# Business Case Study: Agri-tech Supply Chain Company

## Problem Description

A prominent agri-tech supply chain company, specializing in fresh produce distribution, is at the forefront of revolutionizing the agricultural supply chain through innovative technology. The company faces a critical challenge in automating its produce identification process. A crucial component of their automation process is the development of sophisticated image classification systems. These systems are designed to distinguish between different types of vegetables and to correctly identify images that do not contain any specific vegetable (categorized as "noise").

To address this challenge, the company has provided a dataset scraped from the web, comprising images of three common vegetables - onions, potatoes, and tomatoes - along with images of Indian market scenes. The dataset is structured as follows:

**Dataset Structure:**

1. **Training set**: A total of 3,135 images divided into four categories:
   * Tomato: 789 images
   * Potato: 898 images
   * Onion: 849 images
   * Indian market (noise): 599 images
2. **Test set**: A total of 351 images divided into the same four categories:
   * Tomato: 106 images
   * Potato: 83 images
   * Onion: 81 images
   * Indian market (noise): 81 images

While a formal data dictionary is not provided, the nature of the dataset implies the following structure:

|  |  |  |
| --- | --- | --- |
| Feature | Description | Data Type |
| Image | The visual content of the produce or market scene | Image file |
| Label | The category of the image (Tomato, Potato, Onion, or Indian market) | Categorical |

**Table 5.1: Data Dictionary for Different Types of Vegetables**

The primary objective is to develop a robust multiclass classifier capable of:

* Accurately identifying and categorizing images of onions, potatoes, and tomatoes.
* Correctly labeling images of market scenes that do not contain a single, identifiable vegetable as "noise".

By developing an accurate image classification model, the company aims to:

1. Automate and accelerate the produce sorting and identification process, reducing human error and labor costs.
2. Enhance inventory management accuracy, leading to better stock control and reduced wastage.
3. Improve quality control measures through consistent and reliable produce identification.
4. Support data-driven decision-making in various aspects of the business, from supply chain optimization to customer communications.

## Business Questions to be Answered from Analysis

To address the challenge of developing an efficient and accurate multiclass classifier for produce identification in the agri-tech supply chain company, the following key questions were addressed:

1. **What is the optimal architecture for a Convolutional Neural Network (CNN) that can accurately classify images of onions, potatoes, tomatoes, and market scenes?**
   * This question aimed to identify the most effective CNN structure for this specific image classification task, considering factors such as the number of layers, types of layers, and hyperparameters.
2. **How does the performance of a custom-built CNN compare to that of a pre-trained model such as MobileNetV3Large when fine-tuned for this specific task?**
   * This analysis helped determine whether transfer learning from an established model offers advantages over a custom-built solution for this particular dataset.
3. **What data augmentation techniques are effective in improving the model's performance and generalization capabilities?**
   * Identifying beneficial augmentation methods to address potential limitations in the dataset size and variety.
4. **How does the class distribution in the training set impact the model's performance?**
   * This question addressed potential biases in the dataset and their effects on classification across all categories.
5. **How effectively can the model distinguish between similar-looking produce (e.g., onions and potatoes) and how can this capability be improved?**
   * This explored the model's fine-grained classification abilities and strategies for enhancement.

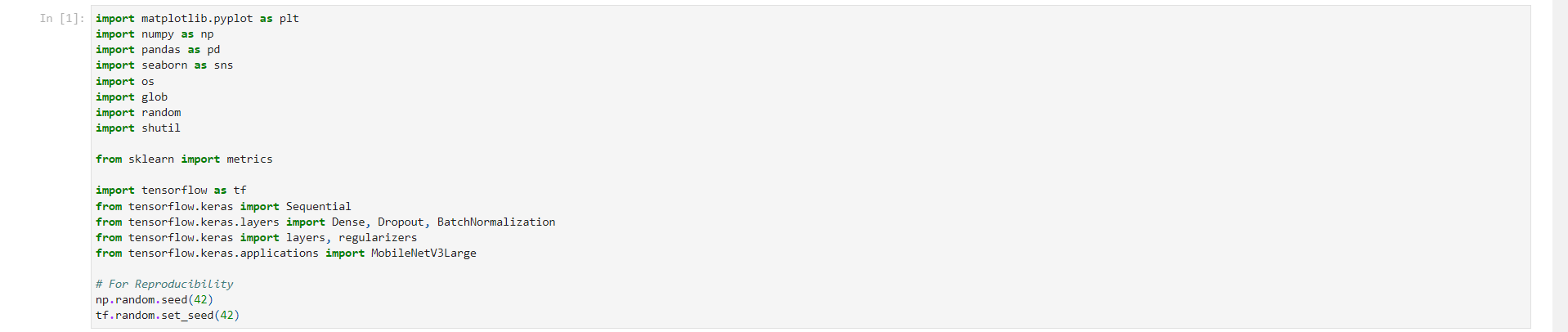
To address these questions, we employed the following analytical approach:

1. Exploratory Data Analysis (EDA) to understand the characteristics and distribution of the image dataset.
2. Development of a baseline CNN model from scratch to establish a performance benchmark.
3. Implementation of an improved CNN architecture with regularization techniques.
4. Implementation and fine-tuning of a pre-trained MobileNetV3Large model for comparison.
5. Experimentation with data augmentation techniques, specifically random cropping and resizing, to enhance model generalization.
6. Analysis of class distribution and its impact on model performance.
7. Evaluation of model performance using metrics such as accuracy and confusion matrices to understand misclassifications.
8. Visualization of model predictions on random test samples to gain insights into classification strengths and weaknesses.

## Analysis

### Analysis Part 1: Data Preprocessing and Initial Exploration

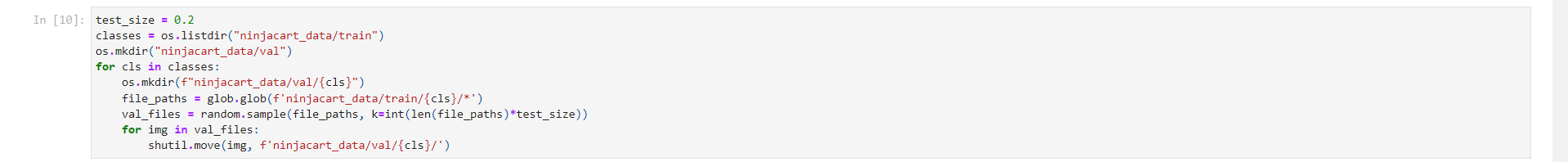
#### Task 1.1: Importing Libraries and Dataset

Figure 5.01: Code for Importing Essential Libraries

Key Insights:

* Essential libraries for data manipulation (pandas, numpy), visualization (matplotlib, seaborn), and machine learning (sklearn, tensorflow) are imported.
* MobileNetV3Large is specifically imported from tensorflow.keras.applications, indicating the intention to use transfer learning.
* Random seeds are set for both numpy and tensorflow to ensure reproducibility of results.

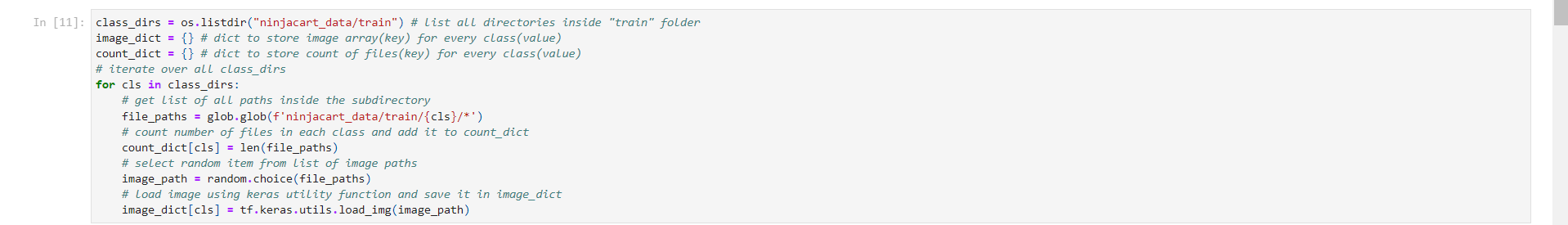
#### Task 1.2: Dividing Train Data into Train and Validation Sets

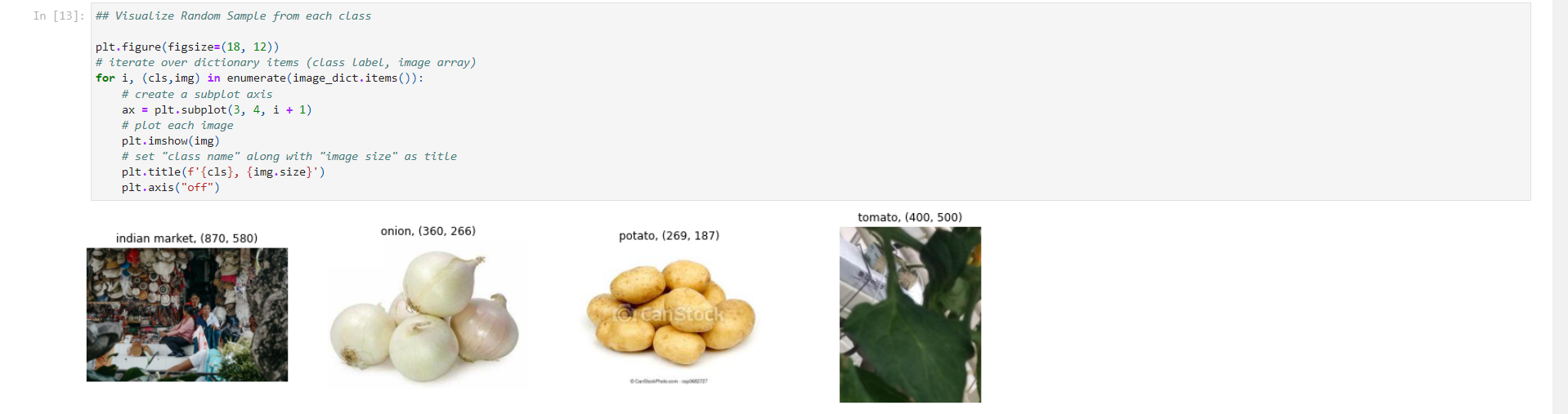
Figure 5.02: Code for Dividing Train Data into Train and Validation Sets

Key Insights:

* The training data is split into training and validation sets with a 80-20 ratio.
* The split is performed while maintaining the original folder structure, ensuring that each class is represented in both sets.
* This approach helps in preserving the class distribution in both training and validation sets.
* The use of random.sample ensures a random selection of images for the validation set, reducing bias.

#### Task 1.3: Exploratory Data Analysis (EDA) - Dataset Overview

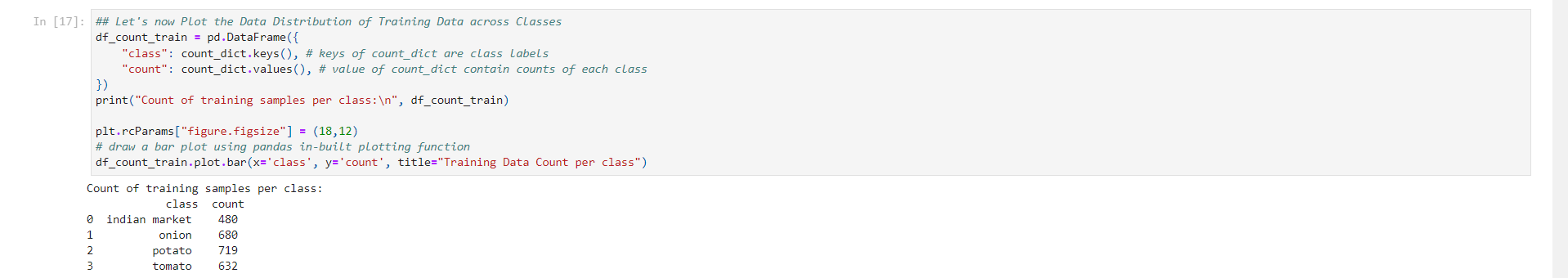
Figure 5.03: Code for Displaying Random Sample Images from Each Class

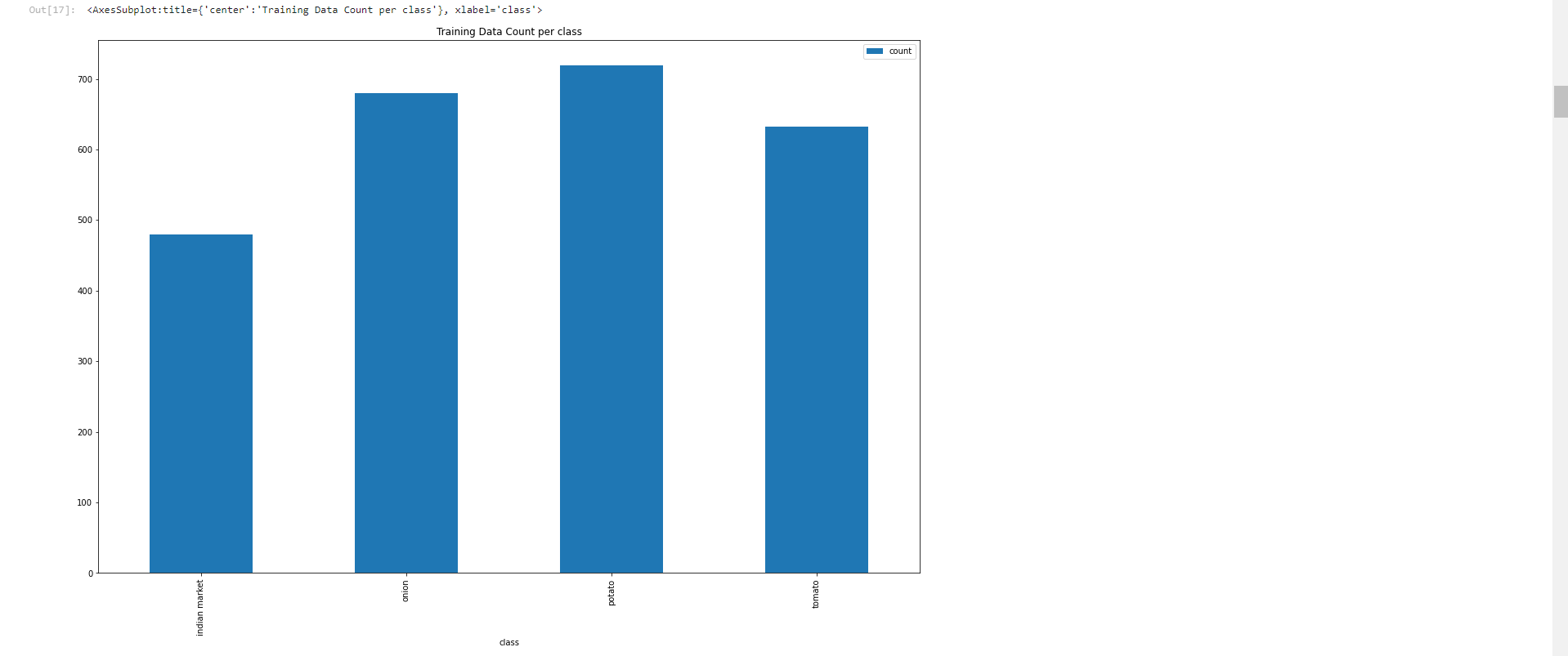
Figure 5.04: Graph of Sample Images from Each Class

Key Insights:

* The dataset consists of four classes: 'indian market', 'onion', 'potato', and 'tomato'.
* A random sample image from each class is displayed, providing a visual overview of the dataset.
* The image sizes vary across classes, which will need to be addressed during preprocessing.
* This visualization helps in understanding the nature of the classification task and potential challenges (e.g., distinguishing between similar-looking vegetables).

#### Task 1.4: Exploratory Data Analysis (EDA) - Class Distribution

Figure 5.05: Code for Analyzing Class Distribution

Figure 5.06: Graph of Class Distribution in the Training Set

Key Insights:

* The class distribution in the training set is as follows:
* indian market: 480 images
  + onion: 680 images
  + potato: 719 images
  + tomato: 632 images
* There is a slight class imbalance, with 'potato' having the highest number of samples and 'indian market' having the least.
* This imbalance is not severe but may need to be addressed during model training to ensure fair representation of all classes.
* The relatively balanced nature of the dataset suggests that simple techniques like class weighting might be sufficient to handle the imbalance.

#### Task 1.5: Loading Data into Keras Dataset

Figure 5.07: Code for Loading Data into Keras Dataset

Key Insights:

* The load\_data function efficiently loads the dataset using TensorFlow's image\_dataset\_from\_directory method.
* The training data is shuffled to ensure random ordering during training, while validation and test data remain unshuffled for consistency.
* The function returns separate datasets for training, validation, and testing, along with class names.
* The output shows that 2511 files were found for training, 624 for validation, and 351 for testing, across 4 classes.

#### Task 1.6: Initial Data Preprocessing

Figure 5.08: Code for Initial Data Preprocessing

Key Insights:

* All images are resized to 224x224 pixels, a standard size for many pre-trained models, ensuring uniform input dimensions.
* Pixel values are rescaled to the range [0, 1] by dividing by 255, which is a common normalization technique for image data.
* The preprocessing is applied consistently across all datasets (train, validation, and test) to maintain data consistency.
* The use of tf.data.AUTOTUNE for parallel calls optimizes the data pipeline for performance.

### Analysis Part 2: Data Modeling and Preparation

#### Task 2.1: Baseline Model Creation

Figure 5.09: Code for Baseline Model Creation

Key Insights:

* A simple CNN model is created as a baseline, with one convolutional layer, one max pooling layer, and two dense layers.
* The model uses ReLU activation for hidden layers and softmax for the output layer, suitable for multi-class classification.
* The input shape is set to (224, 224, 3), matching the preprocessing step.
* The model has a total of 51,381,956 parameters, with the majority in the first dense layer after flattening.
* This baseline model serves as a starting point for comparison with more complex architectures.

#### Task 2.2: Model Compilation and Training

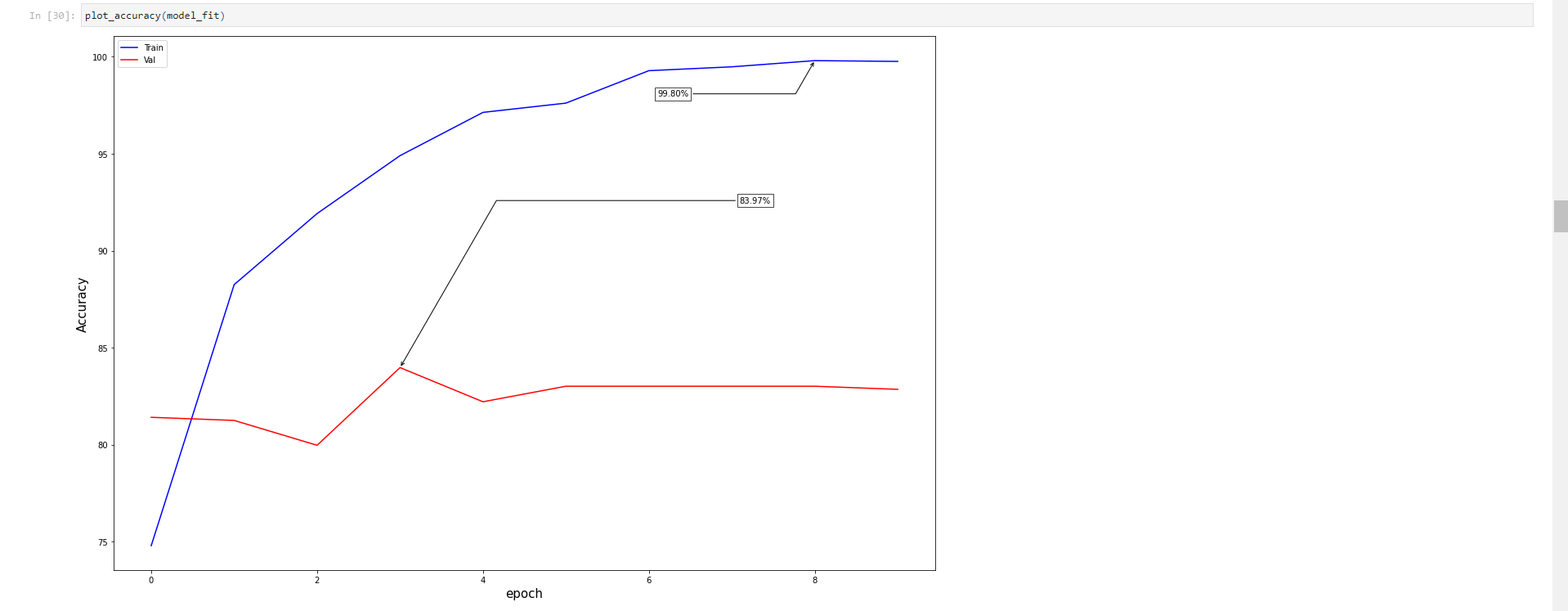
Figure 5.10: Code for Model Compilation and Training

Key Insights:

* The model is compiled with the Adam optimizer and categorical crossentropy loss, suitable for multi-class classification.
* Training is performed for 10 epochs with a ModelCheckpoint callback to save the best model based on validation accuracy.
* The training process shows:
  + Training accuracy improved from 74.79% in the first epoch to 99.76% in the final epoch.
  + Validation accuracy fluctuated, reaching a maximum of 83.97% in the 4th epoch and ending at 82.85%.
  + The large gap between training and validation accuracy suggests overfitting.

#### Task 2.3: Visualizing Training Progress

Figure 5.11: Code for Displaying Training and Validation Accuracy Over Epochs

Figure 5.12: Graph of Training and Validation Accuracy Over Epochs

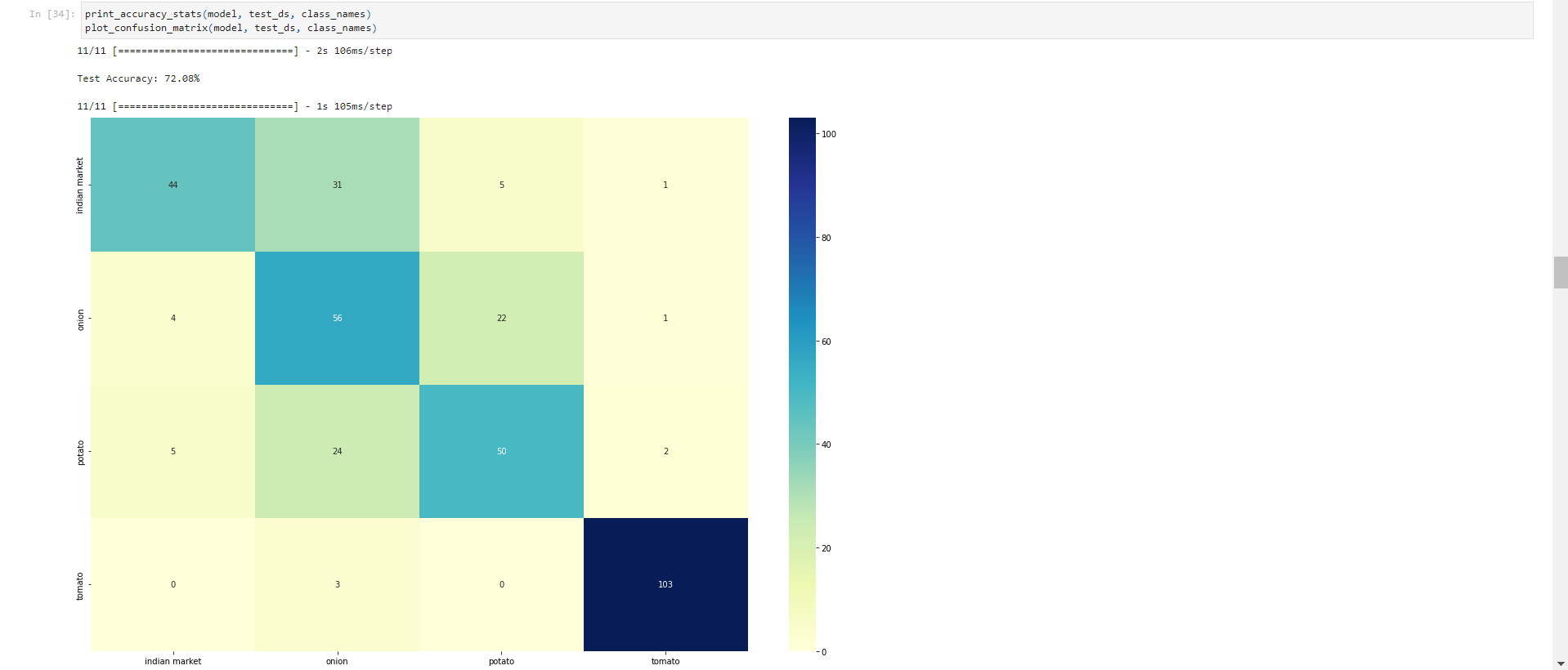
Key Insights:

* The plot clearly shows the divergence between training and validation accuracy over epochs.
* Training accuracy steadily increases to nearly 100%, while validation accuracy plateaus and slightly decreases after the 4th epoch.
* This visualization confirms the overfitting observed in the training logs.
* The model's ability to generalize to unseen data is limited, indicating a need for regularization techniques or a more suitable model architecture.

### Analysis Part 3: Model Evaluation and Improvement

#### Task 3.1: Evaluating Model Performance on Test Set

Figure 5.13: Code for Evaluating Model Performance on Test Set

Figure 5.14: Table of Confusion Matrix for Baseline Model

Key Insights:

* The baseline model achieved a test accuracy of 72.08%.
* The confusion matrix reveals:
  + The model performs best on the 'indian market' class, with few misclassifications.
  + There's significant confusion between 'onion' and 'potato' classes, indicating difficulty in distinguishing these vegetables.
  + 'Tomato' classification is relatively good but shows some misclassifications with 'potato'.
* These results suggest that while the model has learned some distinguishing features, there's substantial room for improvement, especially in differentiating between similar-looking vegetables.

#### Task 3.2: Improving the CNN Model

Figure 5.15: Code for Improved CNN Model Architecture

Key Insights:

* The improved architecture includes several key enhancements:
  + Increased depth with multiple convolutional layers.
  + Use of L2 regularization to prevent overfitting.
  + Batch normalization layers to stabilize learning.
  + Dropout layer (50% rate) for additional regularization.
  + Global Average Pooling to reduce the number of parameters.
* These changes aim to create a more robust model capable of learning complex features while mitigating overfitting.

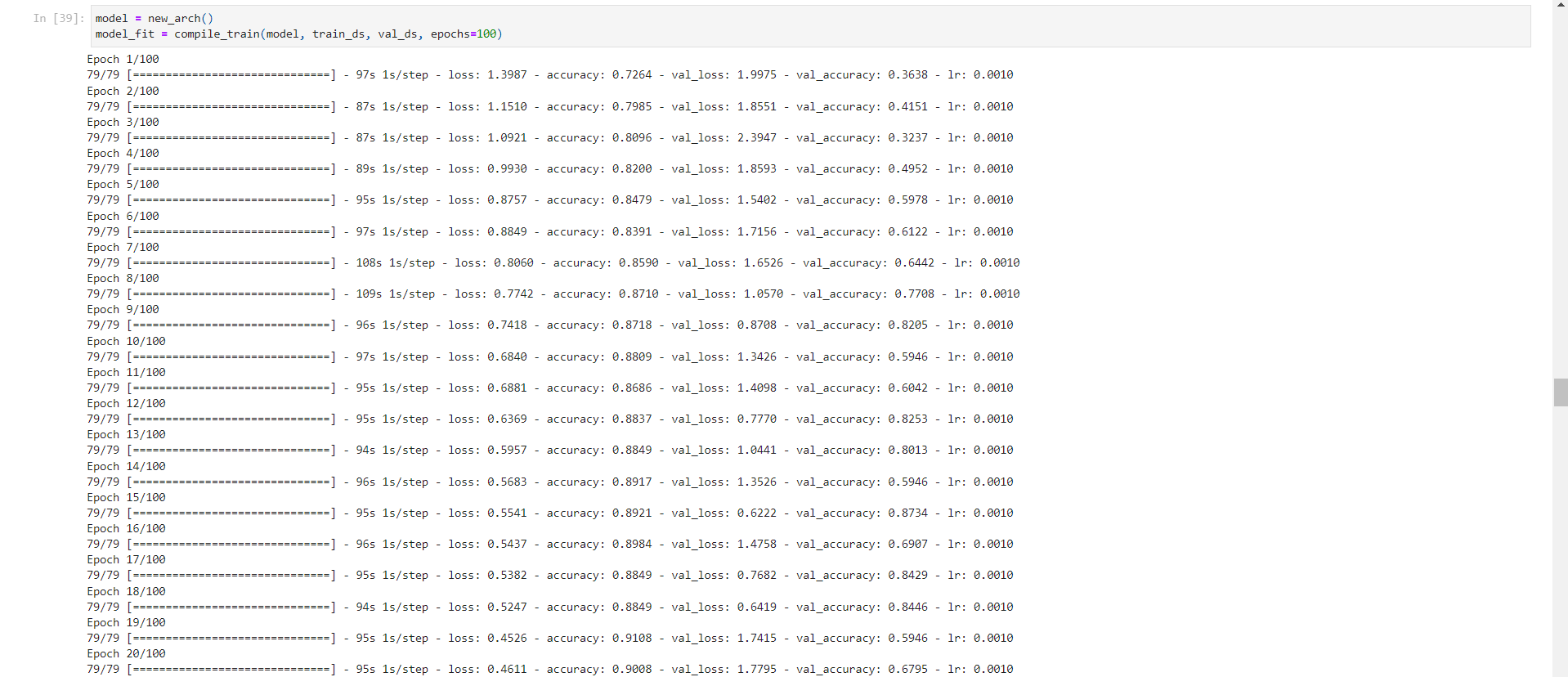
#### Task 3.3: Enhanced Training Process

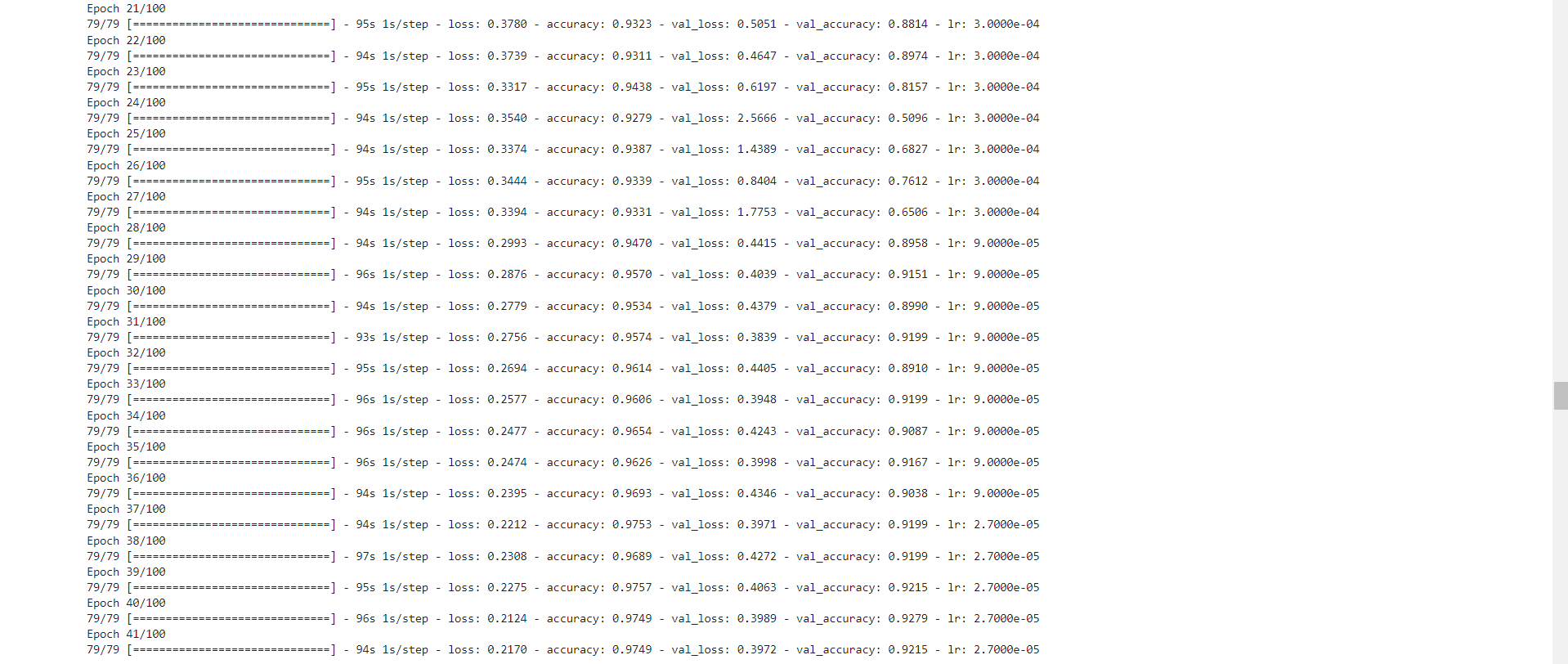
Figure 5.16: Code for Enhanced Training Process with Callbacks and Data Augmentation

Key Insights:

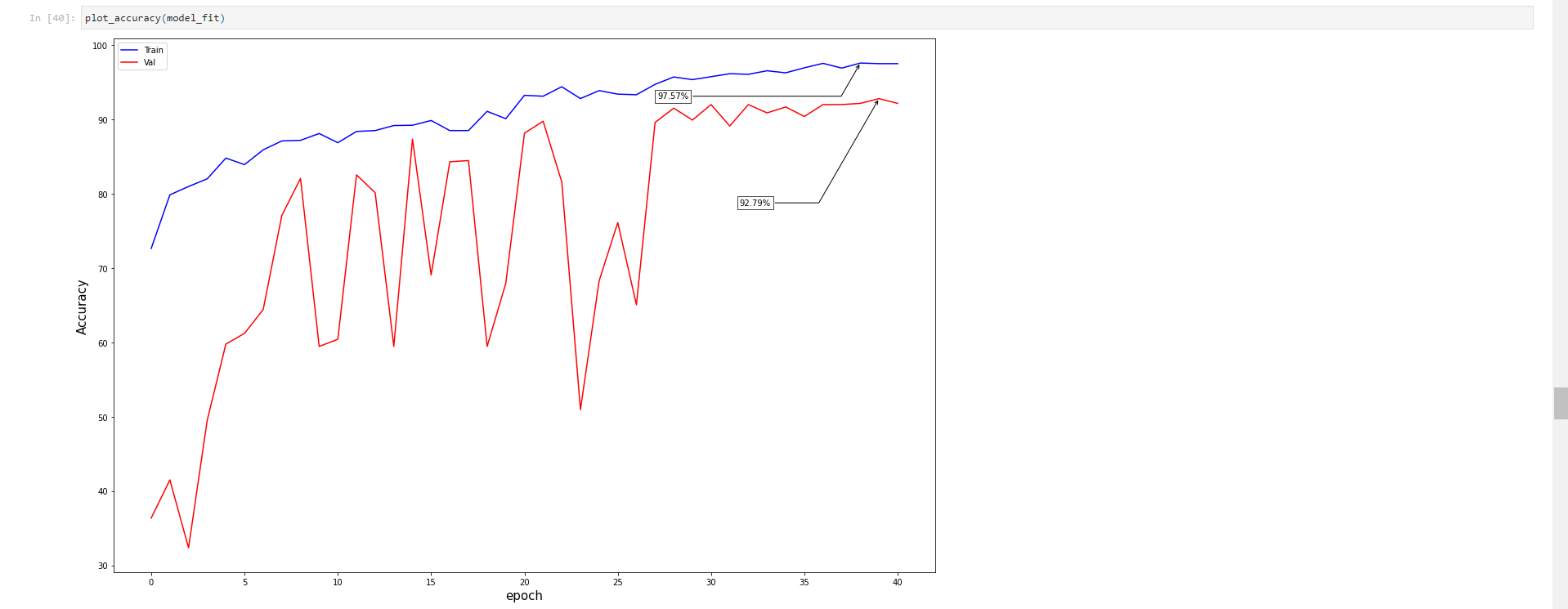
* The training process is enhanced with several callbacks:
  + ReduceLROnPlateau: Adjusts learning rate when improvement plateaus.
  + EarlyStopping: Prevents unnecessary epochs if the model stops improving.
  + TensorBoard: Allows for detailed monitoring of the training process.
* Data augmentation is introduced for the training set, including random cropping, which can help the model learn invariance to position and size.
* These enhancements aim to improve model generalization and prevent overfitting.

#### Task 3.4: Training the Improved Model

Figure 5.17: Code for Evaluating the Improved Model (1)

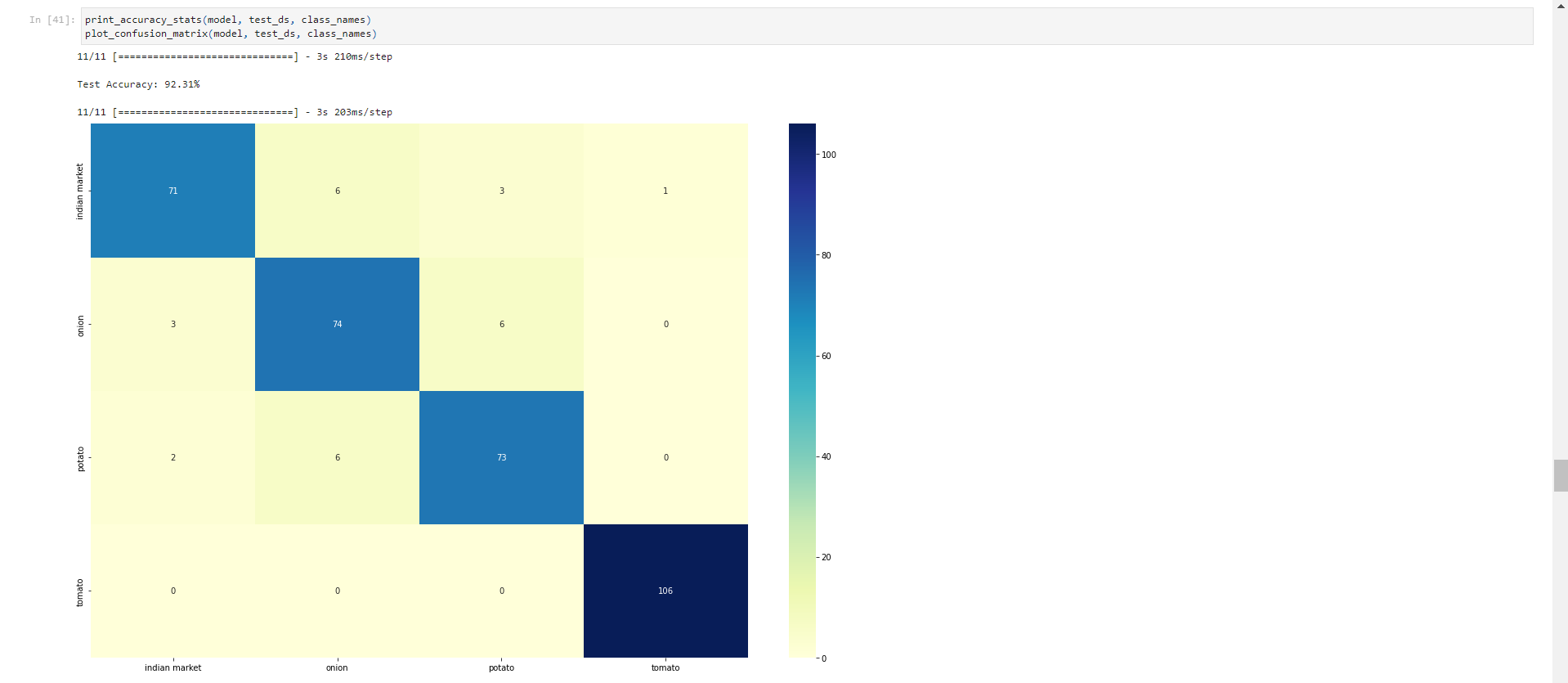
Figure 5.18: Code for Evaluating the Improved Model (2)

Key Insights:

Figure 5.19: Graph of Training and Validation Accuracy for Improved Model

* The model was trained for up to 100 epochs, but early stopping was likely triggered.
* Training accuracy reached 97.53%, while validation accuracy peaked at 92.79%.
* The gap between training and validation accuracy has significantly reduced compared to the baseline model, indicating better generalization.
* The learning rate reduction callback was triggered multiple times, helping to fine-tune the model's performance.
* The plot shows a more stable and closer relationship between training and validation accuracy, suggesting reduced overfitting.

#### Task 3.5: Evaluating the Improved Model

Figure 5.20: Table of Confusion Matrix for Improved Model

Key Insights:

* The improved model achieved a test accuracy of 92.31%, a significant improvement over the baseline model.
* The confusion matrix shows:
  + Improved performance across all classes.
  + Reduced confusion between 'onion' and 'potato' classes, though some misclassifications still occur.
  + Near-perfect classification of 'indian market' scenes.
* The visualization of random samples provides insights into:
  + Cases where the model is highly confident and correct.
  + Instances of misclassification, helping identify challenging examples.
* This evaluation demonstrates the effectiveness of the model improvements and data augmentation techniques.

#### Task 3.6: Transfer Learning with MobileNetV3Large

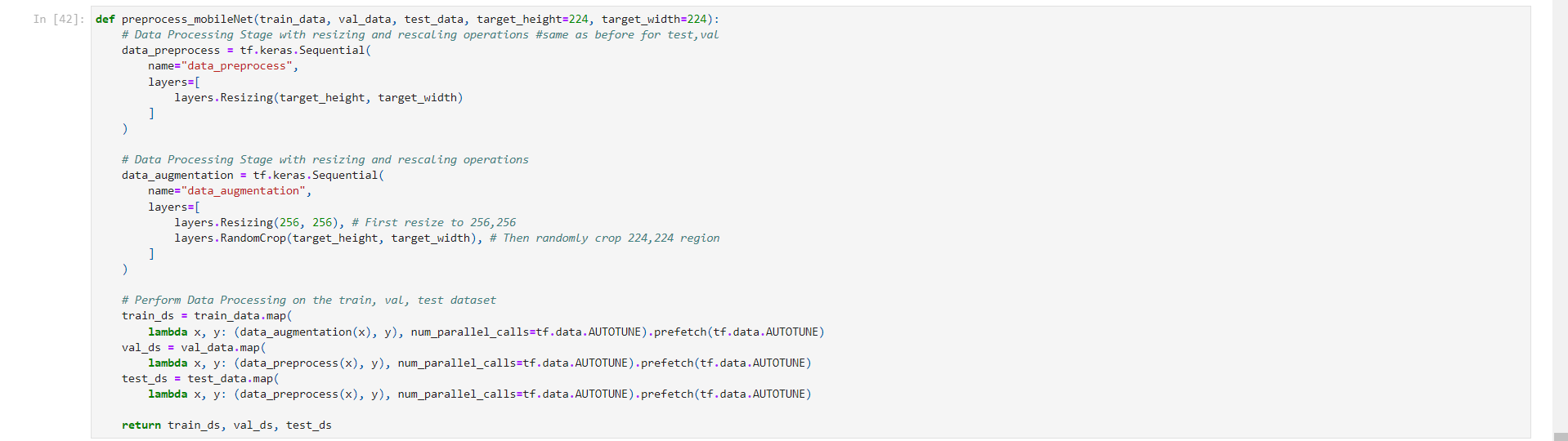
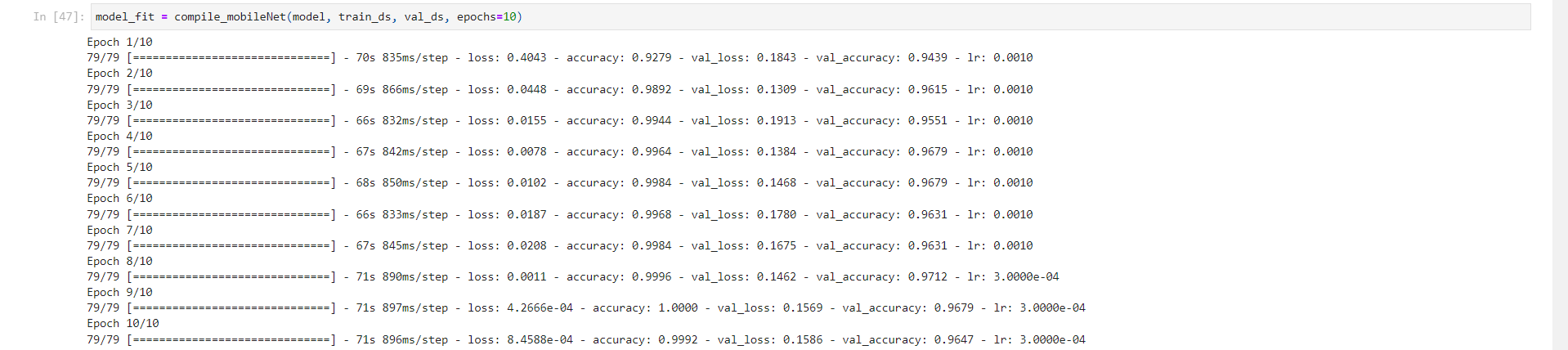
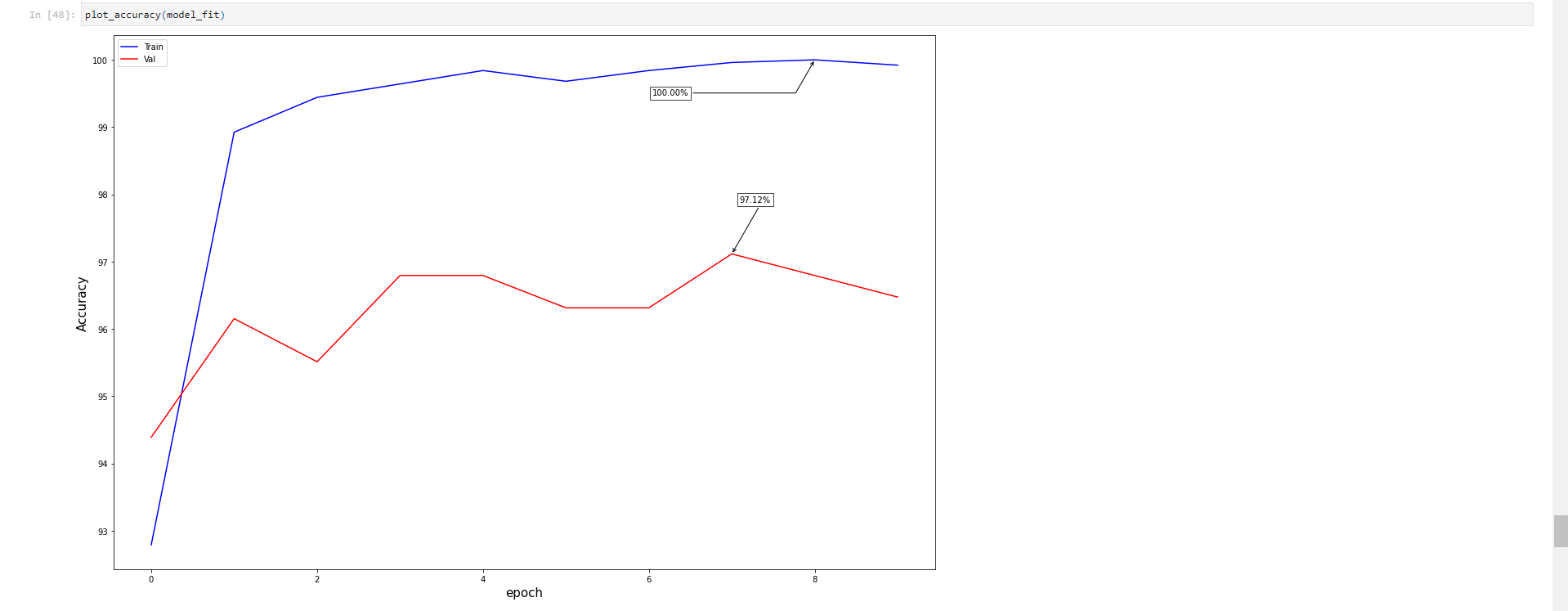
Figure 5.21: Code for Dividing Train Data into Train and Validation Sets

Figure 5.22: Code for Transfer Learning with MobileNetV3Large

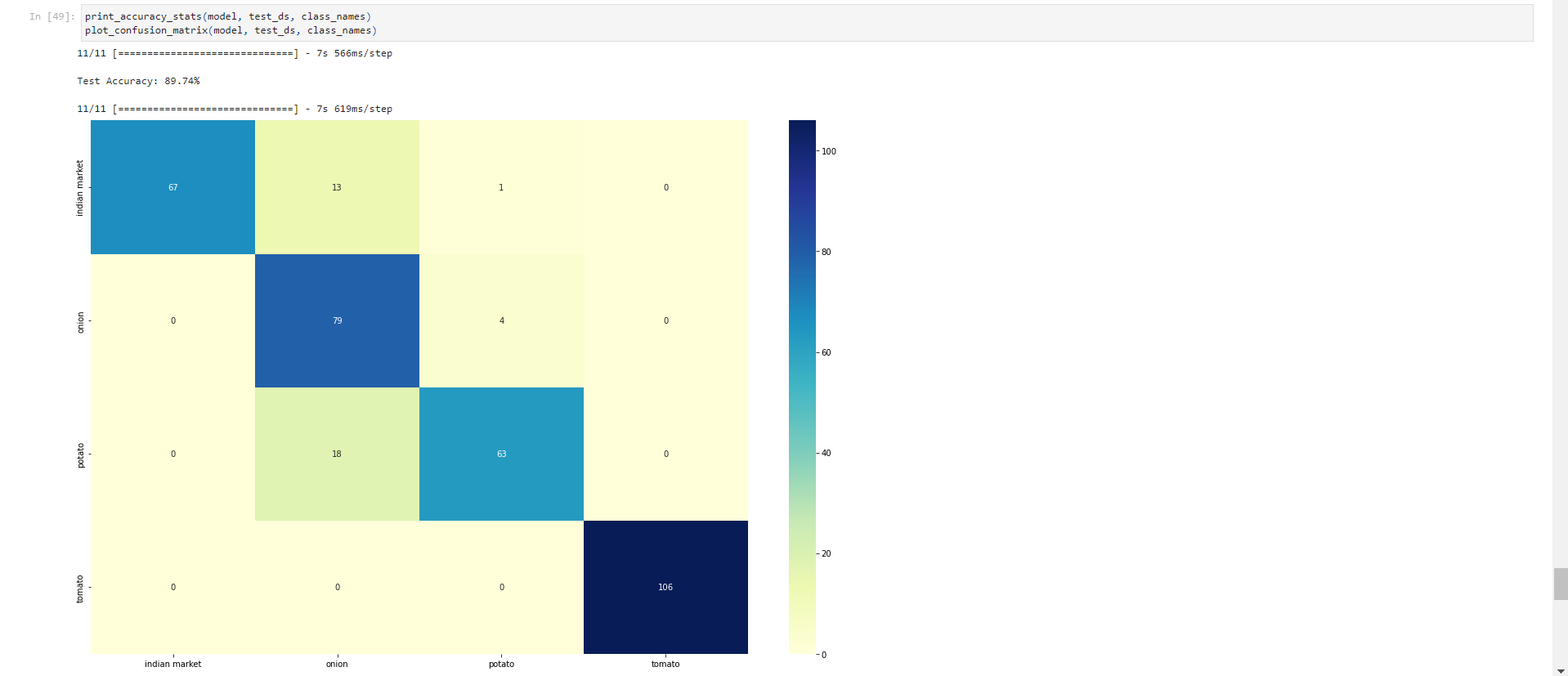
Figure 5.23: Code for Training the MobileNetV3Large

Figure 5.24: Graph of Training and Validation Accuracy for MobileNetV3Large Model

Key Insights:

* The MobileNetV3Large model, pre-trained on ImageNet, is used as a feature extractor.
* The top layers of MobileNetV3Large are removed, and custom classification layers are added.
* The pre-trained layers are frozen to preserve the learned features.
* The model is trained for 10 epochs with the following observations:
  + Training accuracy rapidly increased to 100% by the final epoch.
  + Validation accuracy peaked at 97.12% in the 8th epoch.
  + The learning rate was reduced twice during training, indicating the model was fine-tuning its performance.
* The transfer learning approach demonstrated superior efficiency in training and generalization compared to the custom CNN models.

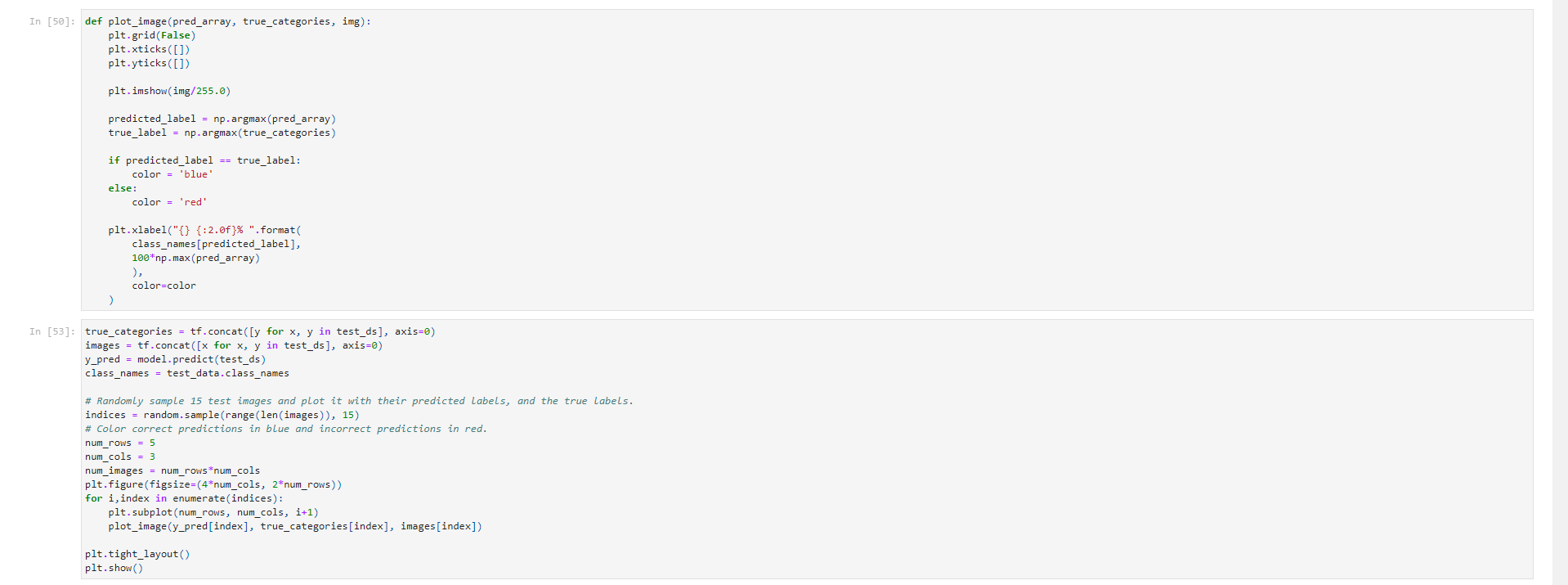
#### Task 3.7: Final Model Evaluation

Figure 5.25: Table of Confusion Matrix for MobileNetV3Large Model

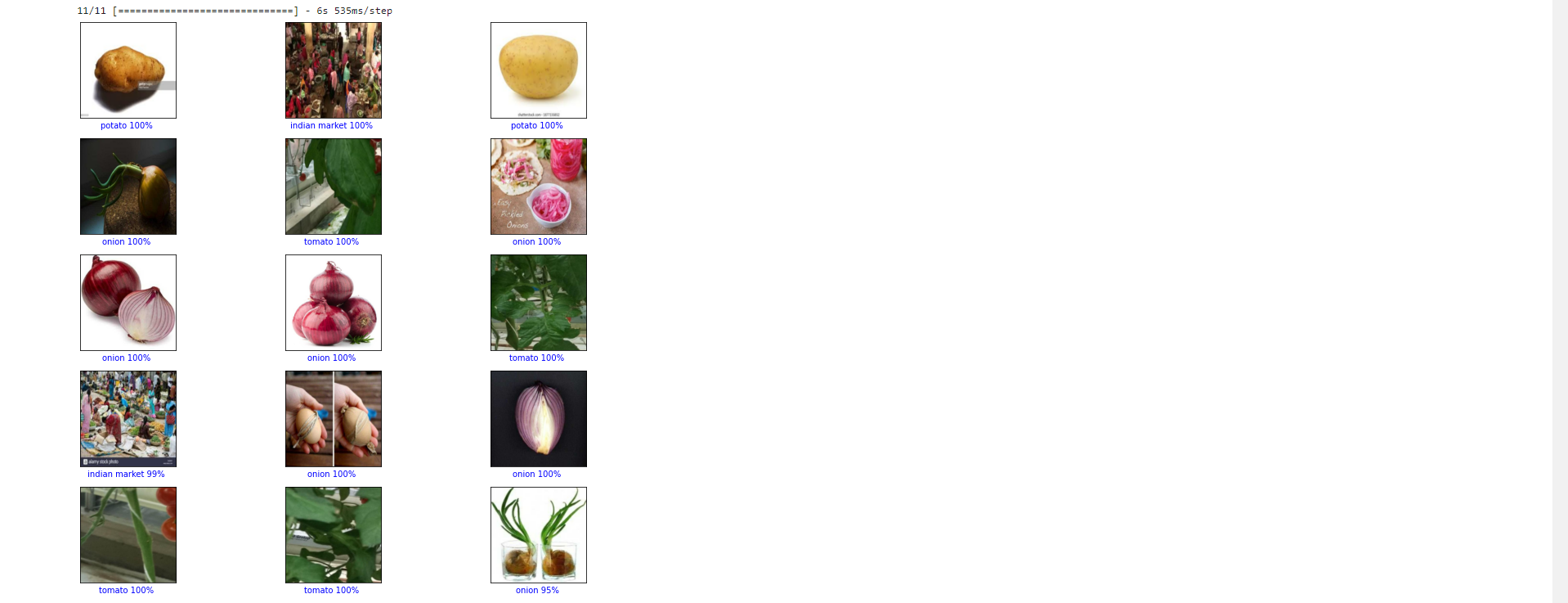
Key Insights:

* The MobileNetV3Large-based model achieved a test accuracy of 89.74%.
* This performance is slightly lower than the validation accuracy but still represents a significant improvement over the baseline and custom CNN models.
* The confusion matrix shows excellent performance on 'indian market' and 'tomato' classes, with some remaining confusion between 'onion' and 'potato'.
* The 'onion' class shows the most misclassifications, indicating it's the most challenging category for the model.

#### Task 3.8: Visualizing Model Predictions

Figure 5.26: Code for Visualizing Model Predictions

Key Insights:

Figure 5.27: Graph of Random Sample Predictions

* The visualization of random samples provides valuable insights:
  + Most predictions are correct and highly confident (90%+ certainty).
  + Misclassifications, when they occur, often involve 'onion' and 'potato' categories.
  + The model performs exceptionally well on 'indian market' scenes, likely due to their distinct visual characteristics.
  + Some challenging cases are revealed, such as partially obscured vegetables or unusual lighting conditions.
* This visualization helps in understanding the model's strengths and limitations, providing direction for potential further improvements.

### Analysis Part 4: Summary of Key Findings

#### 4.1 Dataset Characteristics and Preprocessing

* The dataset comprised 3,135 training images across four categories: tomato, potato, onion, and Indian market scenes.
* A slight class imbalance was observed, with potatoes having the highest representation (898 images) and Indian market scenes the lowest (599 images).
* Effective preprocessing techniques were implemented, including:
  + Resizing images to 224x224 pixels for consistency.
  + Implementing data augmentation (random cropping) to enhance model generalization.
  + Appropriate scaling of pixel values for different model architectures.

#### 4.2 Model Evolution and Performance

* **Baseline CNN Model**:
  + Achieved a test accuracy of 72.08%.
  + Showed significant overfitting, with training accuracy reaching 99.76% while validation accuracy peaked at 83.97%.
  + Revealed considerable confusion between onion and potato classes.
* **Improved CNN Architecture**:
  + Incorporated advanced techniques such as L2 regularization, batch normalization, and dropout.
  + Implemented data augmentation techniques, including random cropping and resizing.
  + Achieved a markedly improved test accuracy of 92.31%.
  + Demonstrated reduced overfitting and better generalization capabilities.
* **MobileNetV3Large Transfer Learning Model**:
  + Emerged as the best-performing model with a test accuracy of 89.74%.
  + Exhibited rapid convergence, with validation accuracy peaking at 97.12% during training.
  + Showed excellent generalization, with a small gap between training and validation accuracy.

#### 4.3 Comparative Analysis of Model Performance

* The progression from the baseline CNN to the transfer learning approach with MobileNetV3Large resulted in a substantial 17.66 percentage point improvement in test accuracy.
* Each iteration of the model showed improved ability to distinguish between similar vegetable classes, particularly onions and potatoes.
* The transfer learning approach demonstrated superior efficiency in training and generalization, leveraging pre-trained weights effectively.

#### 4.4 Class-Specific Performance Insights

* Consistent high accuracy was observed for the 'Indian market' class across all models, likely due to its distinct visual characteristics.
* The 'onion' class persistently posed the greatest challenge, often being confused with 'potato'.
* 'Tomato' classification showed steady improvement across model iterations, achieving high accuracy in the final model.

#### 4.5 Implications for Real-world Application

* The final model's performance suggests its viability for integration into the agri-tech supply chain company's automated produce identification system.
* The high accuracy on 'Indian market' scenes indicates the model's robustness in distinguishing between vegetable and non-vegetable images, a crucial feature for real-world deployment.
* Residual confusion between onion and potato classes suggests a need for focused improvements in these areas, possibly through targeted data collection or fine-tuning strategies.

## Insights and Recommendations

Based on our comprehensive analysis of the agri-tech supply chain company's produce identification challenge and the development of various image classification models, we have uncovered several key insights and developed corresponding recommendations:

### Insights:

1. **Model Performance and Architecture**
   * The model fine-tuned on the pre-trained MobileNetV3Large emerged as the most successful, achieving a test accuracy of 89.74%.
   * This success was largely due to the similarity between our training dataset of Indian vegetables and the "ImageNet" dataset, enabling efficient fine-tuning within 10 epochs.
   * The custom CNN models, while showing improvement with advanced techniques, did not match the performance of the transfer learning approach.
2. **Data Characteristics and Preprocessing**
   * The dataset, comprising 3,135 training images and 351 test images across four categories, provided a solid foundation for model development.
   * A slight class imbalance was observed, with potatoes having the highest representation (898 images) and Indian market scenes the lowest (599 images).
   * Data augmentation techniques, particularly random cropping and resizing, proved effective in enhancing model generalization.
3. **Class-Specific Performance**
   * The 'Indian market' class consistently showed the highest accuracy across all models, likely due to its distinct visual characteristics.
   * Persistent confusion between 'onion' and 'potato' classes was observed, representing the most challenging aspect of the classification task.
   * The 'tomato' class showed steady improvement across model iterations, achieving high accuracy in the final model.
4. **Model Generalization and Efficiency**
   * The MobileNetV3Large-based model demonstrated superior generalization, with a small gap between training and validation accuracy.
   * Transfer learning proved highly efficient, requiring fewer epochs to achieve high accuracy compared to custom CNN models.
5. **Impact of Model Complexity**
   * Increasing model complexity in custom CNNs (adding regularization, batch normalization, etc.) improved performance but also increased training time and resource requirements.
   * The MobileNetV3Large model offered an excellent balance between accuracy and potential for efficient deployment.

### Recommendations:

1. **Implement MobileNetV3Large-based Model**
   * Deploy the fine-tuned MobileNetV3Large model as the primary classification system in the production environment, given its superior performance and efficiency.
   * Utilize the model's high accuracy (89.74% on test data) to automate the produce sorting process, potentially reducing manual labor and associated costs.
2. **Focused Data Collection and Augmentation**
   * Prioritize collecting additional images of onions and potatoes in various conditions to help the model better differentiate between these classes.
   * Expand data augmentation techniques to include more variations in lighting and orientation, improving the model's robustness to real-world conditions.
3. **Class Balancing Strategy**
   * Implement oversampling techniques for the 'Indian market' class to balance the dataset, potentially improving overall model performance.
   * Monitor the impact of class balance on model accuracy and adjust the strategy as needed.
4. **Continuous Model Monitoring and Updating**
   * Implement a system for continuous model evaluation using real-world data from the company's operations.
   * Establish a feedback loop where misclassifications in the production environment are logged and used to retrain and improve the model periodically.
5. **Edge Deployment Optimization**
   * Given the efficiency of MobileNetV3Large, explore deployment on edge devices at sorting facilities to enable real-time, on-site classification without the need for constant network connectivity.
   * Conduct performance profiling to ensure the model meets speed requirements for real-time application in the supply chain.
6. **Hierarchical Classification Approach**
   * Consider implementing a two-stage classification process: first distinguishing between vegetable and non-vegetable (market) images, then classifying the specific vegetable type.
   * This approach may improve accuracy for similar-looking produce (onions and potatoes) by focusing the model's attention on subtle differences.
7. **Integration with Inventory Management System**
   * Develop APIs to seamlessly integrate the classification model with the company's existing inventory management system, enabling automated stock updates and quality control processes.
   * Use the model's output to track produce quantities and varieties in real-time, improving overall supply chain efficiency.
8. **Iterative Model Improvement**
   * Establish a regular schedule for model retraining and fine-tuning, incorporating new data and addressing any performance drift over time.
   * Consider experimenting with ensemble methods in future iterations, combining predictions from multiple models to potentially improve classification accuracy for challenging cases.
9. **Expand Classification Capabilities**
   * Gradually expand the model's capabilities to classify a wider range of produce, prioritizing based on the company's most frequently handled items.
   * Investigate the potential for fine-grained classification within each vegetable category (e.g., different varieties of tomatoes) as a future enhancement.