | 147,456 | average_pooling2… |

| | | | |
|---|---|---|---|
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_71[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_74[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_79[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_80[0][0] |
| activation_70 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_73 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_78 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_79 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| mixed5 (Concatenate) | (None, 17, 17, 768) | 0 | activation_70[0]… activation_73[0]… activation_78[0]… activation_79[0]… |
| conv2d_85 (Conv2D) | (None, 17, 17, 160) | 122,880 | mixed5[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_85[0][0] |
| activation_84 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_86 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_84[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_86[0][0] |
| activation_85 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_82 (Conv2D) | (None, 17, 17, 160) | 122,880 | mixed5[0][0] |
| conv2d_87 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_85[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_82[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_87[0][0] |
| activation_81 | (None, 17, 17, | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| (Activation) | 160) | | |
| activation_86 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| conv2d_83 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_81[0]… |
| conv2d_88 (Conv2D) | (None, 17, 17, 160) | 179,200 | activation_86[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_83[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 160) | 480 | conv2d_88[0][0] |
| activation_82 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| activation_87 (Activation) | (None, 17, 17, 160) | 0 | batch_normalizat… |
| average_pooling2d_5 (AveragePooling2D) | (None, 17, 17, 768) | 0 | mixed5[0][0] |
| conv2d_81 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed5[0][0] |
| conv2d_84 (Conv2D) | (None, 17, 17, 192) | 215,040 | activation_82[0]… |
| conv2d_89 (Conv2D) | (None, 17, 17, 192) | 215,040 | activation_87[0]… |
| conv2d_90 (Conv2D) | (None, 17, 17, 192) | 147,456 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_81[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_84[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_89[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_90[0][0] |
| activation_80 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_83 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_88 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_89 | (None, 17, 17, | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| (Activation) | 192) | | |
| mixed6 (Concatenate) | (None, 17, 17, 768) | 0 | activation_80[0]…<br>activation_83[0]…<br>activation_88[0]…<br>activation_89[0]… |
| conv2d_95 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed6[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_95[0][0] |
| activation_94 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_96 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_94[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_96[0][0] |
| activation_95 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_92 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed6[0][0] |
| conv2d_97 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_95[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_92[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_97[0][0] |
| activation_91 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_96 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_93 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_91[0]… |
| conv2d_98 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_96[0]… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_93[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_98[0][0] |
| activation_92 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_97 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |

| | | | |
|---|---|---|---|
| average_pooling2d_6 (AveragePooling2D) | (None, 17, 17, 768) | 0 | mixed6[0][0] |
| conv2d_91 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed6[0][0] |
| conv2d_94 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_92[0]… |
| conv2d_99 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_97[0]… |
| conv2d_100 (Conv2D) | (None, 17, 17, 192) | 147,456 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_91[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_94[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_99[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_100[0][0] |
| activation_90 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_93 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_98 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_99 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| mixed7 (Concatenate) | (None, 17, 17, 768) | 0 | activation_90[0]… activation_93[0]… activation_98[0]… activation_99[0]… |
| conv2d_103 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed7[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_103[0][0] |
| activation_102 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_104 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_102[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_104[0][0] |

| | | | |
|---|---|---|---|
| activation_103 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_101 (Conv2D) | (None, 17, 17, 192) | 147,456 | mixed7[0][0] |
| conv2d_105 (Conv2D) | (None, 17, 17, 192) | 258,048 | activation_103[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_101[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 17, 17, 192) | 576 | conv2d_105[0][0] |
| activation_100 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| activation_104 (Activation) | (None, 17, 17, 192) | 0 | batch_normalizat… |
| conv2d_102 (Conv2D) | (None, 8, 8, 320) | 552,960 | activation_100[0… |
| conv2d_106 (Conv2D) | (None, 8, 8, 192) | 331,776 | activation_104[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 320) | 960 | conv2d_102[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 192) | 576 | conv2d_106[0][0] |
| activation_101 (Activation) | (None, 8, 8, 320) | 0 | batch_normalizat… |
| activation_105 (Activation) | (None, 8, 8, 192) | 0 | batch_normalizat… |
| max_pooling2d_28 (MaxPooling2D) | (None, 8, 8, 768) | 0 | mixed7[0][0] |
| mixed8 (Concatenate) | (None, 8, 8, 1280) | 0 | activation_101[0… activation_105[0… max_pooling2d_28… |
| conv2d_111 (Conv2D) | (None, 8, 8, 448) | 573,440 | mixed8[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 448) | 1,344 | conv2d_111[0][0] |
| activation_110 (Activation) | (None, 8, 8, 448) | 0 | batch_normalizat… |
| conv2d_108 (Conv2D) | (None, 8, 8, 384) | 491,520 | mixed8[0][0] |
| conv2d_112 (Conv2D) | (None, 8, 8, 384) | 1,548,288 | activation_110[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_108[0][0] |
| batch_normalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_112[0][0] |

| | | | |
|---|---|---|---|
| (BatchNormalizatio… | | | |
| activation_107 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_111 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| conv2d_109 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_107[0… |
| conv2d_110 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_107[0… |
| conv2d_113 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_111[0… |
| conv2d_114 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_111[0… |
| average_pooling2d_7 (AveragePooling2D) | (None, 8, 8, 1280) | 0 | mixed8[0][0] |
| conv2d_107 (Conv2D) | (None, 8, 8, 320) | 409,600 | mixed8[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_109[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_110[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_113[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_114[0][0] |
| conv2d_115 (Conv2D) | (None, 8, 8, 192) | 245,760 | average_pooling2… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 320) | 960 | conv2d_107[0][0] |
| activation_108 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_109 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_112 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_113 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 192) | 576 | conv2d_115[0][0] |
| activation_106 (Activation) | (None, 8, 8, 320) | 0 | batch_normalizat… |
| mixed9_0 (Concatenate) | (None, 8, 8, 768) | 0 | activation_108[0… activation_109[0… |
| concatenate | (None, 8, 8, 768) | 0 | activation_112[0… |

| | | | |
|---|---|---|---|
| (Concatenate) | | | activation_113[0… |
| activation_114 (Activation) | (None, 8, 8, 192) | 0 | batch_normalizat… |
| mixed9 (Concatenate) | (None, 8, 8, 2048) | 0 | activation_106[0… mixed9_0[0][0], concatenate[0][0… activation_114[0… |
| conv2d_120 (Conv2D) | (None, 8, 8, 448) | 917,504 | mixed9[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 448) | 1,344 | conv2d_120[0][0] |
| activation_119 (Activation) | (None, 8, 8, 448) | 0 | batch_normalizat… |
| conv2d_117 (Conv2D) | (None, 8, 8, 384) | 786,432 | mixed9[0][0] |
| conv2d_121 (Conv2D) | (None, 8, 8, 384) | 1,548,288 | activation_119[0… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_117[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_121[0][0] |
| activation_116 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_120 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| conv2d_118 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_116[0… |
| conv2d_119 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_116[0… |
| conv2d_122 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_120[0… |
| conv2d_123 (Conv2D) | (None, 8, 8, 384) | 442,368 | activation_120[0… |
| average_pooling2d_8 (AveragePooling2D) | (None, 8, 8, 2048) | 0 | mixed9[0][0] |
| conv2d_116 (Conv2D) | (None, 8, 8, 320) | 655,360 | mixed9[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_118[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_119[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_122[0][0] |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 384) | 1,152 | conv2d_123[0][0] |
| conv2d_124 (Conv2D) | (None, 8, 8, 192) | 393,216 | average_pooling2… |

| | | | |
|---|---|---|---|
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 320) | 960 | conv2d_116[0][0] |
| activation_117 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_118 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_121 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| activation_122 (Activation) | (None, 8, 8, 384) | 0 | batch_normalizat… |
| batch_normalizatio… (BatchNormalizatio… | (None, 8, 8, 192) | 576 | conv2d_124[0][0] |
| activation_115 (Activation) | (None, 8, 8, 320) | 0 | batch_normalizat… |
| mixed9_1 (Concatenate) | (None, 8, 8, 768) | 0 | activation_117[0… activation_118[0… |
| concatenate_1 (Concatenate) | (None, 8, 8, 768) | 0 | activation_121[0… activation_122[0… |
| activation_123 (Activation) | (None, 8, 8, 192) | 0 | batch_normalizat… |
| mixed10 (Concatenate) | (None, 8, 8, 2048) | 0 | activation_115[0… mixed9_1[0][0], concatenate_1[0]… activation_123[0… |
| global_average_poo… (GlobalAveragePool… | (None, 2048) | 0 | mixed10[0][0] |
| dropout_6 (Dropout) | (None, 2048) | 0 | global_average_p… |
| dense_16 (Dense) | (None, 1024) | 2,098,176 | dropout_6[0][0] |
| dense_17 (Dense) | (None, 4) | 4,100 | dense_16[0][0] |

**Total params:** 42,456,238 (161.96 MB)

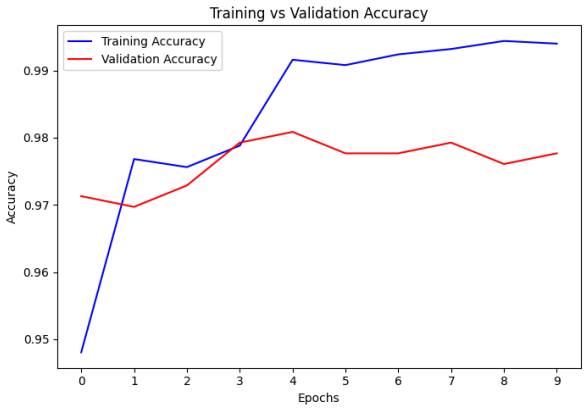
**Trainable params:** 9,275,588 (35.38 MB)

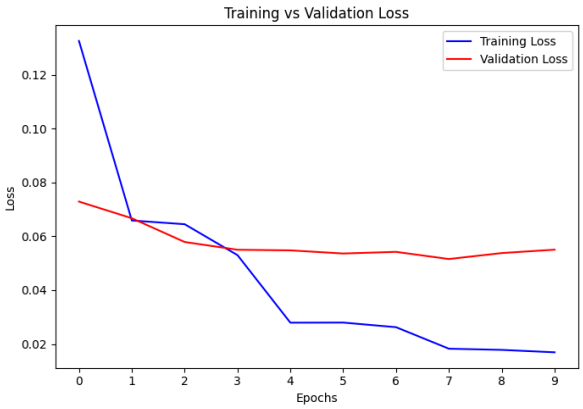
**Non-trainable params:** 14,629,472 (55.81 MB)

**Optimizer params:** 18,551,178 (70.77 MB)

In [ ]:
```
plot_training_history(after_ft_inc_hist)
```

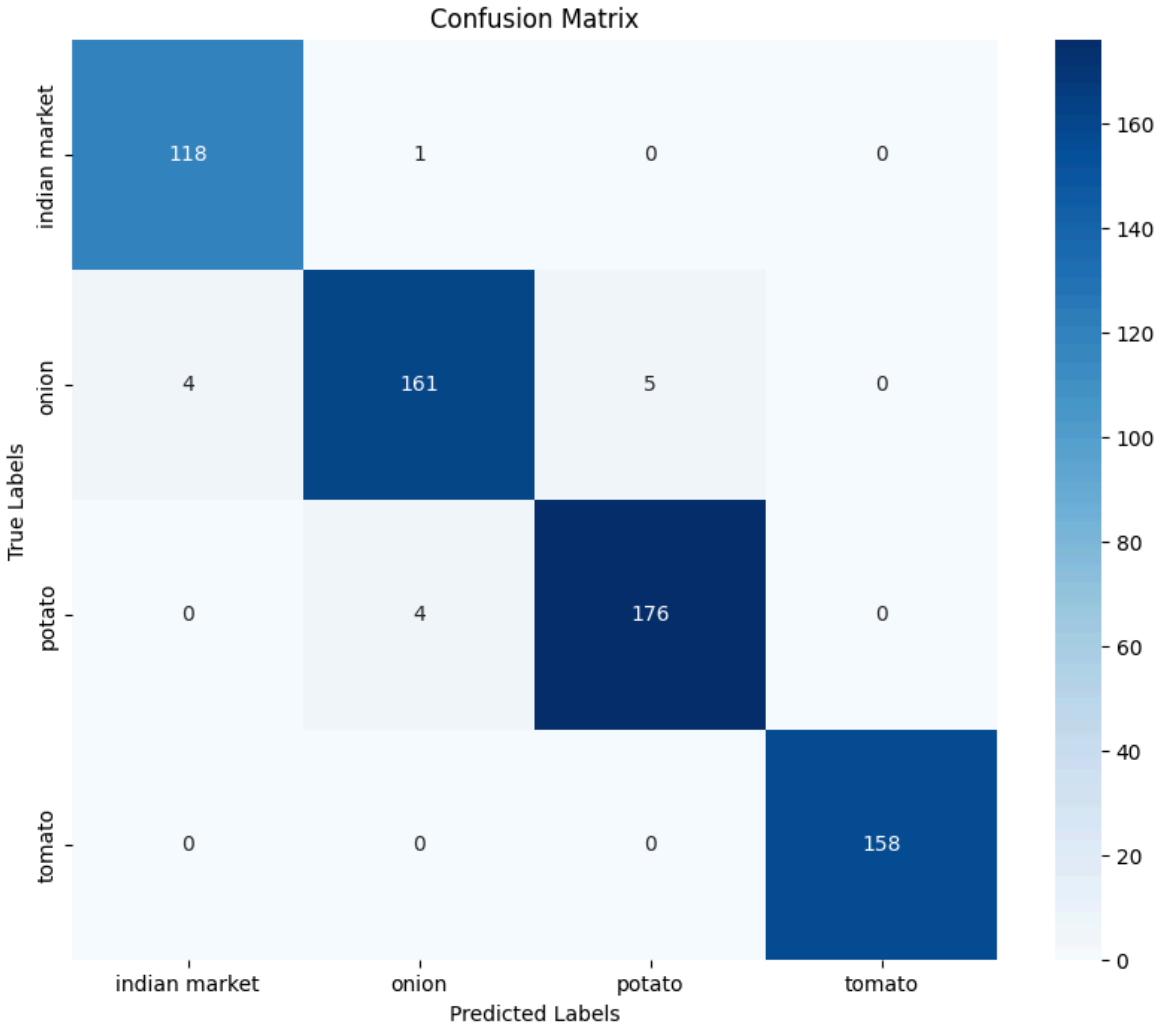
```
History keys: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```




In [ ]:
```
class_names = list(val_data.class_indices.keys())
plot_confusion_matrix(inception_model, val_data, class_names)
```

**20/20** ━━━━━━━━━━━━━━━━━━━━ **21s** 642ms/step



```
Classification Report:

                precision    recall  f1-score   support

indian market       0.97      0.99      0.98       119
        onion       0.97      0.95      0.96       170
       potato       0.97      0.98      0.98       180
       tomato       1.00      1.00      1.00       158

     accuracy                           0.98       627
    macro avg       0.98      0.98      0.98       627
 weighted avg       0.98      0.98      0.98       627
```

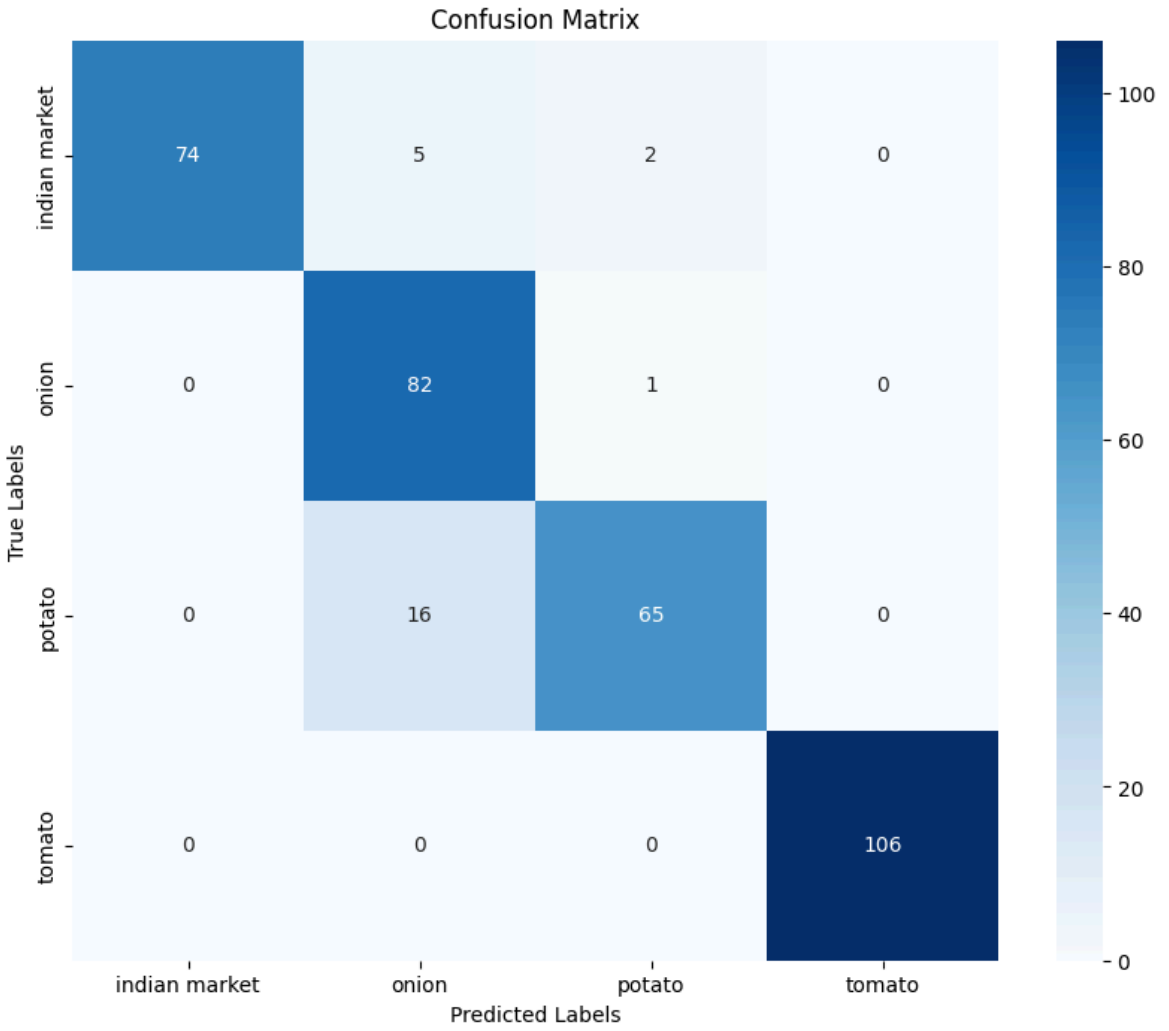
In [ ]:
```
inception_model.save("/content/ninjacart_models/model9_Inception_GAP_DP_FineTune
```

In [ ]:
```
inception_model_finetune = tf.keras.models.load_model("/content/ninjacart_models
loss, acc = inception_model_finetune.evaluate(test_data, verbose=2)
print('After FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
11/11 - 14s - 1s/step - accuracy: 0.9316 - loss: 0.2883
After FineTuning
Restored model, accuracy: 93.16%
```

In [ ]:
```
class_names = list(test_data.class_indices.keys())
plot_confusion_matrix(inception_model, test_data, class_names)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 5s 509ms/step
```



```
Classification Report:

               precision    recall  f1-score   support

indian market       1.00      0.91      0.95        81
        onion       0.80      0.99      0.88        83
       potato       0.96      0.80      0.87        81
       tomato       1.00      1.00      1.00       106

     accuracy                           0.93       351
    macro avg       0.94      0.93      0.93       351
 weighted avg       0.94      0.93      0.93       351
```

# Resnet50

In [ ]:
```
datagen = ImageDataGenerator(preprocessing_function=resnet50.preprocess_input)

train_data = datagen.flow_from_directory('/content/ninjacart_data/train', target
val_data = datagen.flow_from_directory('/content/ninjacart_data/validation', tar
test_data = datagen.flow_from_directory('/content/ninjacart_data/test', target_s
```

```
Found 2502 images belonging to 4 classes.
Found 627 images belonging to 4 classes.
Found 351 images belonging to 4 classes.
```

# Without finetuning

In [ ]:
```
num_classes = 4
# Step 1: Load base model (without top layer)
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 2

# Step 2: Freeze all layers (for feature extraction)
for layer in base_model.layers:
    layer.trainable = False

# Step 3: Add custom top layers
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.5)(x)  # optional regularization
x = tf.keras.layers.Dense(512, activation='relu')(x)
predictions = tf.keras.layers.Dense(num_classes, activation='softmax')(x)  # num

# Final model
resnet50_model = Model(inputs=base_model.input, outputs=predictions)

# Step 4: Compile the model
resnet50_model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_cr

# Step 5: Train only the top layers
before_fine_tuning = resnet50_model.fit(train_data, epochs=10, validation_data=v
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-application
s/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 43s 369ms/step - accuracy: 0.8178 - loss: 0.5439 - val_accuracy: 0.9585 - val_loss: 0.1233
Epoch 2/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 16s 205ms/step - accuracy: 0.9680 - loss: 0.0834 - val_accuracy: 0.9649 - val_loss: 0.1008
Epoch 3/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 16s 203ms/step - accuracy: 0.9704 - loss: 0.0798 - val_accuracy: 0.9697 - val_loss: 0.0889
Epoch 4/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 16s 206ms/step - accuracy: 0.9850 - loss: 0.0401 - val_accuracy: 0.9681 - val_loss: 0.0823
Epoch 5/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 18s 233ms/step - accuracy: 0.9826 - loss: 0.0475 - val_accuracy: 0.9633 - val_loss: 0.0973
Epoch 6/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 16s 207ms/step - accuracy: 0.9881 - loss: 0.0345 - val_accuracy: 0.9777 - val_loss: 0.0612
Epoch 7/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 212ms/step - accuracy: 0.9900 - loss: 0.0276 - val_accuracy: 0.9729 - val_loss: 0.0968
Epoch 8/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 16s 204ms/step - accuracy: 0.9827 - loss: 0.0320 - val_accuracy: 0.9681 - val_loss: 0.0691
Epoch 9/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 18s 231ms/step - accuracy: 0.9891 - loss: 0.0312 - val_accuracy: 0.9761 - val_loss: 0.0820
Epoch 10/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 16s 202ms/step - accuracy: 0.9901 - loss: 0.0289 - val_accuracy: 0.9697 - val_loss: 0.0651
```

In [ ]:
```
resnet50_model.summary()
```

**Model: "functional_12"**

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_layer_12 (InputLayer) | (None, 224, 224, 3) | 0 | - |
| conv1_pad (ZeroPadding2D) | (None, 230, 230, 3) | 0 | input_layer_12[0… |
| conv1_conv (Conv2D) | (None, 112, 112, 64) | 9,472 | conv1_pad[0][0] |
| conv1_bn (BatchNormalizatio… | (None, 112, 112, 64) | 256 | conv1_conv[0][0] |
| conv1_relu (Activation) | (None, 112, 112, 64) | 0 | conv1_bn[0][0] |
| pool1_pad (ZeroPadding2D) | (None, 114, 114, 64) | 0 | conv1_relu[0][0] |
| pool1_pool (MaxPooling2D) | (None, 56, 56, 64) | 0 | pool1_pad[0][0] |
| conv2_block1_1_conv (Conv2D) | (None, 56, 56, 64) | 4,160 | pool1_pool[0][0] |
| conv2_block1_1_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block1_1_c… |
| conv2_block1_1_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block1_1_b… |
| conv2_block1_2_conv (Conv2D) | (None, 56, 56, 64) | 36,928 | conv2_block1_1_r… |
| conv2_block1_2_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block1_2_c… |
| conv2_block1_2_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block1_2_b… |
| conv2_block1_0_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | pool1_pool[0][0] |
| conv2_block1_3_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | conv2_block1_2_r… |
| conv2_block1_0_bn (BatchNormalizatio… | (None, 56, 56, 256) | 1,024 | conv2_block1_0_c… |
| conv2_block1_3_bn (BatchNormalizatio… | (None, 56, 56, 256) | 1,024 | conv2_block1_3_c… |
| conv2_block1_add (Add) | (None, 56, 56, 256) | 0 | conv2_block1_0_b… conv2_block1_3_b… |
| conv2_block1_out (Activation) | (None, 56, 56, 256) | 0 | conv2_block1_add… |

| | | | |
|---|---|---|---|
| conv2_block2_1_conv (Conv2D) | (None, 56, 56, 64) | 16,448 | conv2_block1_out… |
| conv2_block2_1_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block2_1_c… |
| conv2_block2_1_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block2_1_b… |
| conv2_block2_2_conv (Conv2D) | (None, 56, 56, 64) | 36,928 | conv2_block2_1_r… |
| conv2_block2_2_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block2_2_c… |
| conv2_block2_2_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block2_2_b… |
| conv2_block2_3_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | conv2_block2_2_r… |
| conv2_block2_3_bn (BatchNormalizatio… | (None, 56, 56, 256) | 1,024 | conv2_block2_3_c… |
| conv2_block2_add (Add) | (None, 56, 56, 256) | 0 | conv2_block1_out… conv2_block2_3_b… |
| conv2_block2_out (Activation) | (None, 56, 56, 256) | 0 | conv2_block2_add… |
| conv2_block3_1_conv (Conv2D) | (None, 56, 56, 64) | 16,448 | conv2_block2_out… |
| conv2_block3_1_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block3_1_c… |
| conv2_block3_1_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block3_1_b… |
| conv2_block3_2_conv (Conv2D) | (None, 56, 56, 64) | 36,928 | conv2_block3_1_r… |
| conv2_block3_2_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block3_2_c… |
| conv2_block3_2_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block3_2_b… |
| conv2_block3_3_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | conv2_block3_2_r… |
| conv2_block3_3_bn (BatchNormalizatio… | (None, 56, 56, 256) | 1,024 | conv2_block3_3_c… |
| conv2_block3_add (Add) | (None, 56, 56, 256) | 0 | conv2_block2_out… conv2_block3_3_b… |
| conv2_block3_out (Activation) | (None, 56, 56, 256) | 0 | conv2_block3_add… |

| | | | |
|---|---|---|---|
| conv3_block1_1_conv (Conv2D) | (None, 28, 28, 128) | 32,896 | conv2_block3_out… |
| conv3_block1_1_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block1_1_c… |
| conv3_block1_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block1_1_b… |
| conv3_block1_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block1_1_r… |
| conv3_block1_2_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block1_2_c… |
| conv3_block1_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block1_2_b… |
| conv3_block1_0_conv (Conv2D) | (None, 28, 28, 512) | 131,584 | conv2_block3_out… |
| conv3_block1_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block1_2_r… |
| conv3_block1_0_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block1_0_c… |
| conv3_block1_3_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block1_3_c… |
| conv3_block1_add (Add) | (None, 28, 28, 512) | 0 | conv3_block1_0_b… conv3_block1_3_b… |
| conv3_block1_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block1_add… |
| conv3_block2_1_conv (Conv2D) | (None, 28, 28, 128) | 65,664 | conv3_block1_out… |
| conv3_block2_1_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block2_1_c… |
| conv3_block2_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block2_1_b… |
| conv3_block2_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block2_1_r… |
| conv3_block2_2_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block2_2_c… |
| conv3_block2_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block2_2_b… |
| conv3_block2_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block2_2_r… |
| conv3_block2_3_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block2_3_c… |

| | | | |
|---|---|---|---|
| conv3_block2_add (Add) | (None, 28, 28, 512) | 0 | conv3_block1_out… conv3_block2_3_b… |
| conv3_block2_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block2_add… |
| conv3_block3_1_conv (Conv2D) | (None, 28, 28, 128) | 65,664 | conv3_block2_out… |
| conv3_block3_1_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block3_1_c… |
| conv3_block3_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block3_1_b… |
| conv3_block3_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block3_1_r… |
| conv3_block3_2_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block3_2_c… |
| conv3_block3_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block3_2_b… |
| conv3_block3_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block3_2_r… |
| conv3_block3_3_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block3_3_c… |
| conv3_block3_add (Add) | (None, 28, 28, 512) | 0 | conv3_block2_out… conv3_block3_3_b… |
| conv3_block3_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block3_add… |
| conv3_block4_1_conv (Conv2D) | (None, 28, 28, 128) | 65,664 | conv3_block3_out… |
| conv3_block4_1_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block4_1_c… |
| conv3_block4_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block4_1_b… |
| conv3_block4_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block4_1_r… |
| conv3_block4_2_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block4_2_c… |
| conv3_block4_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block4_2_b… |
| conv3_block4_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block4_2_r… |
| conv3_block4_3_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block4_3_c… |

| | | | |
|---|---|---|---|
| conv3_block4_add (Add) | (None, 28, 28, 512) | 0 | conv3_block3_out… conv3_block4_3_b… |
| conv3_block4_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block4_add… |
| conv4_block1_1_conv (Conv2D) | (None, 14, 14, 256) | 131,328 | conv3_block4_out… |
| conv4_block1_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block1_1_c… |
| conv4_block1_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block1_1_b… |
| conv4_block1_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block1_1_r… |
| conv4_block1_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block1_2_c… |
| conv4_block1_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block1_2_b… |
| conv4_block1_0_conv (Conv2D) | (None, 14, 14, 1024) | 525,312 | conv3_block4_out… |
| conv4_block1_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block1_2_r… |
| conv4_block1_0_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block1_0_c… |
| conv4_block1_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block1_3_c… |
| conv4_block1_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block1_0_b… conv4_block1_3_b… |
| conv4_block1_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block1_add… |
| conv4_block2_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block1_out… |
| conv4_block2_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block2_1_c… |
| conv4_block2_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block2_1_b… |
| conv4_block2_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block2_1_r… |
| conv4_block2_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block2_2_c… |
| conv4_block2_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block2_2_b… |

| | | | |
|---|---|---|---|
| conv4_block2_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block2_2_r… |
| conv4_block2_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block2_3_c… |
| conv4_block2_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block1_out… conv4_block2_3_b… |
| conv4_block2_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block2_add… |
| conv4_block3_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block2_out… |
| conv4_block3_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block3_1_c… |
| conv4_block3_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block3_1_b… |
| conv4_block3_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block3_1_r… |
| conv4_block3_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block3_2_c… |
| conv4_block3_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block3_2_b… |
| conv4_block3_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block3_2_r… |
| conv4_block3_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block3_3_c… |
| conv4_block3_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block2_out… conv4_block3_3_b… |
| conv4_block3_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block3_add… |
| conv4_block4_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block3_out… |
| conv4_block4_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block4_1_c… |
| conv4_block4_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block4_1_b… |
| conv4_block4_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block4_1_r… |
| conv4_block4_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block4_2_c… |
| conv4_block4_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block4_2_b… |

| | | | |
|---|---|---|---|
| conv4_block4_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block4_2_r… |
| conv4_block4_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block4_3_c… |
| conv4_block4_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block3_out… conv4_block4_3_b… |
| conv4_block4_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block4_add… |
| conv4_block5_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block4_out… |
| conv4_block5_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block5_1_c… |
| conv4_block5_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block5_1_b… |
| conv4_block5_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block5_1_r… |
| conv4_block5_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block5_2_c… |
| conv4_block5_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block5_2_b… |
| conv4_block5_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block5_2_r… |
| conv4_block5_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block5_3_c… |
| conv4_block5_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block4_out… conv4_block5_3_b… |
| conv4_block5_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block5_add… |
| conv4_block6_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block5_out… |
| conv4_block6_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block6_1_c… |
| conv4_block6_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block6_1_b… |
| conv4_block6_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block6_1_r… |
| conv4_block6_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block6_2_c… |
| conv4_block6_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block6_2_b… |

| | | | |
|---|---|---|---|
| conv4_block6_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block6_2_r… |
| conv4_block6_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block6_3_c… |
| conv4_block6_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block5_out… conv4_block6_3_b… |
| conv4_block6_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block6_add… |
| conv5_block1_1_conv (Conv2D) | (None, 7, 7, 512) | 524,800 | conv4_block6_out… |
| conv5_block1_1_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block1_1_c… |
| conv5_block1_1_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block1_1_b… |
| conv5_block1_2_conv (Conv2D) | (None, 7, 7, 512) | 2,359,808 | conv5_block1_1_r… |
| conv5_block1_2_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block1_2_c… |
| conv5_block1_2_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block1_2_b… |
| conv5_block1_0_conv (Conv2D) | (None, 7, 7, 2048) | 2,099,200 | conv4_block6_out… |
| conv5_block1_3_conv (Conv2D) | (None, 7, 7, 2048) | 1,050,624 | conv5_block1_2_r… |
| conv5_block1_0_bn (BatchNormalizatio… | (None, 7, 7, 2048) | 8,192 | conv5_block1_0_c… |
| conv5_block1_3_bn (BatchNormalizatio… | (None, 7, 7, 2048) | 8,192 | conv5_block1_3_c… |
| conv5_block1_add (Add) | (None, 7, 7, 2048) | 0 | conv5_block1_0_b… conv5_block1_3_b… |
| conv5_block1_out (Activation) | (None, 7, 7, 2048) | 0 | conv5_block1_add… |
| conv5_block2_1_conv (Conv2D) | (None, 7, 7, 512) | 1,049,088 | conv5_block1_out… |
| conv5_block2_1_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block2_1_c… |
| conv5_block2_1_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block2_1_b… |
| conv5_block2_2_conv (Conv2D) | (None, 7, 7, 512) | 2,359,808 | conv5_block2_1_r… |

| | | | |
|---|---|---|---|
| conv5_block2_2_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block2_2_c… |
| conv5_block2_2_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block2_2_b… |
| conv5_block2_3_conv (Conv2D) | (None, 7, 7, 2048) | 1,050,624 | conv5_block2_2_r… |
| conv5_block2_3_bn (BatchNormalizatio… | (None, 7, 7, 2048) | 8,192 | conv5_block2_3_c… |
| conv5_block2_add (Add) | (None, 7, 7, 2048) | 0 | conv5_block1_out… conv5_block2_3_b… |
| conv5_block2_out (Activation) | (None, 7, 7, 2048) | 0 | conv5_block2_add… |
| conv5_block3_1_conv (Conv2D) | (None, 7, 7, 512) | 1,049,088 | conv5_block2_out… |
| conv5_block3_1_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block3_1_c… |
| conv5_block3_1_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block3_1_b… |
| conv5_block3_2_conv (Conv2D) | (None, 7, 7, 512) | 2,359,808 | conv5_block3_1_r… |
| conv5_block3_2_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block3_2_c… |
| conv5_block3_2_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block3_2_b… |
| conv5_block3_3_conv (Conv2D) | (None, 7, 7, 2048) | 1,050,624 | conv5_block3_2_r… |
| conv5_block3_3_bn (BatchNormalizatio… | (None, 7, 7, 2048) | 8,192 | conv5_block3_3_c… |
| conv5_block3_add (Add) | (None, 7, 7, 2048) | 0 | conv5_block2_out… conv5_block3_3_b… |
| conv5_block3_out (Activation) | (None, 7, 7, 2048) | 0 | conv5_block3_add… |
| global_average_poo… (GlobalAveragePool… | (None, 2048) | 0 | conv5_block3_out… |
| dropout_7 (Dropout) | (None, 2048) | 0 | global_average_p… |
| dense_18 (Dense) | (None, 512) | 1,049,088 | dropout_7[0][0] |
| dense_19 (Dense) | (None, 4) | 2,052 | dense_18[0][0] |

**Total params:** 26,741,134 (102.01 MB)

**Trainable params:** 1,051,140 (4.01 MB)

**Non-trainable params:** 23,587,712 (89.98 MB)

**Optimizer params:** 2,102,282 (8.02 MB)

In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model10_Resent_GAP_DP_
resnet50_model.save("/content/ninjacart_models/model10_Resent_GAP_DP_FineTune/mo
```

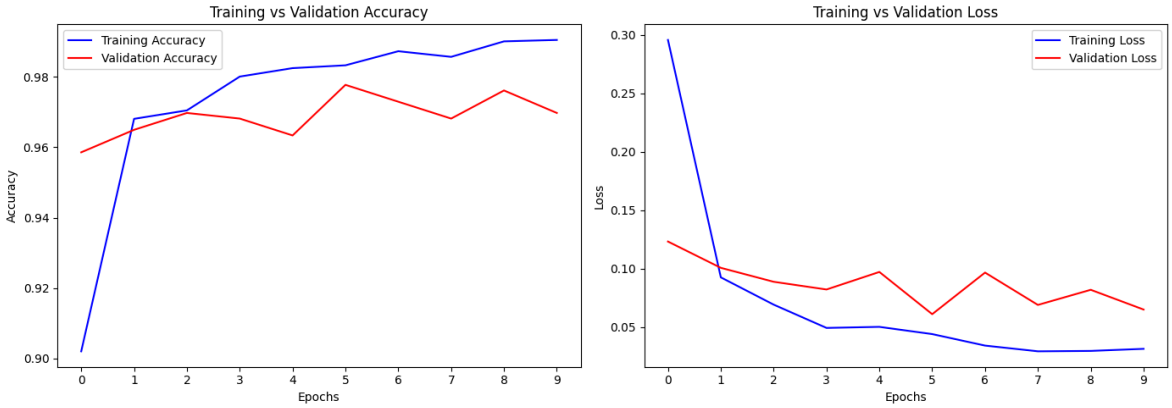
```
Created directory: /content/ninjacart_models/model10_Resent_GAP_DP_FineTune
```

In [ ]:
```
resnet50_model_finetune = tf.keras.models.load_model("/content/ninjacart_models/
loss, acc = resnet50_model_finetune.evaluate(test_data, verbose=2)
print('Before FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
11/11 - 12s - 1s/step - accuracy: 0.9288 - loss: 0.3587
Before FineTuning
Restored model, accuracy: 92.88%
```

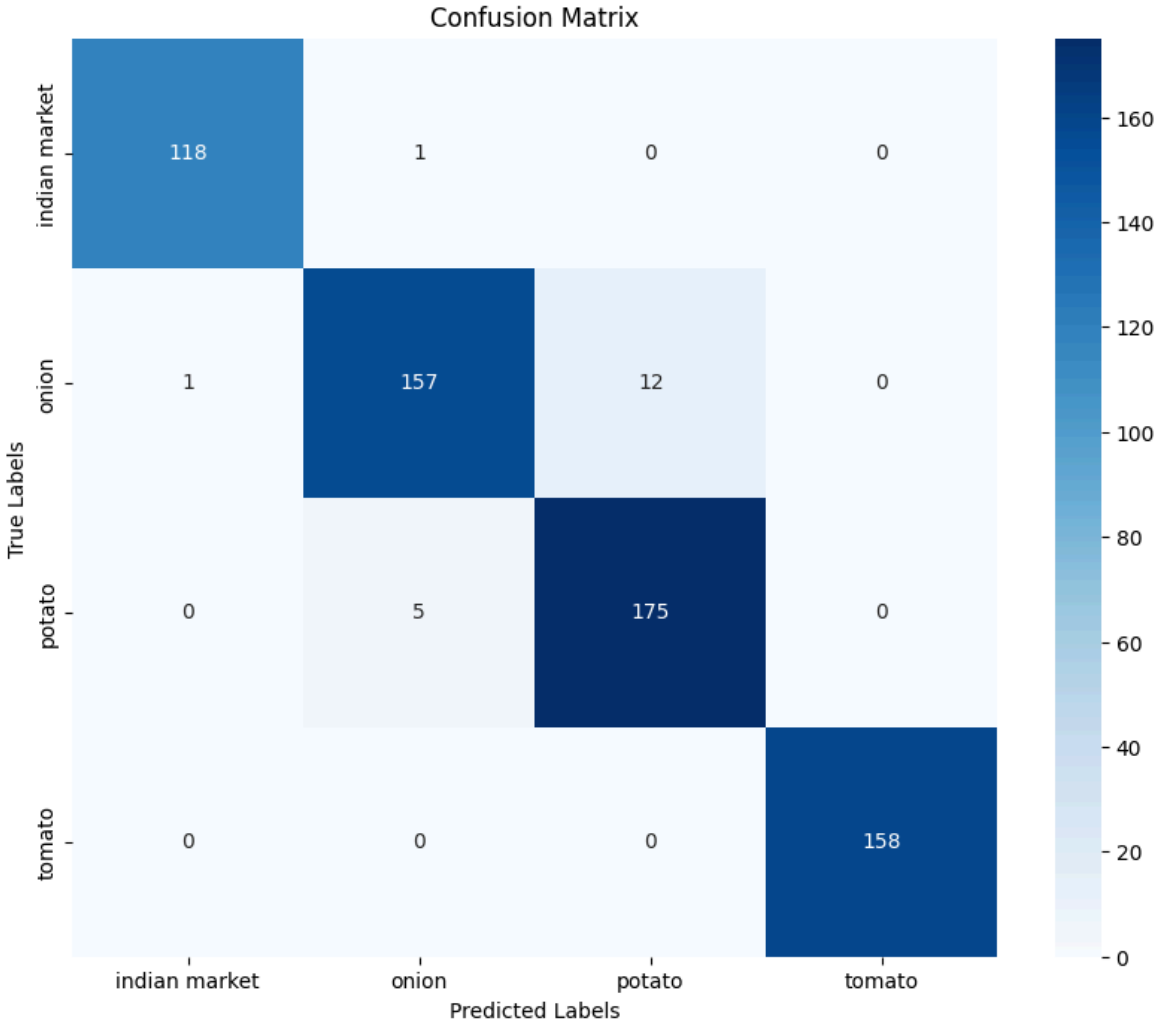
In [ ]:
```
plot_training_history(before_fine_tuning)
```

```
History keys: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```




In [ ]:
```
class_names = list(val_data.class_indices.keys())
plot_confusion_matrix(resnet50_model, val_data, class_names)
```

**20/20** ━━━━━━━━━━━━━━━━━━━━ **12s** 399ms/step




Classification Report:

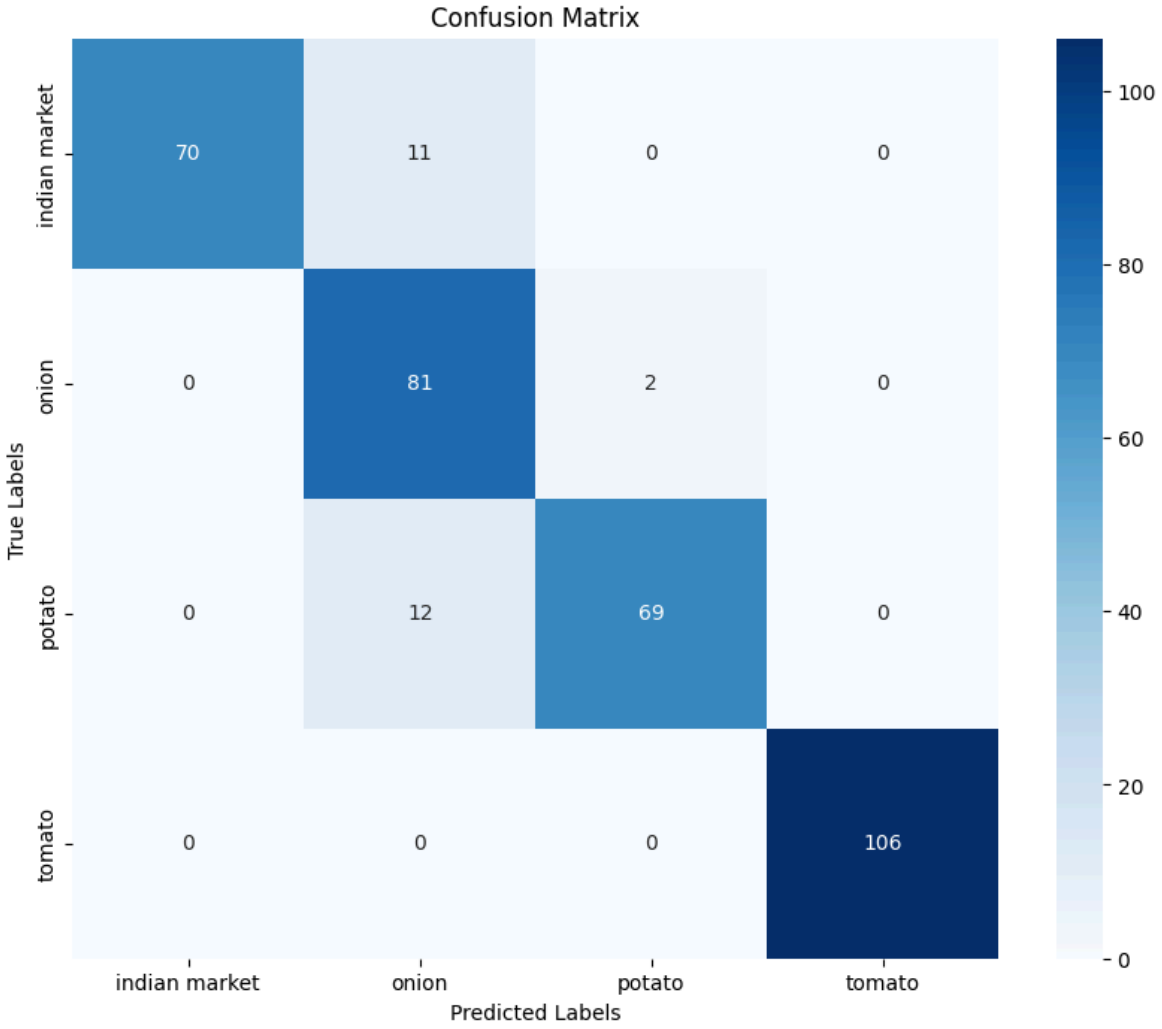
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 0.99 | 0.99 | 0.99 | 119 |
| onion | 0.96 | 0.92 | 0.94 | 170 |
| potato | 0.94 | 0.97 | 0.95 | 180 |
| tomato | 1.00 | 1.00 | 1.00 | 158 |
| accuracy | | | 0.97 | 627 |
| macro avg | 0.97 | 0.97 | 0.97 | 627 |
| weighted avg | 0.97 | 0.97 | 0.97 | 627 |

In [ ]:
```
inception_model_finetune = tf.keras.models.load_model("/content/ninjacart_models
loss, acc = inception_model_finetune.evaluate(test_data, verbose=2)
print('Before FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
11/11 - 9s - 830ms/step - accuracy: 0.9288 - loss: 0.3587
Before FineTuning
Restored model, accuracy: 92.88%
```

In [ ]:
```
class_names = list(test_data.class_indices.keys())
plot_confusion_matrix(resnet50_model, test_data, class_names)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 3s 322ms/step
```




Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 1.00 | 0.86 | 0.93 | 81 |
| onion | 0.78 | 0.98 | 0.87 | 83 |
| potato | 0.97 | 0.85 | 0.91 | 81 |
| tomato | 1.00 | 1.00 | 1.00 | 106 |
| accuracy | | | 0.93 | 351 |
| macro avg | 0.94 | 0.92 | 0.93 | 351 |
| weighted avg | 0.94 | 0.93 | 0.93 | 351 |

# After Finetuning

In [ ]:
```
# Step 6: Unfreeze last few layers of ResNet for fine-tuning
for layer in base_model.layers[-30:]:  # last 30 layers
    if not isinstance(layer, tf.keras.layers.BatchNormalization):
        layer.trainable = True

# Re-compile with a smaller learning rate
resnet50_model.compile(optimizer=Adam(learning_rate=1e-5), loss='categorical_cro

# Step 7: Continue training (fine-tuning)
after_fine_tuning = resnet50_model.fit(train_data, epochs=10, validation_data=va
```

```
Epoch 1/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 48s 413ms/step - accuracy: 0.9917 - loss: 0.0250 - val
_accuracy: 0.9713 - val_loss: 0.0784
Epoch 2/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 20s 216ms/step - accuracy: 0.9970 - loss: 0.0060 - val
_accuracy: 0.9713 - val_loss: 0.0806
Epoch 3/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 18s 224ms/step - accuracy: 0.9979 - loss: 0.0048 - val
_accuracy: 0.9777 - val_loss: 0.0533
Epoch 4/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 20s 211ms/step - accuracy: 0.9971 - loss: 0.0052 - val
_accuracy: 0.9713 - val_loss: 0.0766
Epoch 5/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 215ms/step - accuracy: 0.9972 - loss: 0.0054 - val
_accuracy: 0.9777 - val_loss: 0.0554
Epoch 6/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 214ms/step - accuracy: 0.9974 - loss: 0.0085 - val
_accuracy: 0.9745 - val_loss: 0.0667
Epoch 7/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 18s 225ms/step - accuracy: 0.9998 - loss: 0.0022 - val
_accuracy: 0.9729 - val_loss: 0.0812
Epoch 8/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 215ms/step - accuracy: 0.9991 - loss: 0.0059 - val
_accuracy: 0.9729 - val_loss: 0.0721
Epoch 9/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 219ms/step - accuracy: 0.9997 - loss: 0.0019 - val
_accuracy: 0.9745 - val_loss: 0.0844
Epoch 10/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 17s 218ms/step - accuracy: 0.9986 - loss: 0.0026 - val
_accuracy: 0.9809 - val_loss: 0.0676
```

In [ ]:
```
resnet50_model.summary()
```

```
Model: "functional_12"
```

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_layer_12 (InputLayer) | (None, 224, 224, 3) | 0 | - |
| conv1_pad (ZeroPadding2D) | (None, 230, 230, 3) | 0 | input_layer_12[0… |
| conv1_conv (Conv2D) | (None, 112, 112, 64) | 9,472 | conv1_pad[0][0] |
| conv1_bn (BatchNormalizatio… | (None, 112, 112, 64) | 256 | conv1_conv[0][0] |
| conv1_relu (Activation) | (None, 112, 112, 64) | 0 | conv1_bn[0][0] |
| pool1_pad (ZeroPadding2D) | (None, 114, 114, 64) | 0 | conv1_relu[0][0] |
| pool1_pool (MaxPooling2D) | (None, 56, 56, 64) | 0 | pool1_pad[0][0] |
| conv2_block1_1_conv (Conv2D) | (None, 56, 56, 64) | 4,160 | pool1_pool[0][0] |
| conv2_block1_1_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block1_1_c… |
| conv2_block1_1_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block1_1_b… |
| conv2_block1_2_conv (Conv2D) | (None, 56, 56, 64) | 36,928 | conv2_block1_1_r… |
| conv2_block1_2_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block1_2_c… |
| conv2_block1_2_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block1_2_b… |
| conv2_block1_0_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | pool1_pool[0][0] |
| conv2_block1_3_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | conv2_block1_2_r… |
| conv2_block1_0_bn (BatchNormalizatio… | (None, 56, 56, 256) | 1,024 | conv2_block1_0_c… |
| conv2_block1_3_bn (BatchNormalizatio… | (None, 56, 56, 256) | 1,024 | conv2_block1_3_c… |
| conv2_block1_add (Add) | (None, 56, 56, 256) | 0 | conv2_block1_0_b… conv2_block1_3_b… |
| conv2_block1_out (Activation) | (None, 56, 56, 256) | 0 | conv2_block1_add… |

| | | | |
|---|---|---|---|
| conv2_block2_1_conv (Conv2D) | (None, 56, 56, 64) | 16,448 | conv2_block1_out… |
| conv2_block2_1_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block2_1_c… |
| conv2_block2_1_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block2_1_b… |
| conv2_block2_2_conv (Conv2D) | (None, 56, 56, 64) | 36,928 | conv2_block2_1_r… |
| conv2_block2_2_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block2_2_c… |
| conv2_block2_2_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block2_2_b… |
| conv2_block2_3_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | conv2_block2_2_r… |
| conv2_block2_3_bn (BatchNormalizatio… | (None, 56, 56, 256) | 1,024 | conv2_block2_3_c… |
| conv2_block2_add (Add) | (None, 56, 56, 256) | 0 | conv2_block1_out… conv2_block2_3_b… |
| conv2_block2_out (Activation) | (None, 56, 56, 256) | 0 | conv2_block2_add… |
| conv2_block3_1_conv (Conv2D) | (None, 56, 56, 64) | 16,448 | conv2_block2_out… |
| conv2_block3_1_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block3_1_c… |
| conv2_block3_1_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block3_1_b… |
| conv2_block3_2_conv (Conv2D) | (None, 56, 56, 64) | 36,928 | conv2_block3_1_r… |
| conv2_block3_2_bn (BatchNormalizatio… | (None, 56, 56, 64) | 256 | conv2_block3_2_c… |
| conv2_block3_2_relu (Activation) | (None, 56, 56, 64) | 0 | conv2_block3_2_b… |
| conv2_block3_3_conv (Conv2D) | (None, 56, 56, 256) | 16,640 | conv2_block3_2_r… |
| conv2_block3_3_bn (BatchNormalizatio… | (None, 56, 56, 256) | 1,024 | conv2_block3_3_c… |
| conv2_block3_add (Add) | (None, 56, 56, 256) | 0 | conv2_block2_out… conv2_block3_3_b… |
| conv2_block3_out (Activation) | (None, 56, 56, 256) | 0 | conv2_block3_add… |

| | | | |
|---|---|---|---|
| conv3_block1_1_conv (Conv2D) | (None, 28, 28, 128) | 32,896 | conv2_block3_out… |
| conv3_block1_1_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block1_1_c… |
| conv3_block1_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block1_1_b… |
| conv3_block1_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block1_1_r… |
| conv3_block1_2_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block1_2_c… |
| conv3_block1_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block1_2_b… |
| conv3_block1_0_conv (Conv2D) | (None, 28, 28, 512) | 131,584 | conv2_block3_out… |
| conv3_block1_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block1_2_r… |
| conv3_block1_0_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block1_0_c… |
| conv3_block1_3_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block1_3_c… |
| conv3_block1_add (Add) | (None, 28, 28, 512) | 0 | conv3_block1_0_b… conv3_block1_3_b… |
| conv3_block1_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block1_add… |
| conv3_block2_1_conv (Conv2D) | (None, 28, 28, 128) | 65,664 | conv3_block1_out… |
| conv3_block2_1_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block2_1_c… |
| conv3_block2_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block2_1_b… |
| conv3_block2_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block2_1_r… |
| conv3_block2_2_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block2_2_c… |
| conv3_block2_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block2_2_b… |
| conv3_block2_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block2_2_r… |
| conv3_block2_3_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block2_3_c… |

| | | | |
|---|---|---|---|
| conv3_block2_add (Add) | (None, 28, 28, 512) | 0 | conv3_block1_out… conv3_block2_3_b… |
| conv3_block2_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block2_add… |
| conv3_block3_1_conv (Conv2D) | (None, 28, 28, 128) | 65,664 | conv3_block2_out… |
| conv3_block3_1_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block3_1_c… |
| conv3_block3_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block3_1_b… |
| conv3_block3_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block3_1_r… |
| conv3_block3_2_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block3_2_c… |
| conv3_block3_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block3_2_b… |
| conv3_block3_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block3_2_r… |
| conv3_block3_3_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block3_3_c… |
| conv3_block3_add (Add) | (None, 28, 28, 512) | 0 | conv3_block2_out… conv3_block3_3_b… |
| conv3_block3_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block3_add… |
| conv3_block4_1_conv (Conv2D) | (None, 28, 28, 128) | 65,664 | conv3_block3_out… |
| conv3_block4_1_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block4_1_c… |
| conv3_block4_1_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block4_1_b… |
| conv3_block4_2_conv (Conv2D) | (None, 28, 28, 128) | 147,584 | conv3_block4_1_r… |
| conv3_block4_2_bn (BatchNormalizatio… | (None, 28, 28, 128) | 512 | conv3_block4_2_c… |
| conv3_block4_2_relu (Activation) | (None, 28, 28, 128) | 0 | conv3_block4_2_b… |
| conv3_block4_3_conv (Conv2D) | (None, 28, 28, 512) | 66,048 | conv3_block4_2_r… |
| conv3_block4_3_bn (BatchNormalizatio… | (None, 28, 28, 512) | 2,048 | conv3_block4_3_c… |

| | | | |
|---|---|---|---|
| conv3_block4_add (Add) | (None, 28, 28, 512) | 0 | conv3_block3_out… conv3_block4_3_b… |
| conv3_block4_out (Activation) | (None, 28, 28, 512) | 0 | conv3_block4_add… |
| conv4_block1_1_conv (Conv2D) | (None, 14, 14, 256) | 131,328 | conv3_block4_out… |
| conv4_block1_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block1_1_c… |
| conv4_block1_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block1_1_b… |
| conv4_block1_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block1_1_r… |
| conv4_block1_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block1_2_c… |
| conv4_block1_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block1_2_b… |
| conv4_block1_0_conv (Conv2D) | (None, 14, 14, 1024) | 525,312 | conv3_block4_out… |
| conv4_block1_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block1_2_r… |
| conv4_block1_0_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block1_0_c… |
| conv4_block1_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block1_3_c… |
| conv4_block1_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block1_0_b… conv4_block1_3_b… |
| conv4_block1_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block1_add… |
| conv4_block2_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block1_out… |
| conv4_block2_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block2_1_c… |
| conv4_block2_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block2_1_b… |
| conv4_block2_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block2_1_r… |
| conv4_block2_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block2_2_c… |
| conv4_block2_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block2_2_b… |

| | | | |
|---|---|---|---|
| conv4_block2_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block2_2_r… |
| conv4_block2_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block2_3_c… |
| conv4_block2_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block1_out… conv4_block2_3_b… |
| conv4_block2_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block2_add… |
| conv4_block3_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block2_out… |
| conv4_block3_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block3_1_c… |
| conv4_block3_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block3_1_b… |
| conv4_block3_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block3_1_r… |
| conv4_block3_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block3_2_c… |
| conv4_block3_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block3_2_b… |
| conv4_block3_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block3_2_r… |
| conv4_block3_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block3_3_c… |
| conv4_block3_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block2_out… conv4_block3_3_b… |
| conv4_block3_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block3_add… |
| conv4_block4_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block3_out… |
| conv4_block4_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block4_1_c… |
| conv4_block4_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block4_1_b… |
| conv4_block4_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block4_1_r… |
| conv4_block4_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block4_2_c… |
| conv4_block4_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block4_2_b… |

| | | | |
|---|---|---|---|
| conv4_block4_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block4_2_r… |
| conv4_block4_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block4_3_c… |
| conv4_block4_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block3_out… conv4_block4_3_b… |
| conv4_block4_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block4_add… |
| conv4_block5_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block4_out… |
| conv4_block5_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block5_1_c… |
| conv4_block5_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block5_1_b… |
| conv4_block5_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block5_1_r… |
| conv4_block5_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block5_2_c… |
| conv4_block5_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block5_2_b… |
| conv4_block5_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block5_2_r… |
| conv4_block5_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block5_3_c… |
| conv4_block5_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block4_out… conv4_block5_3_b… |
| conv4_block5_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block5_add… |
| conv4_block6_1_conv (Conv2D) | (None, 14, 14, 256) | 262,400 | conv4_block5_out… |
| conv4_block6_1_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block6_1_c… |
| conv4_block6_1_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block6_1_b… |
| conv4_block6_2_conv (Conv2D) | (None, 14, 14, 256) | 590,080 | conv4_block6_1_r… |
| conv4_block6_2_bn (BatchNormalizatio… | (None, 14, 14, 256) | 1,024 | conv4_block6_2_c… |
| conv4_block6_2_relu (Activation) | (None, 14, 14, 256) | 0 | conv4_block6_2_b… |

| | | | |
|---|---|---|---|
| conv4_block6_3_conv (Conv2D) | (None, 14, 14, 1024) | 263,168 | conv4_block6_2_r… |
| conv4_block6_3_bn (BatchNormalizatio… | (None, 14, 14, 1024) | 4,096 | conv4_block6_3_c… |
| conv4_block6_add (Add) | (None, 14, 14, 1024) | 0 | conv4_block5_out… conv4_block6_3_b… |
| conv4_block6_out (Activation) | (None, 14, 14, 1024) | 0 | conv4_block6_add… |
| conv5_block1_1_conv (Conv2D) | (None, 7, 7, 512) | 524,800 | conv4_block6_out… |
| conv5_block1_1_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block1_1_c… |
| conv5_block1_1_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block1_1_b… |
| conv5_block1_2_conv (Conv2D) | (None, 7, 7, 512) | 2,359,808 | conv5_block1_1_r… |
| conv5_block1_2_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block1_2_c… |
| conv5_block1_2_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block1_2_b… |
| conv5_block1_0_conv (Conv2D) | (None, 7, 7, 2048) | 2,099,200 | conv4_block6_out… |
| conv5_block1_3_conv (Conv2D) | (None, 7, 7, 2048) | 1,050,624 | conv5_block1_2_r… |
| conv5_block1_0_bn (BatchNormalizatio… | (None, 7, 7, 2048) | 8,192 | conv5_block1_0_c… |
| conv5_block1_3_bn (BatchNormalizatio… | (None, 7, 7, 2048) | 8,192 | conv5_block1_3_c… |
| conv5_block1_add (Add) | (None, 7, 7, 2048) | 0 | conv5_block1_0_b… conv5_block1_3_b… |
| conv5_block1_out (Activation) | (None, 7, 7, 2048) | 0 | conv5_block1_add… |
| conv5_block2_1_conv (Conv2D) | (None, 7, 7, 512) | 1,049,088 | conv5_block1_out… |
| conv5_block2_1_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block2_1_c… |
| conv5_block2_1_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block2_1_b… |
| conv5_block2_2_conv (Conv2D) | (None, 7, 7, 512) | 2,359,808 | conv5_block2_1_r… |

| | | | |
|---|---|---|---|
| conv5_block2_2_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block2_2_c… |
| conv5_block2_2_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block2_2_b… |
| conv5_block2_3_conv (Conv2D) | (None, 7, 7, 2048) | 1,050,624 | conv5_block2_2_r… |
| conv5_block2_3_bn (BatchNormalizatio… | (None, 7, 7, 2048) | 8,192 | conv5_block2_3_c… |
| conv5_block2_add (Add) | (None, 7, 7, 2048) | 0 | conv5_block1_out… conv5_block2_3_b… |
| conv5_block2_out (Activation) | (None, 7, 7, 2048) | 0 | conv5_block2_add… |
| conv5_block3_1_conv (Conv2D) | (None, 7, 7, 512) | 1,049,088 | conv5_block2_out… |
| conv5_block3_1_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block3_1_c… |
| conv5_block3_1_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block3_1_b… |
| conv5_block3_2_conv (Conv2D) | (None, 7, 7, 512) | 2,359,808 | conv5_block3_1_r… |
| conv5_block3_2_bn (BatchNormalizatio… | (None, 7, 7, 512) | 2,048 | conv5_block3_2_c… |
| conv5_block3_2_relu (Activation) | (None, 7, 7, 512) | 0 | conv5_block3_2_b… |
| conv5_block3_3_conv (Conv2D) | (None, 7, 7, 2048) | 1,050,624 | conv5_block3_2_r… |
| conv5_block3_3_bn (BatchNormalizatio… | (None, 7, 7, 2048) | 8,192 | conv5_block3_3_c… |
| conv5_block3_add (Add) | (None, 7, 7, 2048) | 0 | conv5_block2_out… conv5_block3_3_b… |
| conv5_block3_out (Activation) | (None, 7, 7, 2048) | 0 | conv5_block3_add… |
| global_average_poo… (GlobalAveragePool… | (None, 2048) | 0 | conv5_block3_out… |
| dropout_7 (Dropout) | (None, 2048) | 0 | global_average_p… |
| dense_18 (Dense) | (None, 512) | 1,049,088 | dropout_7[0][0] |
| dense_19 (Dense) | (None, 4) | 2,052 | dense_18[0][0] |

**Total params:** 55,598,478 (212.09 MB)

**Trainable params:** 15,479,812 (59.05 MB)

**Non-trainable params:** 9,159,040 (34.94 MB)

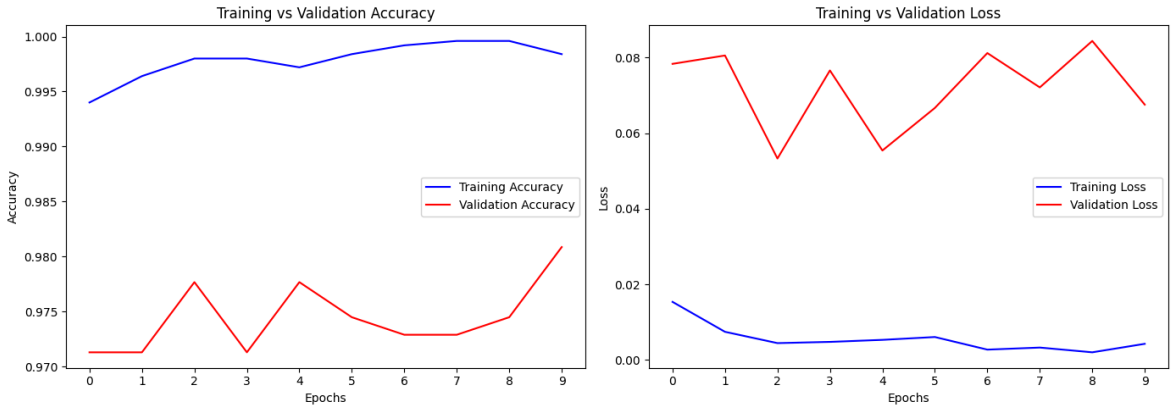
**Optimizer params:** 30,959,626 (118.10 MB)

In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model10_Resent_GAP_DP_
resnet50_model.save("/content/ninjacart_models/model10_Resent_GAP_DP_FineTune/mo
```

```
Directory already exists: /content/ninjacart_models/model10_Resent_GAP_DP_FineTun
e
```

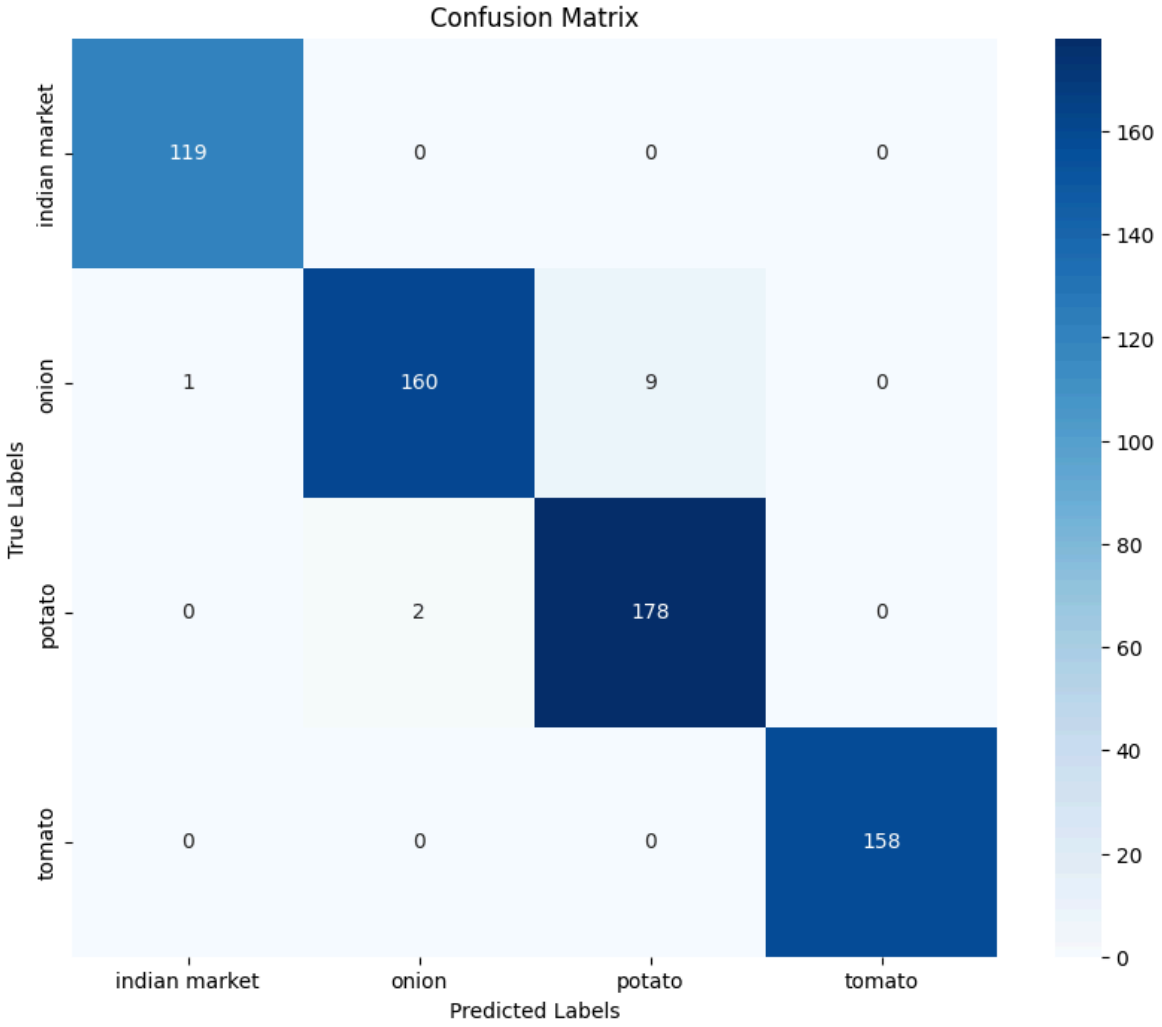
In [ ]:
```
plot_training_history(after_fine_tuning)
```

```
History keys: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```




In [ ]:
```
class_names = list(val_data.class_indices.keys())
plot_confusion_matrix(resnet50_model, val_data, class_names)
```

**20/20** ━━━━━━━━━━━━━━━━━━━━ **10s** 327ms/step




Classification Report:

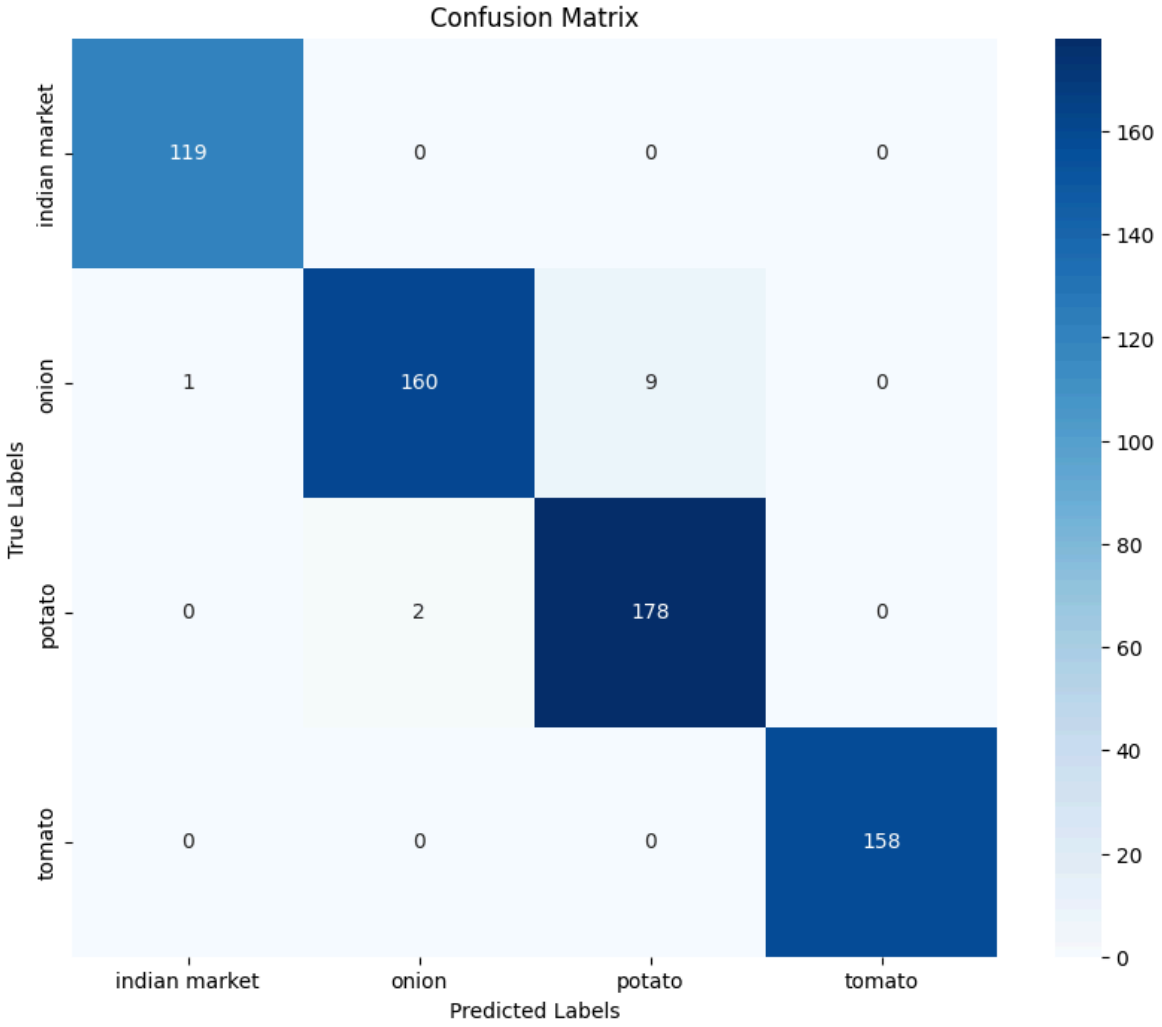
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 0.99 | 1.00 | 1.00 | 119 |
| onion | 0.99 | 0.94 | 0.96 | 170 |
| potato | 0.95 | 0.99 | 0.97 | 180 |
| tomato | 1.00 | 1.00 | 1.00 | 158 |
| accuracy | | | 0.98 | 627 |
| macro avg | 0.98 | 0.98 | 0.98 | 627 |
| weighted avg | 0.98 | 0.98 | 0.98 | 627 |

In [ ]:
```
resnet50_model_finetune = tf.keras.models.load_model("/content/ninjacart_models/
loss, acc = resnet50_model_finetune.evaluate(test_data, verbose=2)
print('After FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
11/11 - 9s - 810ms/step - accuracy: 0.9459 - loss: 0.3997
After FineTuning
Restored model, accuracy: 94.59%
```

In [ ]:
```
class_names = list(val_data.class_indices.keys())
plot_confusion_matrix(resnet50_model, val_data, class_names)
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 4s 182ms/step
```




Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 0.99 | 1.00 | 1.00 | 119 |
| onion | 0.99 | 0.94 | 0.96 | 170 |
| potato | 0.95 | 0.99 | 0.97 | 180 |
| tomato | 1.00 | 1.00 | 1.00 | 158 |
| accuracy | | | 0.98 | 627 |
| macro avg | 0.98 | 0.98 | 0.98 | 627 |
| weighted avg | 0.98 | 0.98 | 0.98 | 627 |

# MobileNet

# Before FineTuning

In [ ]:
```
IMAGE_SIZE = (224, 224)
BATCH_SIZE = 32
NUM_CLASSES = 4
EPOCHS = 10

# 1. Data Preparation
train_gen = ImageDataGenerator(
    rescale=1./255,
```

```
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True,
)

test_gen = ImageDataGenerator(rescale=1./255)
val_gen = ImageDataGenerator(rescale=1./255)

train_generator = train_gen.flow_from_directory(
    '/content/ninjacart_data/train',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

val_generator = val_gen.flow_from_directory(
    '/content/ninjacart_data/validation',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)

test_generator = test_gen.flow_from_directory(
    '/content/ninjacart_data/test',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)

# 2. Load Pretrained MobileNetV2
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224
base_model.trainable = False  # Freeze base layers initially

# 3. Add custom head
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)
x = tf.keras.layers.Dropout(0.5)(x)
x = tf.keras.layers.Dense(256, activation='relu')(x)
output = tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=output)

# 4. Compile and train top layers
model.compile(optimizer=Adam(1e-4), loss='categorical_crossentropy', metrics=['a
hist_before_ft =model.fit(train_generator, validation_data=val_generator, epochs
```

```
Found 2502 images belonging to 4 classes.
Found 627 images belonging to 4 classes.
Found 351 images belonging to 4 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-application
s/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_datas
et_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__ini
t__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multipr
ocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
```

```
Epoch 1/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 62s 636ms/step - accuracy: 0.5426 - loss: 1.1460 - val
_accuracy: 0.8788 - val_loss: 0.3333
Epoch 2/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 40s 511ms/step - accuracy: 0.8636 - loss: 0.4008 - val
_accuracy: 0.9123 - val_loss: 0.2318
Epoch 3/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 41s 517ms/step - accuracy: 0.9000 - loss: 0.2687 - val
_accuracy: 0.9346 - val_loss: 0.1889
Epoch 4/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 40s 513ms/step - accuracy: 0.9110 - loss: 0.2441 - val
_accuracy: 0.9362 - val_loss: 0.1736
Epoch 5/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 41s 515ms/step - accuracy: 0.9375 - loss: 0.1815 - val
_accuracy: 0.9410 - val_loss: 0.1649
Epoch 6/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 41s 522ms/step - accuracy: 0.9417 - loss: 0.1677 - val
_accuracy: 0.9506 - val_loss: 0.1501
Epoch 7/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 41s 523ms/step - accuracy: 0.9538 - loss: 0.1372 - val
_accuracy: 0.9506 - val_loss: 0.1430
Epoch 8/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 41s 524ms/step - accuracy: 0.9565 - loss: 0.1242 - val
_accuracy: 0.9522 - val_loss: 0.1379
Epoch 9/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 81s 512ms/step - accuracy: 0.9539 - loss: 0.1132 - val
_accuracy: 0.9553 - val_loss: 0.1317
Epoch 10/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 43s 539ms/step - accuracy: 0.9446 - loss: 0.1448 - val
_accuracy: 0.9537 - val_loss: 0.1324
```

In [ ]:
```
model.summary()
```

```
Model: "functional_13"
```

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_layer_13 (InputLayer) | (None, 224, 224, 3) | 0 | - |
| Conv1 (Conv2D) | (None, 112, 112, 32) | 864 | input_layer_13[0… |
| bn_Conv1 (BatchNormalizatio… | (None, 112, 112, 32) | 128 | Conv1[0][0] |
| Conv1_relu (ReLU) | (None, 112, 112, 32) | 0 | bn_Conv1[0][0] |
| expanded_conv_dept… (DepthwiseConv2D) | (None, 112, 112, 32) | 288 | Conv1_relu[0][0] |
| expanded_conv_dept… (BatchNormalizatio… | (None, 112, 112, 32) | 128 | expanded_conv_de… |
| expanded_conv_dept… (ReLU) | (None, 112, 112, 32) | 0 | expanded_conv_de… |
| expanded_conv_proj… (Conv2D) | (None, 112, 112, 16) | 512 | expanded_conv_de… |
| expanded_conv_proj… (BatchNormalizatio… | (None, 112, 112, 16) | 64 | expanded_conv_pr… |
| block_1_expand (Conv2D) | (None, 112, 112, 96) | 1,536 | expanded_conv_pr… |
| block_1_expand_BN (BatchNormalizatio… | (None, 112, 112, 96) | 384 | block_1_expand[0… |
| block_1_expand_relu (ReLU) | (None, 112, 112, 96) | 0 | block_1_expand_B… |
| block_1_pad (ZeroPadding2D) | (None, 113, 113, 96) | 0 | block_1_expand_r… |
| block_1_depthwise (DepthwiseConv2D) | (None, 56, 56, 96) | 864 | block_1_pad[0][0] |
| block_1_depthwise_… (BatchNormalizatio… | (None, 56, 56, 96) | 384 | block_1_depthwis… |
| block_1_depthwise_… (ReLU) | (None, 56, 56, 96) | 0 | block_1_depthwis… |
| block_1_project (Conv2D) | (None, 56, 56, 24) | 2,304 | block_1_depthwis… |
| block_1_project_BN (BatchNormalizatio… | (None, 56, 56, 24) | 96 | block_1_project[… |
| block_2_expand (Conv2D) | (None, 56, 56, 144) | 3,456 | block_1_project_… |

| | | | |
|---|---|---|---|
| block_2_expand_BN (BatchNormalizatio… | (None, 56, 56, 144) | 576 | block_2_expand[0… |
| block_2_expand_relu (ReLU) | (None, 56, 56, 144) | 0 | block_2_expand_B… |
| block_2_depthwise (DepthwiseConv2D) | (None, 56, 56, 144) | 1,296 | block_2_expand_r… |
| block_2_depthwise_… (BatchNormalizatio… | (None, 56, 56, 144) | 576 | block_2_depthwis… |
| block_2_depthwise_… (ReLU) | (None, 56, 56, 144) | 0 | block_2_depthwis… |
| block_2_project (Conv2D) | (None, 56, 56, 24) | 3,456 | block_2_depthwis… |
| block_2_project_BN (BatchNormalizatio… | (None, 56, 56, 24) | 96 | block_2_project[… |
| block_2_add (Add) | (None, 56, 56, 24) | 0 | block_1_project_… block_2_project_… |
| block_3_expand (Conv2D) | (None, 56, 56, 144) | 3,456 | block_2_add[0][0] |
| block_3_expand_BN (BatchNormalizatio… | (None, 56, 56, 144) | 576 | block_3_expand[0… |
| block_3_expand_relu (ReLU) | (None, 56, 56, 144) | 0 | block_3_expand_B… |
| block_3_pad (ZeroPadding2D) | (None, 57, 57, 144) | 0 | block_3_expand_r… |
| block_3_depthwise (DepthwiseConv2D) | (None, 28, 28, 144) | 1,296 | block_3_pad[0][0] |
| block_3_depthwise_… (BatchNormalizatio… | (None, 28, 28, 144) | 576 | block_3_depthwis… |
| block_3_depthwise_… (ReLU) | (None, 28, 28, 144) | 0 | block_3_depthwis… |
| block_3_project (Conv2D) | (None, 28, 28, 32) | 4,608 | block_3_depthwis… |
| block_3_project_BN (BatchNormalizatio… | (None, 28, 28, 32) | 128 | block_3_project[… |
| block_4_expand (Conv2D) | (None, 28, 28, 192) | 6,144 | block_3_project_… |
| block_4_expand_BN (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_4_expand[0… |
| block_4_expand_relu (ReLU) | (None, 28, 28, 192) | 0 | block_4_expand_B… |

| | | | |
|---|---|---|---|
| block_4_depthwise (DepthwiseConv2D) | (None, 28, 28, 192) | 1,728 | block_4_expand_r… |
| block_4_depthwise_… (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_4_depthwis… |
| block_4_depthwise_… (ReLU) | (None, 28, 28, 192) | 0 | block_4_depthwis… |
| block_4_project (Conv2D) | (None, 28, 28, 32) | 6,144 | block_4_depthwis… |
| block_4_project_BN (BatchNormalizatio… | (None, 28, 28, 32) | 128 | block_4_project[… |
| block_4_add (Add) | (None, 28, 28, 32) | 0 | block_3_project_… block_4_project_… |
| block_5_expand (Conv2D) | (None, 28, 28, 192) | 6,144 | block_4_add[0][0] |
| block_5_expand_BN (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_5_expand[0… |
| block_5_expand_relu (ReLU) | (None, 28, 28, 192) | 0 | block_5_expand_B… |
| block_5_depthwise (DepthwiseConv2D) | (None, 28, 28, 192) | 1,728 | block_5_expand_r… |
| block_5_depthwise_… (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_5_depthwis… |
| block_5_depthwise_… (ReLU) | (None, 28, 28, 192) | 0 | block_5_depthwis… |
| block_5_project (Conv2D) | (None, 28, 28, 32) | 6,144 | block_5_depthwis… |
| block_5_project_BN (BatchNormalizatio… | (None, 28, 28, 32) | 128 | block_5_project[… |
| block_5_add (Add) | (None, 28, 28, 32) | 0 | block_4_add[0][0… block_5_project_… |
| block_6_expand (Conv2D) | (None, 28, 28, 192) | 6,144 | block_5_add[0][0] |
| block_6_expand_BN (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_6_expand[0… |
| block_6_expand_relu (ReLU) | (None, 28, 28, 192) | 0 | block_6_expand_B… |
| block_6_pad (ZeroPadding2D) | (None, 29, 29, 192) | 0 | block_6_expand_r… |
| block_6_depthwise (DepthwiseConv2D) | (None, 14, 14, 192) | 1,728 | block_6_pad[0][0] |

| | | | |
|---|---|---|---|
| block_6_depthwise_… (BatchNormalizatio… | (None, 14, 14, 192) | 768 | block_6_depthwis… |
| block_6_depthwise_… (ReLU) | (None, 14, 14, 192) | 0 | block_6_depthwis… |
| block_6_project (Conv2D) | (None, 14, 14, 64) | 12,288 | block_6_depthwis… |
| block_6_project_BN (BatchNormalizatio… | (None, 14, 14, 64) | 256 | block_6_project[… |
| block_7_expand (Conv2D) | (None, 14, 14, 384) | 24,576 | block_6_project_… |
| block_7_expand_BN (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_7_expand[0… |
| block_7_expand_relu (ReLU) | (None, 14, 14, 384) | 0 | block_7_expand_B… |
| block_7_depthwise (DepthwiseConv2D) | (None, 14, 14, 384) | 3,456 | block_7_expand_r… |
| block_7_depthwise_… (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_7_depthwis… |
| block_7_depthwise_… (ReLU) | (None, 14, 14, 384) | 0 | block_7_depthwis… |
| block_7_project (Conv2D) | (None, 14, 14, 64) | 24,576 | block_7_depthwis… |
| block_7_project_BN (BatchNormalizatio… | (None, 14, 14, 64) | 256 | block_7_project[… |
| block_7_add (Add) | (None, 14, 14, 64) | 0 | block_6_project_… block_7_project_… |
| block_8_expand (Conv2D) | (None, 14, 14, 384) | 24,576 | block_7_add[0][0] |
| block_8_expand_BN (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_8_expand[0… |
| block_8_expand_relu (ReLU) | (None, 14, 14, 384) | 0 | block_8_expand_B… |
| block_8_depthwise (DepthwiseConv2D) | (None, 14, 14, 384) | 3,456 | block_8_expand_r… |
| block_8_depthwise_… (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_8_depthwis… |
| block_8_depthwise_… (ReLU) | (None, 14, 14, 384) | 0 | block_8_depthwis… |
| block_8_project (Conv2D) | (None, 14, 14, 64) | 24,576 | block_8_depthwis… |

| | | | |
|---|---|---|---|
| block_8_project_BN (BatchNormalizatio… | (None, 14, 14, 64) | 256 | block_8_project[… |
| block_8_add (Add) | (None, 14, 14, 64) | 0 | block_7_add[0][0… block_8_project_… |
| block_9_expand (Conv2D) | (None, 14, 14, 384) | 24,576 | block_8_add[0][0] |
| block_9_expand_BN (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_9_expand[0… |
| block_9_expand_relu (ReLU) | (None, 14, 14, 384) | 0 | block_9_expand_B… |
| block_9_depthwise (DepthwiseConv2D) | (None, 14, 14, 384) | 3,456 | block_9_expand_r… |
| block_9_depthwise_… (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_9_depthwis… |
| block_9_depthwise_… (ReLU) | (None, 14, 14, 384) | 0 | block_9_depthwis… |
| block_9_project (Conv2D) | (None, 14, 14, 64) | 24,576 | block_9_depthwis… |
| block_9_project_BN (BatchNormalizatio… | (None, 14, 14, 64) | 256 | block_9_project[… |
| block_9_add (Add) | (None, 14, 14, 64) | 0 | block_8_add[0][0… block_9_project_… |
| block_10_expand (Conv2D) | (None, 14, 14, 384) | 24,576 | block_9_add[0][0] |
| block_10_expand_BN (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_10_expand[… |
| block_10_expand_re… (ReLU) | (None, 14, 14, 384) | 0 | block_10_expand_… |
| block_10_depthwise (DepthwiseConv2D) | (None, 14, 14, 384) | 3,456 | block_10_expand_… |
| block_10_depthwise… (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_10_depthwi… |
| block_10_depthwise… (ReLU) | (None, 14, 14, 384) | 0 | block_10_depthwi… |
| block_10_project (Conv2D) | (None, 14, 14, 96) | 36,864 | block_10_depthwi… |
| block_10_project_BN (BatchNormalizatio… | (None, 14, 14, 96) | 384 | block_10_project… |
| block_11_expand (Conv2D) | (None, 14, 14, 576) | 55,296 | block_10_project… |

| | | | |
|---|---|---|---|
| block_11_expand_BN (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_11_expand[… |
| block_11_expand_re… (ReLU) | (None, 14, 14, 576) | 0 | block_11_expand_… |
| block_11_depthwise (DepthwiseConv2D) | (None, 14, 14, 576) | 5,184 | block_11_expand_… |
| block_11_depthwise… (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_11_depthwi… |
| block_11_depthwise… (ReLU) | (None, 14, 14, 576) | 0 | block_11_depthwi… |
| block_11_project (Conv2D) | (None, 14, 14, 96) | 55,296 | block_11_depthwi… |
| block_11_project_BN (BatchNormalizatio… | (None, 14, 14, 96) | 384 | block_11_project… |
| block_11_add (Add) | (None, 14, 14, 96) | 0 | block_10_project… block_11_project… |
| block_12_expand (Conv2D) | (None, 14, 14, 576) | 55,296 | block_11_add[0][… |
| block_12_expand_BN (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_12_expand[… |
| block_12_expand_re… (ReLU) | (None, 14, 14, 576) | 0 | block_12_expand_… |
| block_12_depthwise (DepthwiseConv2D) | (None, 14, 14, 576) | 5,184 | block_12_expand_… |
| block_12_depthwise… (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_12_depthwi… |
| block_12_depthwise… (ReLU) | (None, 14, 14, 576) | 0 | block_12_depthwi… |
| block_12_project (Conv2D) | (None, 14, 14, 96) | 55,296 | block_12_depthwi… |
| block_12_project_BN (BatchNormalizatio… | (None, 14, 14, 96) | 384 | block_12_project… |
| block_12_add (Add) | (None, 14, 14, 96) | 0 | block_11_add[0][… block_12_project… |
| block_13_expand (Conv2D) | (None, 14, 14, 576) | 55,296 | block_12_add[0][… |
| block_13_expand_BN (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_13_expand[… |
| block_13_expand_re… (ReLU) | (None, 14, 14, 576) | 0 | block_13_expand_… |

| | | | |
|---|---|---|---|
| block_13_pad (ZeroPadding2D) | (None, 15, 15, 576) | 0 | block_13_expand_… |
| block_13_depthwise (DepthwiseConv2D) | (None, 7, 7, 576) | 5,184 | block_13_pad[0][… |
| block_13_depthwise… (BatchNormalizatio… | (None, 7, 7, 576) | 2,304 | block_13_depthwi… |
| block_13_depthwise… (ReLU) | (None, 7, 7, 576) | 0 | block_13_depthwi… |
| block_13_project (Conv2D) | (None, 7, 7, 160) | 92,160 | block_13_depthwi… |
| block_13_project_BN (BatchNormalizatio… | (None, 7, 7, 160) | 640 | block_13_project… |
| block_14_expand (Conv2D) | (None, 7, 7, 960) | 153,600 | block_13_project… |
| block_14_expand_BN (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_14_expand[… |
| block_14_expand_re… (ReLU) | (None, 7, 7, 960) | 0 | block_14_expand_… |
| block_14_depthwise (DepthwiseConv2D) | (None, 7, 7, 960) | 8,640 | block_14_expand_… |
| block_14_depthwise… (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_14_depthwi… |
| block_14_depthwise… (ReLU) | (None, 7, 7, 960) | 0 | block_14_depthwi… |
| block_14_project (Conv2D) | (None, 7, 7, 160) | 153,600 | block_14_depthwi… |
| block_14_project_BN (BatchNormalizatio… | (None, 7, 7, 160) | 640 | block_14_project… |
| block_14_add (Add) | (None, 7, 7, 160) | 0 | block_13_project… block_14_project… |
| block_15_expand (Conv2D) | (None, 7, 7, 960) | 153,600 | block_14_add[0][… |
| block_15_expand_BN (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_15_expand[… |
| block_15_expand_re… (ReLU) | (None, 7, 7, 960) | 0 | block_15_expand_… |
| block_15_depthwise (DepthwiseConv2D) | (None, 7, 7, 960) | 8,640 | block_15_expand_… |
| block_15_depthwise… (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_15_depthwi… |

| | | | |
|---|---|---|---|
| block_15_depthwise… (ReLU) | (None, 7, 7, 960) | 0 | block_15_depthwi… |
| block_15_project (Conv2D) | (None, 7, 7, 160) | 153,600 | block_15_depthwi… |
| block_15_project_BN (BatchNormalizatio… | (None, 7, 7, 160) | 640 | block_15_project… |
| block_15_add (Add) | (None, 7, 7, 160) | 0 | block_14_add[0][… block_15_project… |
| block_16_expand (Conv2D) | (None, 7, 7, 960) | 153,600 | block_15_add[0][… |
| block_16_expand_BN (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_16_expand[… |
| block_16_expand_re… (ReLU) | (None, 7, 7, 960) | 0 | block_16_expand_… |
| block_16_depthwise (DepthwiseConv2D) | (None, 7, 7, 960) | 8,640 | block_16_expand_… |
| block_16_depthwise… (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_16_depthwi… |
| block_16_depthwise… (ReLU) | (None, 7, 7, 960) | 0 | block_16_depthwi… |
| block_16_project (Conv2D) | (None, 7, 7, 320) | 307,200 | block_16_depthwi… |
| block_16_project_BN (BatchNormalizatio… | (None, 7, 7, 320) | 1,280 | block_16_project… |
| Conv_1 (Conv2D) | (None, 7, 7, 1280) | 409,600 | block_16_project… |
| Conv_1_bn (BatchNormalizatio… | (None, 7, 7, 1280) | 5,120 | Conv_1[0][0] |
| out_relu (ReLU) | (None, 7, 7, 1280) | 0 | Conv_1_bn[0][0] |
| global_average_poo… (GlobalAveragePool… | (None, 1280) | 0 | out_relu[0][0] |
| dropout_8 (Dropout) | (None, 1280) | 0 | global_average_p… |
| dense_20 (Dense) | (None, 256) | 327,936 | dropout_8[0][0] |
| dense_21 (Dense) | (None, 4) | 1,028 | dense_20[0][0] |

**Total params:** 3,244,878 (12.38 MB)

**Trainable params:** 328,964 (1.25 MB)

**Non-trainable params:** 2,257,984 (8.61 MB)

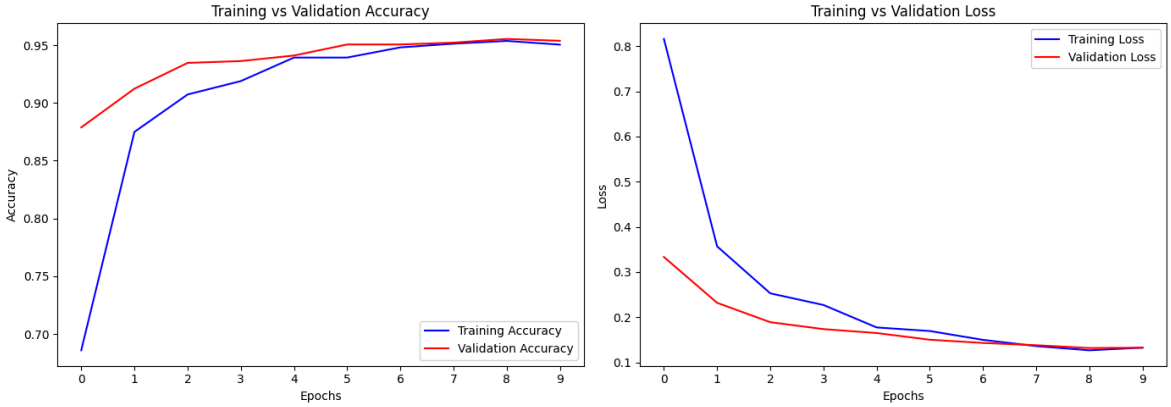
**Optimizer params:** 657,930 (2.51 MB)

In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model11_mobilenet_GAP_
resnet50_model.save("/content/ninjacart_models/model11_mobilenet_GAP_DP_FineTune
```

```
Created directory: /content/ninjacart_models/model11_mobilenet_GAP_DP_FineTune
```

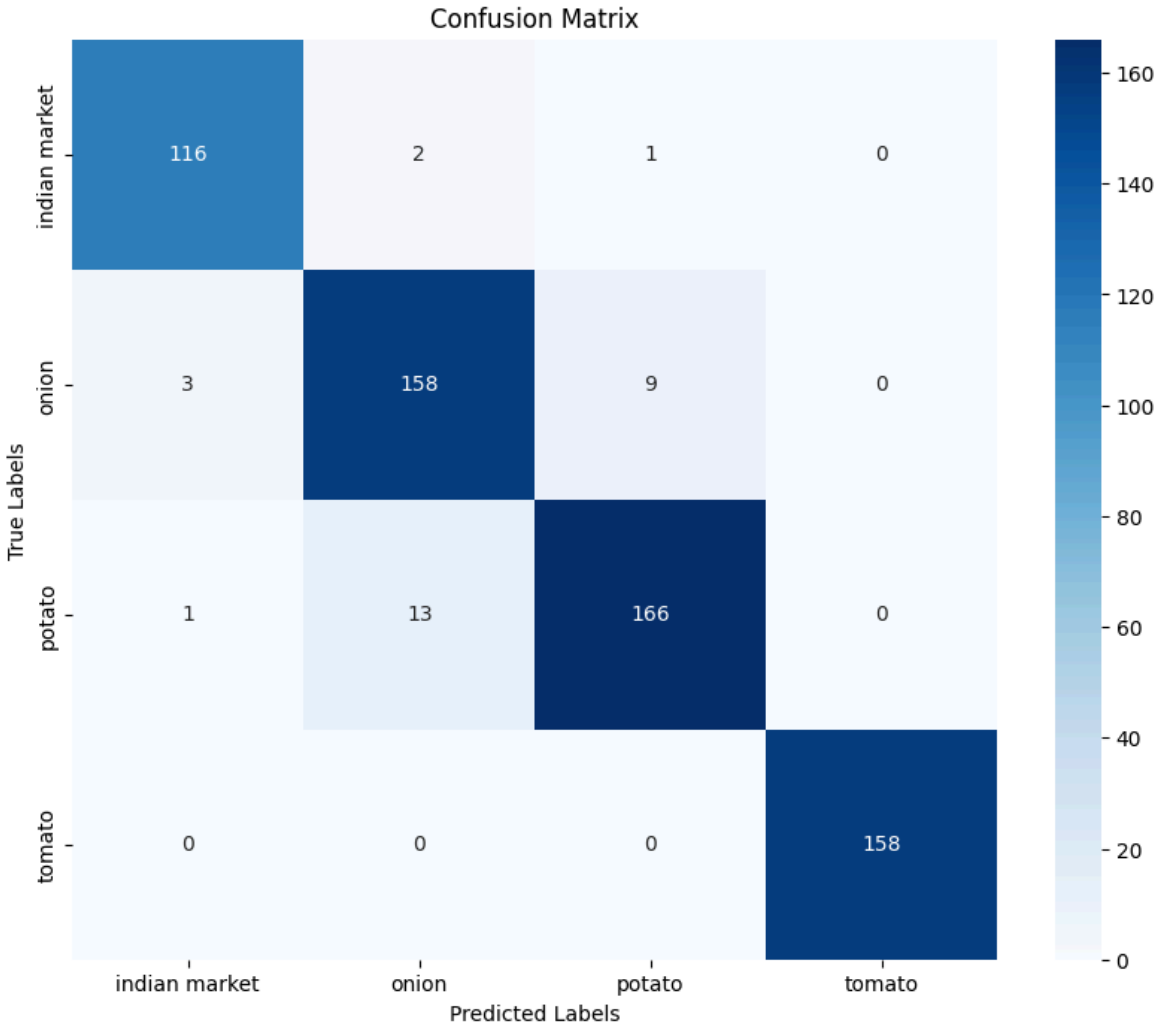
In [ ]:
```
plot_training_history(hist_before_ft)
```

```
History keys: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```




In [ ]:
```
class_names = list(val_generator.class_indices.keys())
plot_confusion_matrix(model, val_generator, class_names)
```

**20/20** ━━━━━━━━━━━━━━━━━━━━ **10s** 334ms/step



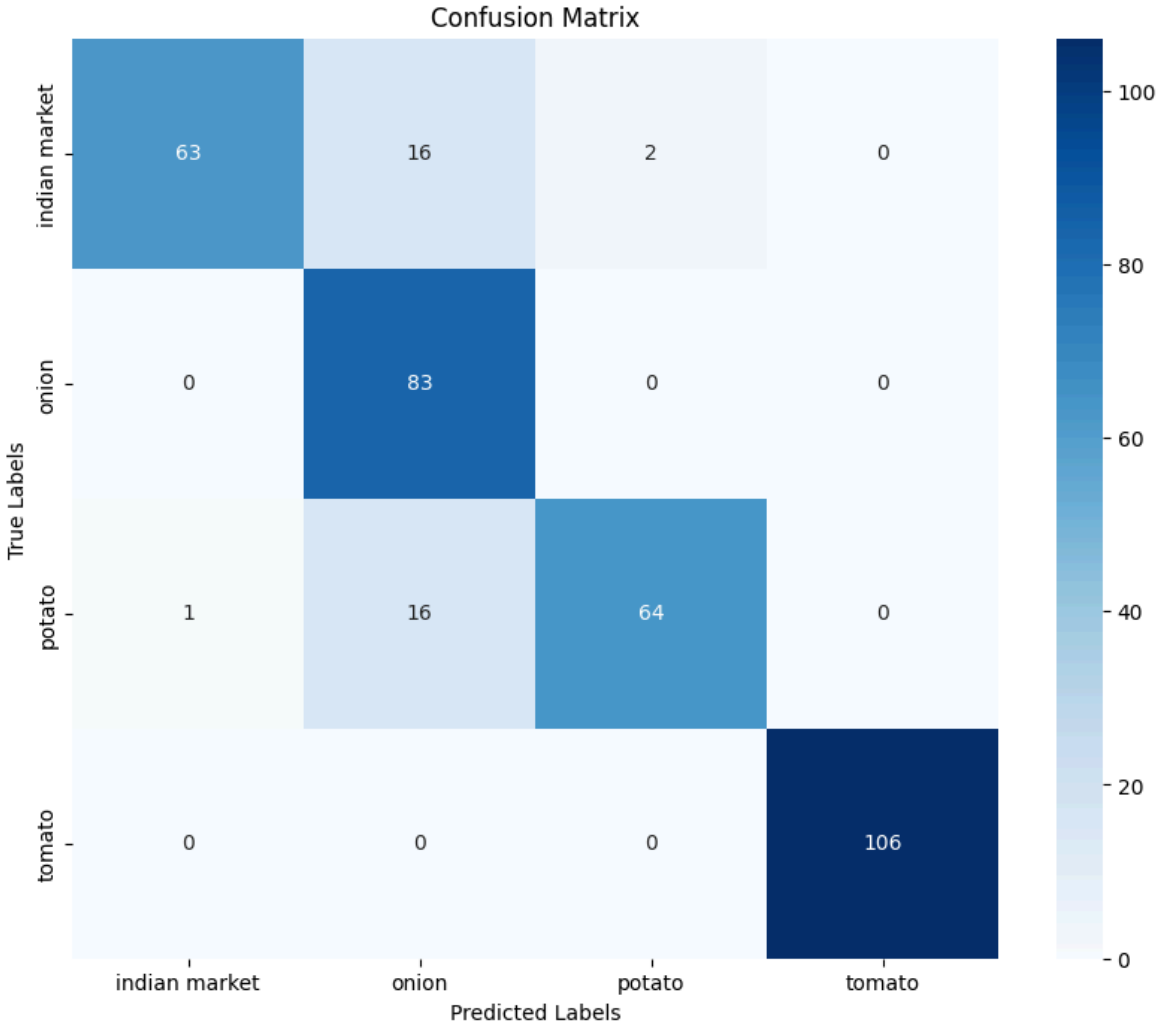
```
Classification Report:

               precision    recall  f1-score   support

indian market       0.97      0.97      0.97       119
        onion       0.91      0.93      0.92       170
       potato       0.94      0.92      0.93       180
       tomato       1.00      1.00      1.00       158

     accuracy                           0.95       627
    macro avg       0.96      0.96      0.96       627
 weighted avg       0.95      0.95      0.95       627
```

In [ ]:
```
mobilenetv2_model_finetune = tf.keras.models.load_model("/content/ninjacart_mode
loss, acc = mobilenetv2_model_finetune.evaluate(test_generator, verbose=2)
print('After FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_datas
et_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__ini
t__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multipr
ocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will
be ignored.
  self._warn_if_super_not_called()
11/11 - 9s - 781ms/step - accuracy: 0.2308 - loss: 9.6986
After FineTuning
Restored model, accuracy: 23.08%
```

In [ ]:
```
class_names = list(val_generator.class_indices.keys())
plot_confusion_matrix(model, test_generator, class_names)
```

```
11/11 ━━━━━━━━━━━━━━━━━━━━ 6s 568ms/step
```




Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 0.98 | 0.78 | 0.87 | 81 |
| onion | 0.72 | 1.00 | 0.84 | 83 |
| potato | 0.97 | 0.79 | 0.87 | 81 |
| tomato | 1.00 | 1.00 | 1.00 | 106 |
| accuracy | | | 0.90 | 351 |
| macro avg | 0.92 | 0.89 | 0.89 | 351 |
| weighted avg | 0.92 | 0.90 | 0.90 | 351 |

# After Finetuning

In [ ]:
```
# 5. Unfreeze some deeper layers for fine-tuning
base_model.trainable = True
for layer in base_model.layers[:100]:
    layer.trainable = False

model.compile(optimizer=Adam(1e-5), loss='categorical_crossentropy', metrics=['a
history = model.fit(train_generator, validation_data=val_generator, epochs=EPOCH
```

```
Epoch 1/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 83s 708ms/step - accuracy: 0.9698 - loss: 0.0861 - val
_accuracy: 0.9601 - val_loss: 0.1119
Epoch 2/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 56s 492ms/step - accuracy: 0.9687 - loss: 0.0919 - val
_accuracy: 0.9601 - val_loss: 0.1110
Epoch 3/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 39s 497ms/step - accuracy: 0.9790 - loss: 0.0671 - val
_accuracy: 0.9601 - val_loss: 0.1106
Epoch 4/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 39s 489ms/step - accuracy: 0.9654 - loss: 0.0738 - val
_accuracy: 0.9617 - val_loss: 0.1093
Epoch 5/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 39s 490ms/step - accuracy: 0.9810 - loss: 0.0590 - val
_accuracy: 0.9617 - val_loss: 0.1109
Epoch 6/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 39s 490ms/step - accuracy: 0.9818 - loss: 0.0557 - val
_accuracy: 0.9633 - val_loss: 0.1093
Epoch 7/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 38s 487ms/step - accuracy: 0.9840 - loss: 0.0536 - val
_accuracy: 0.9569 - val_loss: 0.1091
Epoch 8/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 38s 474ms/step - accuracy: 0.9822 - loss: 0.0575 - val
_accuracy: 0.9601 - val_loss: 0.1079
Epoch 9/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 38s 483ms/step - accuracy: 0.9848 - loss: 0.0489 - val
_accuracy: 0.9617 - val_loss: 0.1132
Epoch 10/10
79/79 ━━━━━━━━━━━━━━━━━━━━ 38s 487ms/step - accuracy: 0.9906 - loss: 0.0386 - val
_accuracy: 0.9617 - val_loss: 0.1106
```

In [ ]:
```
model.summary()
```

```
Model: "functional_13"
```

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_layer_13 (InputLayer) | (None, 224, 224, 3) | 0 | - |
| Conv1 (Conv2D) | (None, 112, 112, 32) | 864 | input_layer_13[0… |
| bn_Conv1 (BatchNormalizatio… | (None, 112, 112, 32) | 128 | Conv1[0][0] |
| Conv1_relu (ReLU) | (None, 112, 112, 32) | 0 | bn_Conv1[0][0] |
| expanded_conv_dept… (DepthwiseConv2D) | (None, 112, 112, 32) | 288 | Conv1_relu[0][0] |
| expanded_conv_dept… (BatchNormalizatio… | (None, 112, 112, 32) | 128 | expanded_conv_de… |
| expanded_conv_dept… (ReLU) | (None, 112, 112, 32) | 0 | expanded_conv_de… |
| expanded_conv_proj… (Conv2D) | (None, 112, 112, 16) | 512 | expanded_conv_de… |
| expanded_conv_proj… (BatchNormalizatio… | (None, 112, 112, 16) | 64 | expanded_conv_pr… |
| block_1_expand (Conv2D) | (None, 112, 112, 96) | 1,536 | expanded_conv_pr… |
| block_1_expand_BN (BatchNormalizatio… | (None, 112, 112, 96) | 384 | block_1_expand[0… |
| block_1_expand_relu (ReLU) | (None, 112, 112, 96) | 0 | block_1_expand_B… |
| block_1_pad (ZeroPadding2D) | (None, 113, 113, 96) | 0 | block_1_expand_r… |
| block_1_depthwise (DepthwiseConv2D) | (None, 56, 56, 96) | 864 | block_1_pad[0][0] |
| block_1_depthwise_… (BatchNormalizatio… | (None, 56, 56, 96) | 384 | block_1_depthwis… |
| block_1_depthwise_… (ReLU) | (None, 56, 56, 96) | 0 | block_1_depthwis… |
| block_1_project (Conv2D) | (None, 56, 56, 24) | 2,304 | block_1_depthwis… |
| block_1_project_BN (BatchNormalizatio… | (None, 56, 56, 24) | 96 | block_1_project[… |
| block_2_expand (Conv2D) | (None, 56, 56, 144) | 3,456 | block_1_project_… |

| | | | |
|---|---|---|---|
| block_2_expand_BN (BatchNormalizatio… | (None, 56, 56, 144) | 576 | block_2_expand[0… |
| block_2_expand_relu (ReLU) | (None, 56, 56, 144) | 0 | block_2_expand_B… |
| block_2_depthwise (DepthwiseConv2D) | (None, 56, 56, 144) | 1,296 | block_2_expand_r… |
| block_2_depthwise_… (BatchNormalizatio… | (None, 56, 56, 144) | 576 | block_2_depthwis… |
| block_2_depthwise_… (ReLU) | (None, 56, 56, 144) | 0 | block_2_depthwis… |
| block_2_project (Conv2D) | (None, 56, 56, 24) | 3,456 | block_2_depthwis… |
| block_2_project_BN (BatchNormalizatio… | (None, 56, 56, 24) | 96 | block_2_project[… |
| block_2_add (Add) | (None, 56, 56, 24) | 0 | block_1_project_… block_2_project_… |
| block_3_expand (Conv2D) | (None, 56, 56, 144) | 3,456 | block_2_add[0][0] |
| block_3_expand_BN (BatchNormalizatio… | (None, 56, 56, 144) | 576 | block_3_expand[0… |
| block_3_expand_relu (ReLU) | (None, 56, 56, 144) | 0 | block_3_expand_B… |
| block_3_pad (ZeroPadding2D) | (None, 57, 57, 144) | 0 | block_3_expand_r… |
| block_3_depthwise (DepthwiseConv2D) | (None, 28, 28, 144) | 1,296 | block_3_pad[0][0] |
| block_3_depthwise_… (BatchNormalizatio… | (None, 28, 28, 144) | 576 | block_3_depthwis… |
| block_3_depthwise_… (ReLU) | (None, 28, 28, 144) | 0 | block_3_depthwis… |
| block_3_project (Conv2D) | (None, 28, 28, 32) | 4,608 | block_3_depthwis… |
| block_3_project_BN (BatchNormalizatio… | (None, 28, 28, 32) | 128 | block_3_project[… |
| block_4_expand (Conv2D) | (None, 28, 28, 192) | 6,144 | block_3_project_… |
| block_4_expand_BN (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_4_expand[0… |
| block_4_expand_relu (ReLU) | (None, 28, 28, 192) | 0 | block_4_expand_B… |

| | | | |
|---|---|---|---|
| block_4_depthwise (DepthwiseConv2D) | (None, 28, 28, 192) | 1,728 | block_4_expand_r… |
| block_4_depthwise_… (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_4_depthwis… |
| block_4_depthwise_… (ReLU) | (None, 28, 28, 192) | 0 | block_4_depthwis… |
| block_4_project (Conv2D) | (None, 28, 28, 32) | 6,144 | block_4_depthwis… |
| block_4_project_BN (BatchNormalizatio… | (None, 28, 28, 32) | 128 | block_4_project[… |
| block_4_add (Add) | (None, 28, 28, 32) | 0 | block_3_project_… block_4_project_… |
| block_5_expand (Conv2D) | (None, 28, 28, 192) | 6,144 | block_4_add[0][0] |
| block_5_expand_BN (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_5_expand[0… |
| block_5_expand_relu (ReLU) | (None, 28, 28, 192) | 0 | block_5_expand_B… |
| block_5_depthwise (DepthwiseConv2D) | (None, 28, 28, 192) | 1,728 | block_5_expand_r… |
| block_5_depthwise_… (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_5_depthwis… |
| block_5_depthwise_… (ReLU) | (None, 28, 28, 192) | 0 | block_5_depthwis… |
| block_5_project (Conv2D) | (None, 28, 28, 32) | 6,144 | block_5_depthwis… |
| block_5_project_BN (BatchNormalizatio… | (None, 28, 28, 32) | 128 | block_5_project[… |
| block_5_add (Add) | (None, 28, 28, 32) | 0 | block_4_add[0][0… block_5_project_… |
| block_6_expand (Conv2D) | (None, 28, 28, 192) | 6,144 | block_5_add[0][0] |
| block_6_expand_BN (BatchNormalizatio… | (None, 28, 28, 192) | 768 | block_6_expand[0… |
| block_6_expand_relu (ReLU) | (None, 28, 28, 192) | 0 | block_6_expand_B… |
| block_6_pad (ZeroPadding2D) | (None, 29, 29, 192) | 0 | block_6_expand_r… |
| block_6_depthwise (DepthwiseConv2D) | (None, 14, 14, 192) | 1,728 | block_6_pad[0][0] |

| | | | |
|---|---|---|---|
| block_6_depthwise_… (BatchNormalizatio… | (None, 14, 14, 192) | 768 | block_6_depthwis… |
| block_6_depthwise_… (ReLU) | (None, 14, 14, 192) | 0 | block_6_depthwis… |
| block_6_project (Conv2D) | (None, 14, 14, 64) | 12,288 | block_6_depthwis… |
| block_6_project_BN (BatchNormalizatio… | (None, 14, 14, 64) | 256 | block_6_project[… |
| block_7_expand (Conv2D) | (None, 14, 14, 384) | 24,576 | block_6_project_… |
| block_7_expand_BN (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_7_expand[0… |
| block_7_expand_relu (ReLU) | (None, 14, 14, 384) | 0 | block_7_expand_B… |
| block_7_depthwise (DepthwiseConv2D) | (None, 14, 14, 384) | 3,456 | block_7_expand_r… |
| block_7_depthwise_… (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_7_depthwis… |
| block_7_depthwise_… (ReLU) | (None, 14, 14, 384) | 0 | block_7_depthwis… |
| block_7_project (Conv2D) | (None, 14, 14, 64) | 24,576 | block_7_depthwis… |
| block_7_project_BN (BatchNormalizatio… | (None, 14, 14, 64) | 256 | block_7_project[… |
| block_7_add (Add) | (None, 14, 14, 64) | 0 | block_6_project_… block_7_project_… |
| block_8_expand (Conv2D) | (None, 14, 14, 384) | 24,576 | block_7_add[0][0] |
| block_8_expand_BN (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_8_expand[0… |
| block_8_expand_relu (ReLU) | (None, 14, 14, 384) | 0 | block_8_expand_B… |
| block_8_depthwise (DepthwiseConv2D) | (None, 14, 14, 384) | 3,456 | block_8_expand_r… |
| block_8_depthwise_… (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_8_depthwis… |
| block_8_depthwise_… (ReLU) | (None, 14, 14, 384) | 0 | block_8_depthwis… |
| block_8_project (Conv2D) | (None, 14, 14, 64) | 24,576 | block_8_depthwis… |

| | | | |
|---|---|---|---|
| block_8_project_BN (BatchNormalizatio… | (None, 14, 14, 64) | 256 | block_8_project[… |
| block_8_add (Add) | (None, 14, 14, 64) | 0 | block_7_add[0][0… block_8_project_… |
| block_9_expand (Conv2D) | (None, 14, 14, 384) | 24,576 | block_8_add[0][0] |
| block_9_expand_BN (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_9_expand[0… |
| block_9_expand_relu (ReLU) | (None, 14, 14, 384) | 0 | block_9_expand_B… |
| block_9_depthwise (DepthwiseConv2D) | (None, 14, 14, 384) | 3,456 | block_9_expand_r… |
| block_9_depthwise_… (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_9_depthwis… |
| block_9_depthwise_… (ReLU) | (None, 14, 14, 384) | 0 | block_9_depthwis… |
| block_9_project (Conv2D) | (None, 14, 14, 64) | 24,576 | block_9_depthwis… |
| block_9_project_BN (BatchNormalizatio… | (None, 14, 14, 64) | 256 | block_9_project[… |
| block_9_add (Add) | (None, 14, 14, 64) | 0 | block_8_add[0][0… block_9_project_… |
| block_10_expand (Conv2D) | (None, 14, 14, 384) | 24,576 | block_9_add[0][0] |
| block_10_expand_BN (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_10_expand[… |
| block_10_expand_re… (ReLU) | (None, 14, 14, 384) | 0 | block_10_expand_… |
| block_10_depthwise (DepthwiseConv2D) | (None, 14, 14, 384) | 3,456 | block_10_expand_… |
| block_10_depthwise… (BatchNormalizatio… | (None, 14, 14, 384) | 1,536 | block_10_depthwi… |
| block_10_depthwise… (ReLU) | (None, 14, 14, 384) | 0 | block_10_depthwi… |
| block_10_project (Conv2D) | (None, 14, 14, 96) | 36,864 | block_10_depthwi… |
| block_10_project_BN (BatchNormalizatio… | (None, 14, 14, 96) | 384 | block_10_project… |
| block_11_expand (Conv2D) | (None, 14, 14, 576) | 55,296 | block_10_project… |

| | | | |
|---|---|---|---|
| block_11_expand_BN (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_11_expand[… |
| block_11_expand_re… (ReLU) | (None, 14, 14, 576) | 0 | block_11_expand_… |
| block_11_depthwise (DepthwiseConv2D) | (None, 14, 14, 576) | 5,184 | block_11_expand_… |
| block_11_depthwise… (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_11_depthwi… |
| block_11_depthwise… (ReLU) | (None, 14, 14, 576) | 0 | block_11_depthwi… |
| block_11_project (Conv2D) | (None, 14, 14, 96) | 55,296 | block_11_depthwi… |
| block_11_project_BN (BatchNormalizatio… | (None, 14, 14, 96) | 384 | block_11_project… |
| block_11_add (Add) | (None, 14, 14, 96) | 0 | block_10_project… block_11_project… |
| block_12_expand (Conv2D) | (None, 14, 14, 576) | 55,296 | block_11_add[0][… |
| block_12_expand_BN (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_12_expand[… |
| block_12_expand_re… (ReLU) | (None, 14, 14, 576) | 0 | block_12_expand_… |
| block_12_depthwise (DepthwiseConv2D) | (None, 14, 14, 576) | 5,184 | block_12_expand_… |
| block_12_depthwise… (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_12_depthwi… |
| block_12_depthwise… (ReLU) | (None, 14, 14, 576) | 0 | block_12_depthwi… |
| block_12_project (Conv2D) | (None, 14, 14, 96) | 55,296 | block_12_depthwi… |
| block_12_project_BN (BatchNormalizatio… | (None, 14, 14, 96) | 384 | block_12_project… |
| block_12_add (Add) | (None, 14, 14, 96) | 0 | block_11_add[0][… block_12_project… |
| block_13_expand (Conv2D) | (None, 14, 14, 576) | 55,296 | block_12_add[0][… |
| block_13_expand_BN (BatchNormalizatio… | (None, 14, 14, 576) | 2,304 | block_13_expand[… |
| block_13_expand_re… (ReLU) | (None, 14, 14, 576) | 0 | block_13_expand_… |

| | | | |
|---|---|---|---|
| block_13_pad (ZeroPadding2D) | (None, 15, 15, 576) | 0 | block_13_expand_… |
| block_13_depthwise (DepthwiseConv2D) | (None, 7, 7, 576) | 5,184 | block_13_pad[0][… |
| block_13_depthwise… (BatchNormalizatio… | (None, 7, 7, 576) | 2,304 | block_13_depthwi… |
| block_13_depthwise… (ReLU) | (None, 7, 7, 576) | 0 | block_13_depthwi… |
| block_13_project (Conv2D) | (None, 7, 7, 160) | 92,160 | block_13_depthwi… |
| block_13_project_BN (BatchNormalizatio… | (None, 7, 7, 160) | 640 | block_13_project… |
| block_14_expand (Conv2D) | (None, 7, 7, 960) | 153,600 | block_13_project… |
| block_14_expand_BN (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_14_expand[… |
| block_14_expand_re… (ReLU) | (None, 7, 7, 960) | 0 | block_14_expand_… |
| block_14_depthwise (DepthwiseConv2D) | (None, 7, 7, 960) | 8,640 | block_14_expand_… |
| block_14_depthwise… (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_14_depthwi… |
| block_14_depthwise… (ReLU) | (None, 7, 7, 960) | 0 | block_14_depthwi… |
| block_14_project (Conv2D) | (None, 7, 7, 160) | 153,600 | block_14_depthwi… |
| block_14_project_BN (BatchNormalizatio… | (None, 7, 7, 160) | 640 | block_14_project… |
| block_14_add (Add) | (None, 7, 7, 160) | 0 | block_13_project… block_14_project… |
| block_15_expand (Conv2D) | (None, 7, 7, 960) | 153,600 | block_14_add[0][… |
| block_15_expand_BN (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_15_expand[… |
| block_15_expand_re… (ReLU) | (None, 7, 7, 960) | 0 | block_15_expand_… |
| block_15_depthwise (DepthwiseConv2D) | (None, 7, 7, 960) | 8,640 | block_15_expand_… |
| block_15_depthwise… (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_15_depthwi… |

| | | | |
|---|---|---|---|
| block_15_depthwise… (ReLU) | (None, 7, 7, 960) | 0 | block_15_depthwi… |
| block_15_project (Conv2D) | (None, 7, 7, 160) | 153,600 | block_15_depthwi… |
| block_15_project_BN (BatchNormalizatio… | (None, 7, 7, 160) | 640 | block_15_project… |
| block_15_add (Add) | (None, 7, 7, 160) | 0 | block_14_add[0][… block_15_project… |
| block_16_expand (Conv2D) | (None, 7, 7, 960) | 153,600 | block_15_add[0][… |
| block_16_expand_BN (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_16_expand[… |
| block_16_expand_re… (ReLU) | (None, 7, 7, 960) | 0 | block_16_expand_… |
| block_16_depthwise (DepthwiseConv2D) | (None, 7, 7, 960) | 8,640 | block_16_expand_… |
| block_16_depthwise… (BatchNormalizatio… | (None, 7, 7, 960) | 3,840 | block_16_depthwi… |
| block_16_depthwise… (ReLU) | (None, 7, 7, 960) | 0 | block_16_depthwi… |
| block_16_project (Conv2D) | (None, 7, 7, 320) | 307,200 | block_16_depthwi… |
| block_16_project_BN (BatchNormalizatio… | (None, 7, 7, 320) | 1,280 | block_16_project… |
| Conv_1 (Conv2D) | (None, 7, 7, 1280) | 409,600 | block_16_project… |
| Conv_1_bn (BatchNormalizatio… | (None, 7, 7, 1280) | 5,120 | Conv_1[0][0] |
| out_relu (ReLU) | (None, 7, 7, 1280) | 0 | Conv_1_bn[0][0] |
| global_average_poo… (GlobalAveragePool… | (None, 1280) | 0 | out_relu[0][0] |
| dropout_8 (Dropout) | (None, 1280) | 0 | global_average_p… |
| dense_20 (Dense) | (None, 256) | 327,936 | dropout_8[0][0] |
| dense_21 (Dense) | (None, 4) | 1,028 | dense_20[0][0] |

**Total params:** 6,967,758 (26.58 MB)

**Trainable params:** 2,190,404 (8.36 MB)

**Non-trainable params:** 396,544 (1.51 MB)

**Optimizer params:** 4,380,810 (16.71 MB)

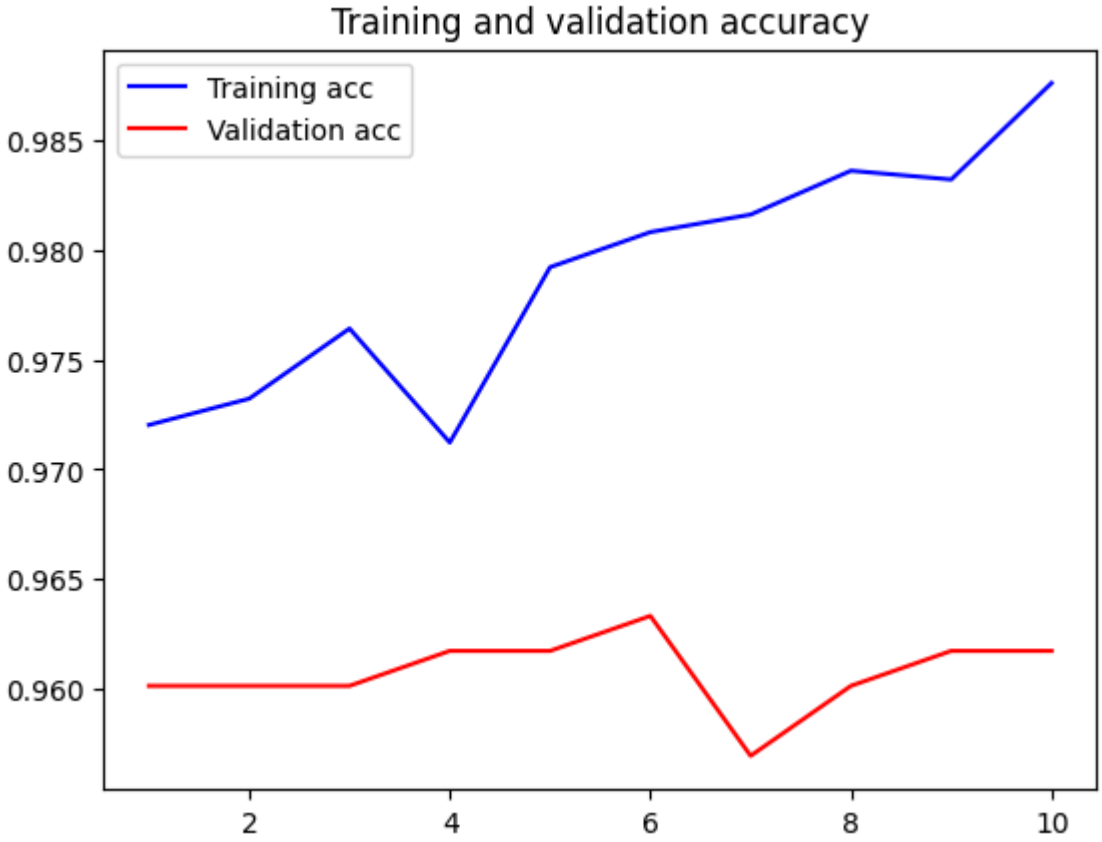
In [ ]:
```
create_directory_if_not_exists('/content/ninjacart_models/model11_mobilenet_GAP_
model.save("/content/ninjacart_models/model11_mobilenet_GAP_DP_FineTune/model11_
```

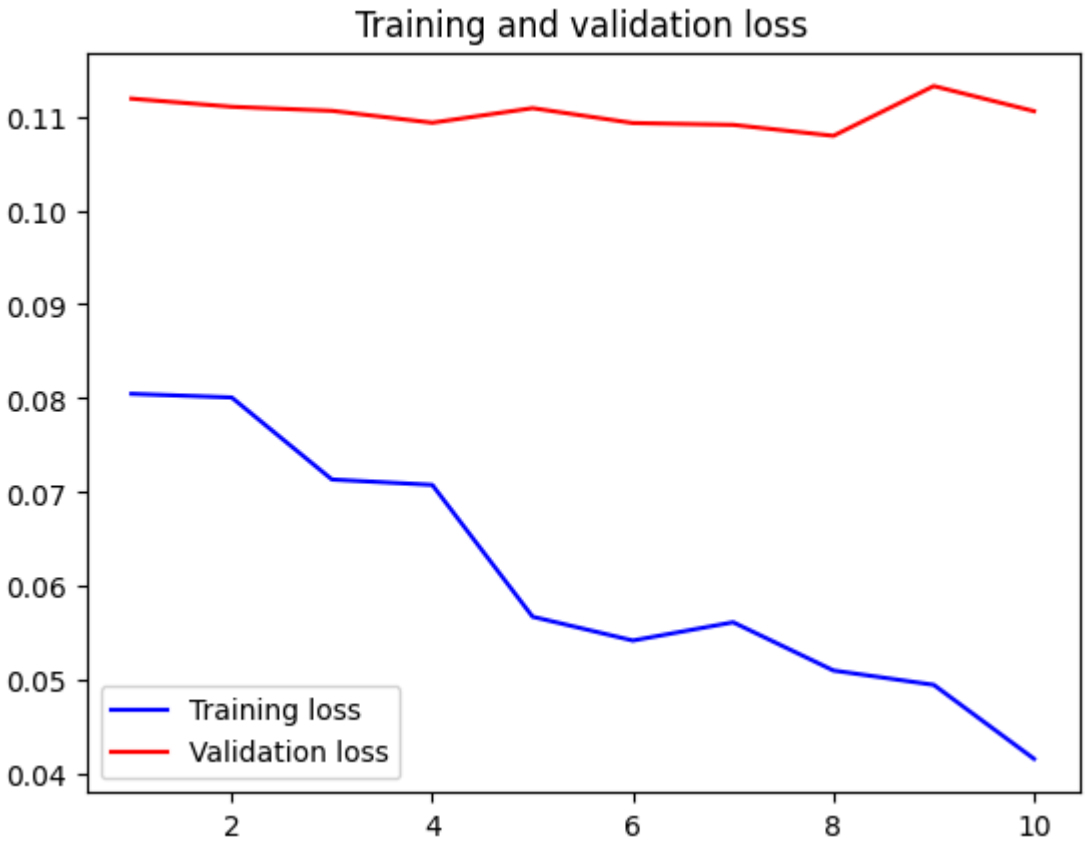
```
Directory already exists: /content/ninjacart_models/model11_mobilenet_GAP_DP_Fine
Tune
```

In [ ]:
```
def plot_training_history(history):
    print("History keys:", history.history.keys())  # This will work now

    acc = history.history.get('accuracy', [])
    val_acc = history.history.get('val_accuracy', [])
    loss = history.history.get('loss', [])
    val_loss = history.history.get('val_loss', [])

    import matplotlib.pyplot as plt

    epochs = range(1, len(acc) + 1)

    plt.plot(epochs, acc, 'b', label='Training acc')
    plt.plot(epochs, val_acc, 'r', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.legend()
    plt.figure()

    plt.plot(epochs, loss, 'b', label='Training loss')
    plt.plot(epochs, val_loss, 'r', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()
    plt.show()
```

In [ ]:
```
plot_training_history(history)
```

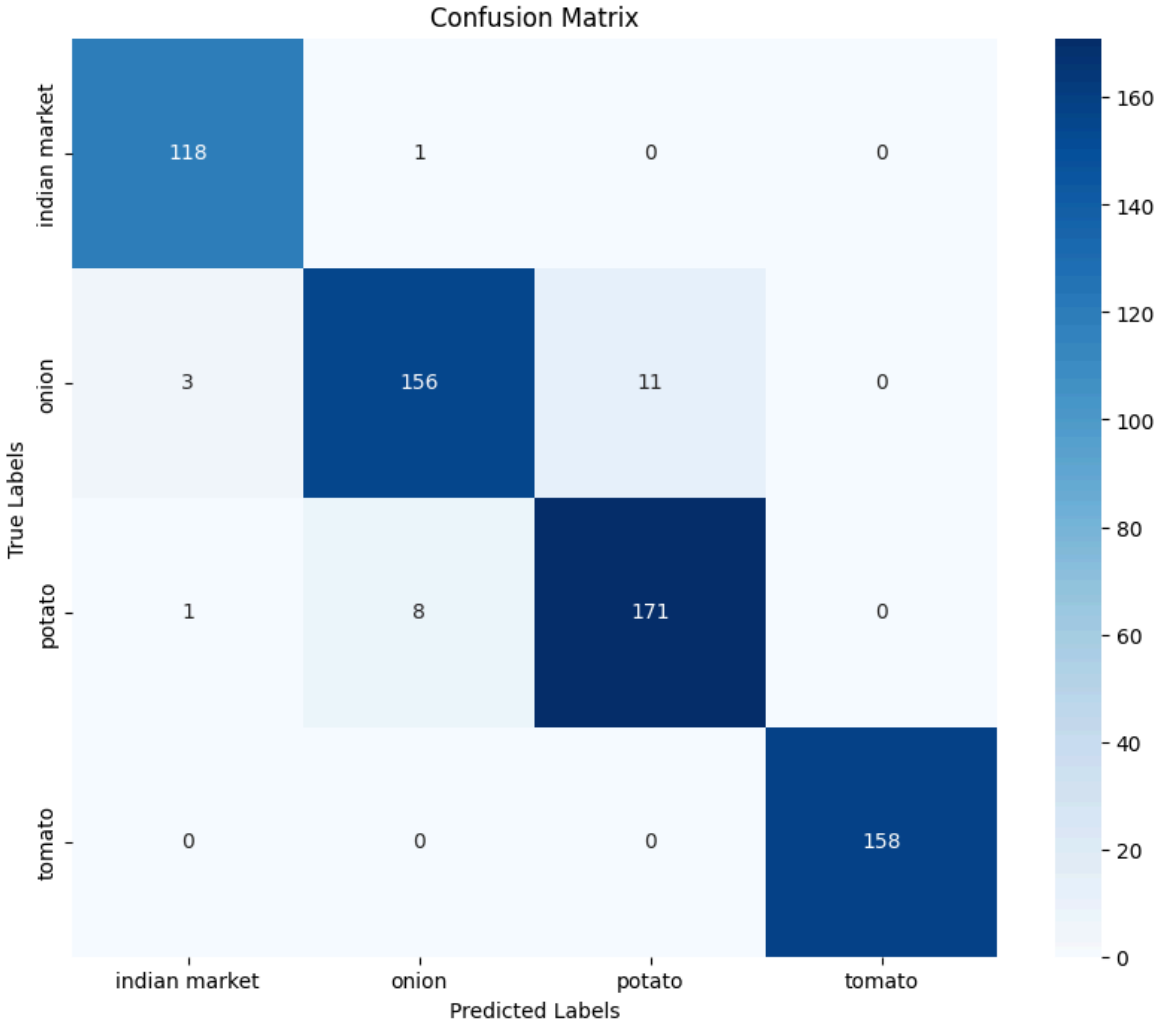
```
History keys: dict_keys(['accuracy', 'loss', 'val_accuracy', 'val_loss'])
```




In [ ]:
```
class_names = list(val_generator.class_indices.keys())
plot_confusion_matrix(model, val_generator, class_names)
```

```
20/20 ━━━━━━━━━━━━━━━━━━━━ 8s 283ms/step
```




Classification Report:

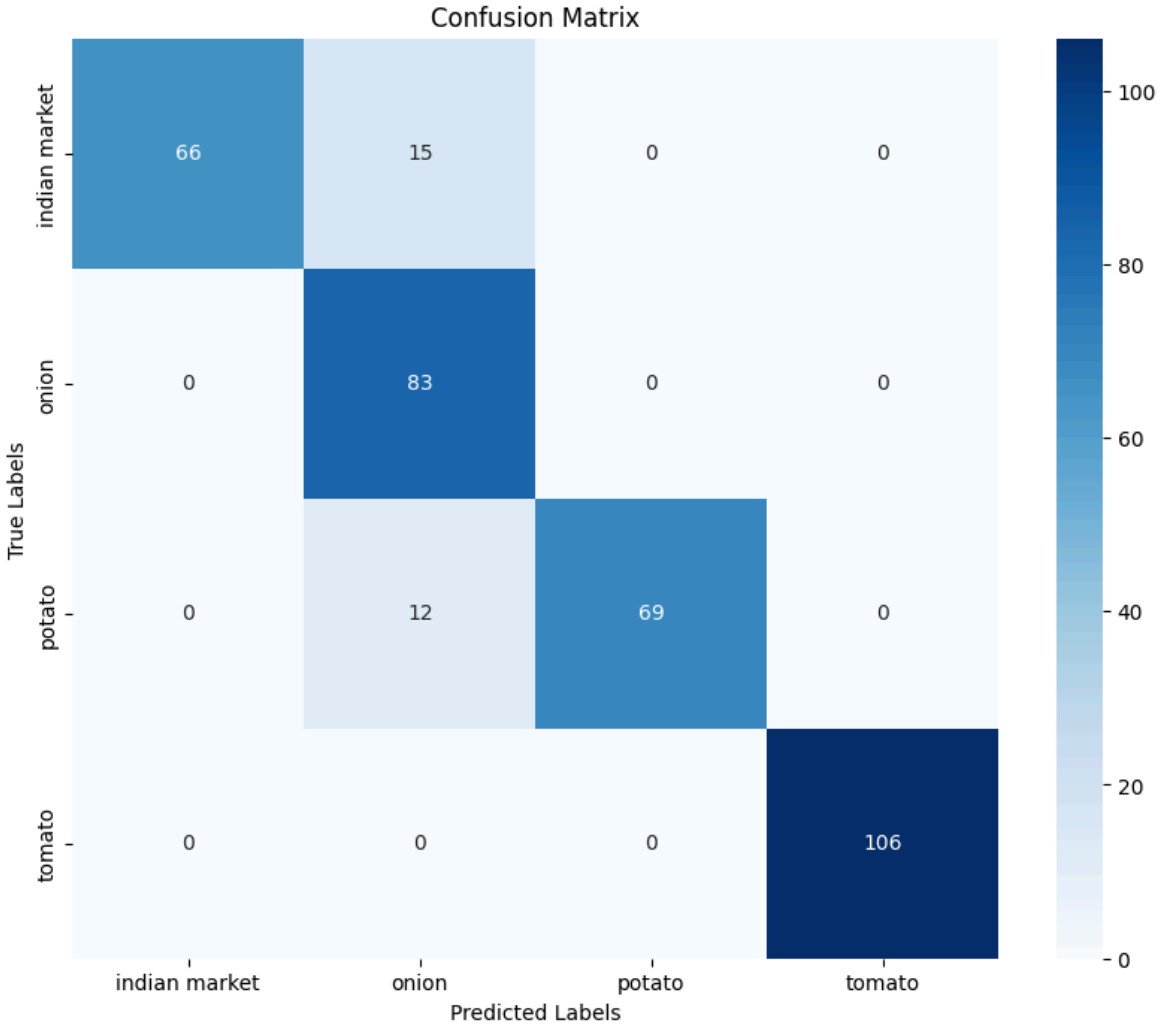
| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 0.97 | 0.99 | 0.98 | 119 |
| onion | 0.95 | 0.92 | 0.93 | 170 |
| potato | 0.94 | 0.95 | 0.94 | 180 |
| tomato | 1.00 | 1.00 | 1.00 | 158 |
| accuracy | | | 0.96 | 627 |
| macro avg | 0.96 | 0.96 | 0.96 | 627 |
| weighted avg | 0.96 | 0.96 | 0.96 | 627 |

In [ ]:
```
mobilenet_model_finetune = tf.keras.models.load_model("/content/ninjacart_models
loss, acc = mobilenet_model_finetune.evaluate(test_generator, verbose=2)
print('After FineTuning')
print("Restored model, accuracy: {:5.2f}%".format(100 * acc))
```

```
11/11 - 8s - 738ms/step - accuracy: 0.9031 - loss: 0.2774
After FineTuning
Restored model, accuracy: 90.31%
```

In [ ]:
```
class_names = list(val_data.class_indices.keys())
plot_confusion_matrix(model, test_generator, class_names)
```

**11/11** ━━━━━━━━━━━━━━━━━━━━ **5s** 524ms/step




Classification Report:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| indian market | 1.00 | 0.81 | 0.90 | 81 |
| onion | 0.75 | 1.00 | 0.86 | 83 |
| potato | 1.00 | 0.85 | 0.92 | 81 |
| tomato | 1.00 | 1.00 | 1.00 | 106 |
| accuracy | | | 0.92 | 351 |
| macro avg | 0.94 | 0.92 | 0.92 | 351 |
| weighted avg | 0.94 | 0.92 | 0.92 | 351 |

# Downloading all the models

In [ ]:

```python
def zip_and_download(folder_path, zip_name="zipped_output"):
    """
    Zips the given folder and creates a download link.

    Args:
        folder_path (str): Path to the folder to zip.
        zip_name (str): Name of the resulting zip file (without .zip).
    """
    zip_path = f"{zip_name}.zip"

    # Remove existing zip if exists
```

```
    if os.path.exists(zip_path):
        os.remove(zip_path)

    # Create zip archive
    shutil.make_archive(zip_name, 'zip', folder_path)
    print(f"✅ Folder zipped successfully: {zip_path}")

    # Display download link (works in Colab/Jupyter)
    display(FileLink(zip_path))
```

In [ ]:
```
zip_and_download('/content/ninjacart_models', 'ninjacart_all_models')
```

```
✅ Folder zipped successfully: ninjacart_all_models.zip
```

ninjacart_all_models.zip

# Summary and Insights of the Custom CNN Models

Built seven custom Convolutional Neural Network (CNN) models designed for classifying RGB images (128x128x3) into four classes: Indian market, Onion, Potato, and Tomato. Each model varies in architecture, hyperparameters, and training techniques, with performance metrics and observations provided. Below is a summary of the models, followed by key insights based on their architecture, performance, and training characteristics.

# Summary and Insights of Custom CNN Models

The dataset includes seven custom Convolutional Neural Network (CNN) models designed for classifying RGB images (128x128x3) into four classes: **Indian market**, **Onion**, **Potato**, and **Tomato**. Each model varies in architecture, hyperparameters, and training techniques, with performance metrics and observations provided. Below is a detailed summary of each model, followed by key insights and recommendations.

---

## Summary of Models

### Model 1: Baseline CNN

- **Architecture**: Conv2D → MaxPooling2D → Flatten → Dense (256) → Dense (4)
- **Parameters**: 16,778,948 (all trainable)
- **Test Accuracy**: 72.08% (lowest)
- **Key Observations**:
  - High parameter count due to Flatten layer, computationally expensive.
  - Significant overfitting (training accuracy > validation accuracy).
  - Confusion between **Potato** ↔ **Onion** and **Indian market** ↔ **others**.
  - Simplest architecture, serving as a baseline.

## Model 2: CNN with Global Average Pooling

- **Architecture**: Conv2D (5 layers) → MaxPooling2D → GlobalAveragePooling2D → Dense (256) → Dense (4)
- **Parameters**: 459,428 (all trainable)
- **Test Accuracy**: 83.48% (significant improvement)
- **Key Observations**:
  - Global Average Pooling (GAP) reduces parameters, improving efficiency.
  - Reduced overfitting (validation: 87.24%, training: 87.13%).
  - Improved performance on all classes, especially **Potato** and **Onion**.
  - Some confusion persists: **Potato** ↔ **Onion**, **Indian market** ↔ **Onion**.

## Model 3: CNN with Global Average Pooling (30 epochs)

- **Architecture**: Same as Model 2, trained for 30 epochs
- **Parameters**: 459,428 (all trainable)
- **Test Accuracy**: 77.78% (lower than Model 2)
- **Key Observations**:
  - Overfitting evident (training: 99.32% >> validation: 90.59%).
  - **Tomato** classified accurately; **Indian market** ↔ **Onion** (41 misclassifications).
  - Performance drop possibly due to fewer epochs or lack of regularization.

## Model 4: CNN with GAP-BN-DP-LR with ReduceLROnPlateau (100 epochs)

- **Architecture**: Conv2D (5 layers) → MaxPooling2D → GAP → Dense (256) → Dense (4) with BatchNormalization (BN) and Dropout (DP)
- **Parameters**: 462,436 (460,932 trainable, 1,504 non-trainable)
- **Test Accuracy**: 88.03%
- **Key Observations**:
  - BN and DP improve generalization (training: 98.80%, validation: 94.10%).
  - ReduceLROnPlateau optimizes learning rate (factor=0.3, patience=5, min_lr=0.00001).
  - **Tomato** accurate; **Tomato** ↔ **Indian market** (106 cases) significant.
  - Reduced **Indian market** ↔ **Onion** (15) and **Potato** ↔ **Onion** (15).

## Model 5: CNN with GAP-BN-DP-LR with ReduceLROnPlateau and L2 Regularization

- **Architecture**: Same as Model 4 with L2 regularization
- **Parameters**: 462,436 (460,932 trainable, 1,504 non-trainable)
- **Test Accuracy**: 91.74% (highest)
- **Key Observations**:
  - L2 regularization reduces overfitting (training: 99.80%, validation: 94.58%).
  - Best test accuracy, indicating strong generalization.
  - **Tomato** ↔ **Indian market** (106 cases) persists.
  - Lowest misclassifications: **Potato** ↔ **Onion** (7), **Indian market** ↔ **Onion** (12).

## Model 6: CNN with GAP-BN-DP-LR-ReduceLROnPlateau-ES (100 epochs)

- **Architecture**: Same as Model 4 with Early Stopping (ES)
- **Parameters**: 462,436 (460,932 trainable, 1,504 non-trainable)
- **Test Accuracy**: 90.31% (slightly lower than Model 5)
- **Key Observations**:
  - ES prevents excessive training, but overfitting persists (training: 99.80%, validation: 93.78%).
  - **Tomato ↔ Indian market** (106 cases) remains a challenge.
  - Slightly worse than Model 5, possibly due to premature stopping.

## Model 7: CNN with GAP-BN-DP-LR-ReduceLROnPlateau-ES-DA (100 epochs)

- **Architecture**: Same as Model 6 with Data Augmentation (DA)
- **Parameters**: 462,436 (460,932 trainable, 1,504 non-trainable)
- **Test Accuracy**: 86.32% (lowest among Models 4–7)
- **Key Observations**:
  - DA reduces overfitting (training: 95.88%, validation: 90.27%), but test accuracy drops.
  - **Tomato ↔ Indian market** (104 cases) persists.
  - Increased **Potato ↔ Onion** misclassifications (24).
  - DA may have introduced noise or needs better tuning.

---

# Key Insights

1. **Architectural Evolution**:
   - **Model 1**: High parameter count (16.78M) due to Flatten, computationally expensive, prone to overfitting.
   - **Models 2–7**: GAP reduces parameters (~462K), enabling deeper architectures (5 Conv2D layers).
   - **Models 4–7**: BN and DP add non-trainable parameters (1,504), improving generalization.
2. **Performance Trends**:
   - Accuracy improves from 72.08% (Model 1) to 91.74% (Model 5) with GAP, BN, DP, and L2.
   - **Model 5** achieves the highest accuracy (91.74%), balancing performance and generalization.
   - **Model 7** (DA) shows reduced accuracy (86.32%), suggesting suboptimal augmentation.
3. **Overfitting**:
   - All models show overfitting (training > validation accuracy).

- **Model 2**: Smallest gap (training: 87.13%, validation: 87.24%).
- **Model 3**: Largest gap (training: 99.32%, validation: 90.59%).
- **Models 4–7**: BN, DP, L2, and DA reduce the gap, but overfitting persists.
- **Model 7**: DA lowers training accuracy (95.88%), improving generalization but reducing test accuracy.

4. **Misclassification Patterns**:

   - **Tomato**: Consistently accurate across all models.
   - **Indian market** ↔ **Onion** and **Potato** ↔ **Onion**: Common issues, improved in Model 5 (7 and 12 cases).
   - **Tomato** ↔ **Indian market**: Significant in Models 4–7 (104–106 cases), indicating visual similarity.

5. **Impact of Techniques**:

   - **GAP (Model 2)**: Reduces parameters, improves accuracy (83.48% vs. 72.08%).
   - **BN and DP (Models 4–7)**: Enhance generalization, boosting accuracy (88–91.74%).
   - **ReduceLROnPlateau**: Optimizes learning rate, improving convergence.
   - **L2 Regularization (Model 5)**: Best balance, achieving 91.74% accuracy.
   - **Early Stopping (Model 6)**: Prevents overtraining but may stop prematurely.
   - **Data Augmentation (Model 7)**: Reduces overfitting but lowers accuracy, needing tuning.

6. **Challenges and Limitations**:

   - Persistent **Indian market** ↔ **Tomato** and **Indian market** ↔ **Onion** misclassifications suggest visual similarities.
   - Overfitting remains despite regularization, especially in Models 3–6.
   - Parameter efficiency improves significantly in Models 2–7 (~462K vs. 16.78M in Model 1).

---

# Recommendations

1. **Address Misclassifications**:

   - Use Grad-CAM or feature visualization to identify features causing **Indian market** ↔ **Tomato** confusion.
   - Collect more diverse **Indian market** data to improve class separability.
   - Apply class-weighted loss to prioritize problematic classes.

2. **Mitigate Overfitting**:

   - Increase Dropout rates or strengthen L2 regularization in Models 4–6.
   - Fine-tune DA in Model 7 (e.g., adjust rotation, flip, zoom).
   - Explore label smoothing or mixup for regularization.

3. **Optimize Training**:

   - Increase epochs for Model 2 to leverage its low overfitting.
   - Adjust ES patience in Model 6 for balanced training.

- Test other optimizers (e.g., RMSprop) or learning rate schedules.

4. **Model Selection**:
   - **Model 5**: Best choice (91.74% accuracy) for deployment due to high performance and generalization.
   - **Model 2**: Suitable for resource-constrained scenarios (83.48% accuracy, fewer epochs).
5. **Future Improvements**:
   - Explore transfer learning (e.g., ResNet, EfficientNet) for better feature extraction.
   - Increase input resolution (>128x128) for finer details, if resources allow.
   - Implement ensemble methods combining Models 4, 5, and 6 for robustness.

---

## Conclusion

The models evolve from a basic CNN (Model 1, 72.08%) to advanced architectures with GAP, BN, DP, L2, ES, and DA (Model 5, 91.74%). **Model 5** is the top performer, but persistent **Indian market** ↔ **Tomato** misclassifications and overfitting require further attention. Targeted data preprocessing, refined augmentation, and advanced architectures (e.g., transfer learning) can enhance performance further.

# Summary and Insights of Pretrained Models and Comparison with Custom Models

The dataset includes four pretrained Convolutional Neural Network (CNN) models—**VGG16**, **InceptionV3**, **ResNet50**, and **MobileNetV2**—used for classifying RGB images into four classes: **Indian market**, **Onion**, **Potato**, and **Tomato**. Each model is evaluated before and after fine-tuning, with details on architecture, parameters, and performance. Additionally, a comparison is provided with the seven custom CNN models previously described, highlighting differences in performance, efficiency, and generalization.

---

## Summary of Pretrained Models

### VGG16

- **Before Fine-tuning (9 epochs)**:
  - **Model Type**: Pretrained VGG16 with Global Average Pooling
  - **Input Shape**: (224, 224, 3) — RGB images
  - **Architecture**: VGG16 base → GlobalAveragePooling2D → Dense(512) → Dropout → Dense(4)
  - **Parameters**: 15,508,814 (264,708 trainable, 14,714,688 non-trainable)

    - **Test Accuracy**: 81.77%
    - **Key Observations**:
        - High non-trainable parameters due to frozen VGG16 base.
        - Minimal overfitting (training: ~0.9, validation: ~0.85).
        - **Tomato** classified accurately; misclassifications: **Indian market ↔ Onion (13)**, **Potato ↔ Onion (23)**, **Tomato ↔ Indian market (105)**.
        - Moderate accuracy due to limited trainable layers.
- **After Fine-tuning (5 epochs, last 4 layers unfrozen)**:
    - **Parameters**: 29,667,662 (7,344,132 trainable, 7,635,264 non-trainable)
    - **Test Accuracy**: 88.03% (improved)
    - **Key Observations**:
        - Fine-tuning last 4 layers increases trainable parameters, boosting accuracy.
        - Minimal overfitting (training: ~0.96, validation: ~0.95).
        - Misclassifications: **Indian market ↔ Onion (10)**, **Potato ↔ Onion (25)**, **Tomato ↔ Indian market (106)**.
        - Improved generalization, but **Tomato ↔ Indian market** persists.

## InceptionV3

- **Before Fine-tuning**:
    - **Model Type**: InceptionV3 with Global Average Pooling and Dense Layers
    - **Input Shape**: (299, 299, 3) — RGB images
    - **Architecture**: Multiple Conv2D, BatchNormalization, Activation, Pooling, Concatenate → GAP → Dropout → Dense(1024) → Dense(4)
    - **Parameters**: 42,456,238 (9,275,588 trainable, 14,629,472 non-trainable)
    - **Test Accuracy**: 92.88%
    - **Key Observations**:
        - High parameter count, but efficient due to Inception modules.
        - Strong performance without fine-tuning, leveraging pretrained weights.
        - No specific misclassification details provided, but high accuracy suggests robust feature extraction.
- **After Fine-tuning (last 50 layers unfrozen)**:
    - **Parameters**: 42,456,238 (9,275,588 trainable, 14,629,472 non-trainable)
    - **Test Accuracy**: 93.16% (slight improvement)
    - **Key Observations**:
        - Fine-tuning last 50 layers slightly improves accuracy.
        - Learning rate (1e-5) ensures stable updates.
        - High accuracy indicates effective transfer learning for the dataset.

## ResNet50

- **Before Fine-tuning**:
    - **Model Type**: ResNet50 with Global Average Pooling and Dense Layers

    - **Input Shape**: (224, 224, 3) — RGB images
    - **Architecture**: Multiple Conv2D, BatchNormalization, Activation, MaxPooling2D → GAP → Dropout → Dense(512) → Dense(4)
    - **Parameters**: 26,741,134 (1,051,140 trainable, 23,587,712 non-trainable)
    - **Test Accuracy**: 92.88%
    - **Key Observations**:
        - Residual connections enable deep architecture with fewer trainable parameters.
        - High accuracy without fine-tuning, indicating strong pretrained features.
        - No specific misclassification details, but performance is robust.
- **After Fine-tuning (last 30 layers unfrozen)**:
    - **Parameters**: 55,598,478 (15,479,812 trainable, 9,159,040 non-trainable)
    - **Test Accuracy**: 94.59% (highest among pretrained models)
    - **Key Observations**:
        - Fine-tuning significantly increases trainable parameters, boosting accuracy.
        - Learning rate (1e-5) and Adam optimizer ensure stable convergence.
        - Best performance, likely due to residual learning and fine-tuned layers.

### MobileNetV2

- **Before Fine-tuning**:
    - **Model Type**: MobileNetV2 with Global Average Pooling and Dense Layers
    - **Input Shape**: (224, 224, 3) — RGB images
    - **Architecture**: Multiple Conv2D, BatchNormalization, ReLU, DepthwiseConv2D → GAP → Dropout → Dense(256) → Dense(4)
    - **Parameters**: 3,244,878 (328,964 trainable, 2,257,984 non-trainable)
    - **Test Accuracy**: 90.00%
    - **Key Observations**:
        - Lightweight model with few parameters, ideal for resource-constrained environments.
        - High accuracy despite minimal trainable parameters.
        - No specific misclassification details, but performance is strong.
- **After Fine-tuning (last 16 layers unfrozen)**:
    - **Parameters**: 6,967,758 (2,190,404 trainable, 396,544 non-trainable)
    - **Test Accuracy**: 92.00% (improved)
    - **Key Observations**:
        - Fine-tuning increases trainable parameters, improving accuracy.
        - Lightweight architecture maintains efficiency.
        - Solid performance, though slightly lower than InceptionV3 and ResNet50.

---

## Insights on Pretrained Models

1. **Performance Trends**:

- **Before Fine-tuning**: All pretrained models achieve high test accuracies (81.77%–92.88%), significantly outperforming the custom Model 1 (72.08%) and most custom models except Model 5 (91.74%).
- **After Fine-tuning**: Fine-tuning improves accuracy across all models:
  - **VGG16**: 81.77% → 88.03%
  - **InceptionV3**: 92.88% → 93.16%
  - **ResNet50**: 92.88% → 94.59% (highest)
  - **MobileNetV2**: 90.00% → 92.00%
- **ResNet50 (fine-tuned)** achieves the highest accuracy (94.59%), followed closely by **InceptionV3 (fine-tuned)** (93.16%).

2. **Overfitting**:

- Pretrained models show minimal overfitting compared to custom models:
  - **VGG16**: Training (~0.9–0.96) ≈ validation (~0.85–0.95).
  - **InceptionV3, ResNet50, MobileNetV2**: No explicit overfitting data, but high test accuracies suggest good generalization.
- Fine-tuning with low learning rates (1e-5) and Dropout helps maintain stability.

3. **Misclassification Patterns (VGG16)**:

- **Tomato** is consistently classified accurately, similar to custom models.
- Persistent issues: **Indian market ↔ Onion (10–13)**, **Potato ↔ Onion (23–25)**, **Tomato ↔ Indian market (105–106)**.
- Fine-tuning VGG16 slightly reduces **Indian market ↔ Onion** misclassifications (13 → 10) but does not address **Tomato ↔ Indian market**.

4. **Parameter Efficiency**:

- **MobileNetV2**: Most efficient (3.2M–6.9M parameters), ideal for low-resource settings.
- **VGG16, InceptionV3, ResNet50**: Higher parameter counts (15.5M–55.6M), but many are non-trainable, leveraging pretrained weights.
- Fine-tuning increases trainable parameters, enhancing performance but requiring more resources.

5. **Architectural Advantages**:

- **VGG16**: Deep, sequential architecture but computationally heavy.
- **InceptionV3**: Efficient Inception modules handle multi-scale features.
- **ResNet50**: Residual connections enable deeper networks with better gradient flow.
- **MobileNetV2**: Depthwise separable convolutions reduce parameters, maintaining performance.

6. **Fine-tuning Impact**:

- Fine-tuning unfreezes later layers (4–50), allowing adaptation to the specific dataset.
- Low learning rates (1e-5) prevent catastrophic forgetting of pretrained weights.
- ResNet50 benefits most from fine-tuning, gaining 1.71% accuracy.

---

# Comparison: Custom Models vs. Pretrained Models

## 1. Performance

- **Custom Models**:
  - Range from 72.08% (Model 1) to 91.74% (Model 5).
  - **Model 5** (91.74%) is competitive with pretrained models but outperformed by fine-tuned **ResNet50 (94.59%)** and **InceptionV3 (93.16%)**.
  - Custom models struggle with persistent misclassifications (**Tomato ↔ Indian market**, 104–106 cases).
- **Pretrained Models**:
  - Higher baseline accuracies (81.77%–92.88% before fine-tuning).
  - Fine-tuned models achieve 88.03%–94.59%, with **ResNet50** leading.
  - VGG16 shows similar misclassification patterns to custom models, but **InceptionV3** and **ResNet50** likely have fewer issues (no specific data provided).

## 2. Overfitting

- **Custom Models**:
  - Significant overfitting, especially in Models 3–6 (training: 95–99%, validation: 90–94%).
  - Model 7 (DA) reduces overfitting but sacrifices accuracy (86.32%).
- **Pretrained Models**:
  - Minimal overfitting due to pretrained weights and regularization (Dropout, low learning rates).
  - VGG16 shows balanced training/validation accuracies (~0.9–0.96 vs. ~0.85–0.95).

## 3. Parameter Efficiency

- **Custom Models**:
  - Model 1: Inefficient (16.78M parameters).
  - Models 2–7: Efficient (~462K parameters) due to GAP, BN, and DP.
- **Pretrained Models**:
  - Higher parameter counts (3.2M–55.6M), but most are non-trainable, reducing training cost.
  - **MobileNetV2**: Most efficient (3.2M–6.9M), comparable to custom Models 2–7 but with better performance.

## 4. Architecture and Complexity

- **Custom Models**:
  - Simpler architectures (5 Conv2D layers in Models 2–7).
  - Limited feature extraction compared to pretrained models.

    - Require extensive tuning (BN, DP, L2, DA, ES) to approach pretrained performance.
- **Pretrained Models**:
    - Complex architectures (VGG16: deep sequential, InceptionV3: multi-scale, ResNet50: residual, MobileNetV2: depthwise separable).
    - Leverage pretrained weights from large datasets (e.g., ImageNet), enabling robust feature extraction with minimal tuning.

## 5. Misclassification Patterns

- **Custom Models**:
    - Persistent **Indian market** ↔ **Tomato** (104–106 cases), **Indian market** ↔ **Onion**, and **Potato** ↔ **Onion**.
    - Model 5 minimizes some errors (e.g., **Potato** ↔ **Onion**: 7).
- **Pretrained Models**:
    - VGG16 shows similar misclassification patterns (**Tomato** ↔ **Indian market**: 105–106).
    - **InceptionV3** and **ResNet50** likely have fewer misclassifications due to higher accuracies and advanced architectures (no specific data).

## 6. Training Effort

- **Custom Models**:
    - Require extensive training (30–100 epochs) and hyperparameter tuning.
    - Model 7 uses DA, but improper tuning reduces accuracy.
- **Pretrained Models**:
    - Fewer epochs needed (5–9 for VGG16, unspecified for others).
    - Fine-tuning with low learning rates is efficient, leveraging pretrained weights.

---

# Recommendations

1. **Model Selection**:
    - **Pretrained Models**: Prefer **fine-tuned ResNet50 (94.59%)** for best performance or **MobileNetV2 (92.00%)** for resource-constrained environments.
    - **Custom Models**: Use **Model 5 (91.74%)** if pretrained models are not feasible, as it balances performance and efficiency.
2. **Addressing Misclassifications**:
    - For both model types, investigate **Indian market** ↔ **Tomato** and **Indian market** ↔ **Onion** using feature visualization (e.g., Grad-CAM).
    - Collect more diverse **Indian market** data or apply class-weighted loss.
3. **Reducing Overfitting**:

- **Custom Models**: Increase regularization (stronger L2, higher Dropout) or refine DA in Model 7.
- **Pretrained Models**: Already well-regularized; consider fine-tuning more layers for VGG16 to match ResNet50/InceptionV3.

4. **Efficiency**:

   - **MobileNetV2** or custom **Model 5** for low-resource settings.
   - Optimize custom models further with lightweight architectures (e.g., depthwise separable convolutions).

5. **Future Improvements**:

   - Combine **ResNet50** and **InceptionV3** in an ensemble for enhanced robustness.
   - Experiment with advanced pretrained models (e.g., EfficientNet) for better performance.
   - Increase input resolution (e.g., 299x299) for custom models, if resources allow.

---

# Conclusion

**Pretrained models** significantly outperform custom models, with **fine-tuned ResNet50 (94.59%)** and **InceptionV3 (93.16%)** leading due to their complex architectures and pretrained weights. **Custom Model 5 (91.74%)** is competitive but requires more tuning and epochs to match pretrained performance. Persistent misclassifications (**Tomato ↔ Indian market**) affect both model types, suggesting dataset-specific challenges. For deployment, **ResNet50 (fine-tuned)** is the top choice for accuracy, while **MobileNetV2** offers efficiency. Custom models are viable for resource-limited scenarios but need further optimization to rival pretrained models.