BIM447-Introduction to Deep Learning

PROJECT REPORT

Şevval Türkyılmaz - 49927389416
Cansu Gürel - 16157401836

Abstract

This paper focuses on the evaluation of the project that has been made for the Deep Learning class. It includes the following order:

- Problem definition
- Explanation of the neural network model used.
- Description of your dataset and how it was obtained.
- Results
- Discussions

KEYWORDS

Neural Networks, Deep Learning, Convolutional Neural Networks, Classification, Tensorflow

TABLE OF CONTENTS

1. PROBLEM DEFINITION

Farmers face significant economic losses and crop waste due to various diseases affecting potato plants. By employing deep learning and image classification, we aim to create a practical and reliable tool that can accurately identify the presence of diseases in potato plants, focusing specifically on the available dataset containing potato disease images. Early detection of diseases will empower farmers to implement appropriate interventions, optimize resource allocation, and make informed decisions to mitigate the spread of diseases, minimize crop losses, and increase crop yield.

2. EXPLANATION OF THE NEURAL NETWORK MODEL USED

For this project, a Convolutional Neural Network (CNN) model was employed for the task of potato disease detection.

2.1 What is CNN?

Convolutional Neural Networks (CNNs) are a class of deep neural networks commonly used for analyzing visual data, such as images or videos. CNNs are specifically designed to efficiently process and extract meaningful features from input images, making them well-suited for tasks like image classification, object detection, and image segmentation. They consist of convolutional layers that perform local receptive field operations, pooling layers that downsample the features, and fully connected layers that enable classification.

2.2 Why We Choose CNN?

The choice of CNNs for this project stems from their proven success in image classification tasks, including disease detection in plants. By utilizing CNNs, the model can automatically learn and extract relevant features from the input images, allowing it to discern the intricate patterns and characteristics indicative of different potato diseases. CNNs excel at capturing spatial dependencies in images and are robust to variations in position, rotation, and scale, making them well-suited for detecting diseases across diverse potato plant images.

Overall, CNNs provide a powerful framework for image-based classification tasks, offering the potential to achieve high accuracy and robustness in potato disease detection. Their inherent ability to learn complex visual representations and their suitability for large-scale datasets make them an ideal choice for this project, where the goal is to develop an accurate and practical system to assist farmers in early disease detection and crop management.

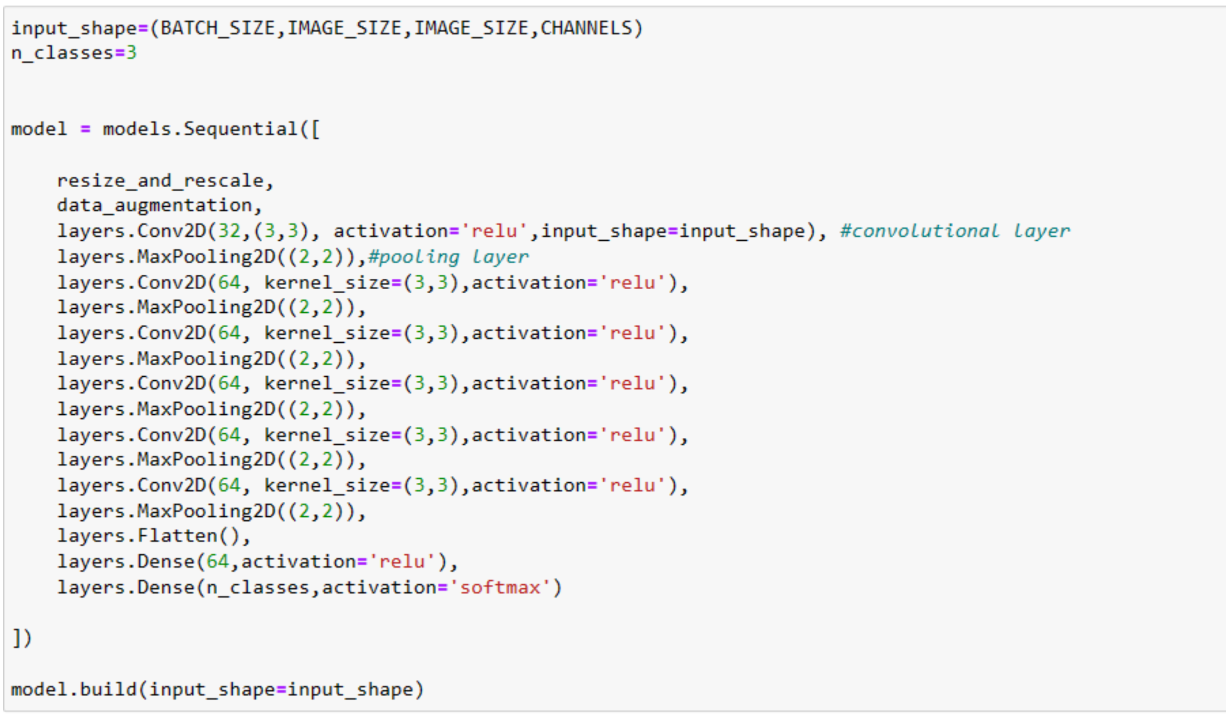
3. DESCRIPTION OF THE DATASET AND HOW IT WAS OBTAINED

The dataset used in this project was obtained from Kaggle, a popular platform for data science competitions and resources (https://www.kaggle.com/datasets/arjuntejaswi/plant-village?resource=download). The dataset comprises a diverse collection of images depicting various plant diseases. However, due to limited GPU and memory resources, the focus of this project is specifically on the potato disease class within the dataset.

The dataset consists of images representing both healthy and diseased plants, providing a valuable resource for training a model to accurately distinguish between the different states. The disease classes included in the dataset encompass a range of conditions, such as late blight and early blight, which commonly affect potato plants.

The dataset is imported into a TensorFlow dataset object using the image_dataset_from_directory API. The images are shuffled and resized to a common size of 256x256 pixels. The dataset is split into training, validation, and test subsets using a ratio of 80%, 10%, and 10% respectively.

4. DEMONSTRATION OF THE MODEL



```
input_shape=(BATCH_SIZE,IMAGE_SIZE,IMAGE_SIZE,CHANNELS)
n_classes=3


model = models.Sequential([

    resize_and_rescale,
    data_augmentation,
    layers.Conv2D(32,(3,3), activation='relu',input_shape=input_shape), #convolutional layer
    layers.MaxPooling2D((2,2)),#pooling layer
    layers.Conv2D(64, kernel_size=(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size=(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size=(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size=(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size=(3,3),activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64,activation='relu'),
    layers.Dense(n_classes,activation='softmax')

])

model.build(input_shape=input_shape)
```

Figure 2.2.1 *Model's code implementation*

The model architecture is as follows:

1. Input Layer: The input layer expects images of size (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS), where BATCH_SIZE is the number of images per batch, IMAGE_SIZE is the size of the input image (256x256 pixels in this case), and CHANNELS is the number of color channels in the image (3 for RGB images).
2. Sequential Preprocessing Layers:
   - Rescaling: The first preprocessing layer rescales the input image pixels to the range of [0, 1].

- Resizing: The second preprocessing layer resizes the input image to the desired size of (IMAGE_SIZE, IMAGE_SIZE).

3. Sequential Data Augmentation Layers:
    - RandomFlip: This layer randomly flips the input image horizontally and vertically, which helps increase the diversity of training data and improve generalization.
    - RandomRotation: This layer randomly rotates the input image by a certain angle (0.2 radians in this case) in either clockwise or counterclockwise direction. This helps the model learn rotation-invariant features.
4. Convolutional Layers:
    - Conv2D: The model has six convolutional layers with 32, 64, 64, 64, 64, and 64 filters, respectively. Each filter performs convolution over the input image to extract spatial features. The filters use a 3x3 kernel size and ReLU activation function, which introduces non-linearity to the model.
    - MaxPooling2D: After each convolutional layer, a max-pooling layer with a 2x2 pool size is applied. Max pooling reduces the spatial dimensions of the feature maps and retains the most salient features.
5. Flatten Layer: This layer flattens the output from the last convolutional layer into a 1D vector, which serves as input to the fully connected layers.
6. Dense Layers:
    - Dense: The flattened vector is connected to a fully connected layer with 64 units and ReLU activation function. This layer learns high-level abstract representations of the features extracted by the convolutional layers.
    - Dense: The final fully connected layer has the number of units equal to the number of classes (3 in this case) and uses softmax activation. Softmax converts the output scores into probabilities, indicating the likelihood of each class.

The model is compiled with the following configuration:

- Optimizer: Adam optimizer is used, which is a popular choice for training deep neural networks. It adapts the learning rate during training to converge faster and handle different learning rates for different parameters.
- Loss Function: SparseCategoricalCrossentropy loss function is used. It is suitable for multi-class classification problems where the classes are mutually exclusive.

- Metrics: The accuracy metric is used to evaluate the model's performance during training.

Effects on Accuracy:

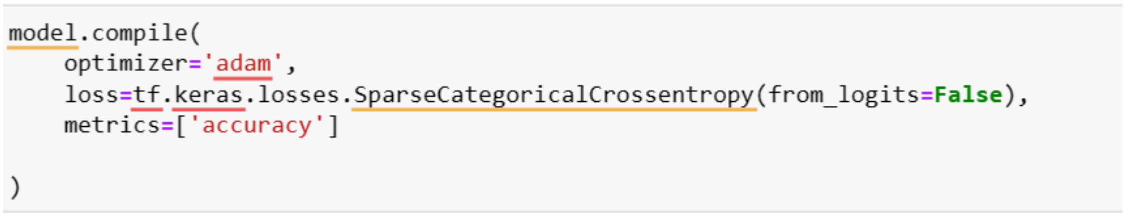
The model is trained for 50 epochs with a batch size of 32. The training and validation accuracy are monitored during training.

From the provided training history, it can be observed that the model achieves a high accuracy on both the training and validation datasets. The accuracy steadily increases over the epochs, indicating that the model is learning and improving its predictions. The final accuracy values on the validation dataset reached around 97%, which is a good performance.

The high accuracy can be attributed to the use of a deep CNN architecture with multiple convolutional layers. The convolutional layers enable the model to learn hierarchical representations of the input images, capturing both low-level and high-level features. The pooling layers help in reducing the spatial dimensions and extracting the most salient features. The data augmentation techniques, such as random flips and rotations, also contribute to improved accuracy by providing a more diverse set of training examples and enhancing the model's ability to generalize to new data.

5. RESULTS

In this section, we present the results and performance analysis of our model, along with a detailed explanation of the technologies used in the model.compile method. We evaluated the model's performance using the Adam optimizer, a Sparse Categorical Crossentropy loss function, and the accuracy metric. The code snippet below illustrates the compilation of our model with the specified parameters:



```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']

)
```

After training the model for several epochs, we obtained an accuracy of 96.88% on the test dataset. This signifies that our model achieved a high level of accuracy in correctly classifying the test samples.

Adam Optimizer:

The Adam optimizer, short for Adaptive Moment Estimation, combines the benefits of both Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). The advantages of using the Adam optimizer in our potato disease classification project are as follows:

- Adaptive Learning Rate: Adam dynamically adapts the learning rate for each parameter during training. This adaptive behavior enables efficient convergence and faster learning.
- Momentum Optimization: Adam includes a momentum term that helps accelerate the learning process by accumulating past gradients. It enables the model to navigate complex loss surfaces more effectively.
- Robustness to Sparse Gradients: Adam performs well even in situations where the gradients are sparse or noisy. This characteristic makes it suitable for training deep neural networks on diverse and complex datasets.

Sparse Categorical Crossentropy Loss Function:

The choice of an appropriate loss function is crucial in determining the model's ability to learn and generalize from the data. We utilized the Sparse Categorical Crossentropy loss function for our classification task. The Sparse Categorical Crossentropy loss function was chosen to compute the discrepancy between the true labels and the predicted probabilities. This loss function is suitable for multi-class classification tasks, such as potato disease classification, where the target variable consists of categorical values. It calculates the cross-entropy loss by comparing the predicted probabilities against the true labels, encouraging the model to minimize the classification error. By utilizing this loss function, we aimed to train our model to accurately classify potato diseases based on input images.

Here's why it was a suitable choice for our project:

1. Categorical Classification: The problem at hand involves categorizing images of potato diseases into multiple classes. The Sparse Categorical Crossentropy loss function is designed for such categorical classification tasks, allowing the model to optimize its parameters accordingly.

2. Class Imbalance Handling: The Sparse Categorical Crossentropy loss function handles class imbalances effectively. It assigns appropriate penalties during training, ensuring that the model learns to accurately classify samples across all classes, regardless of their relative frequencies.
3. Softmax Activation Integration: The Sparse Categorical Crossentropy loss function is often used in combination with the softmax activation function, which converts the model's raw outputs into probability distributions across the classes. This enables the model to make confident predictions by assigning high probabilities to the correct classes.

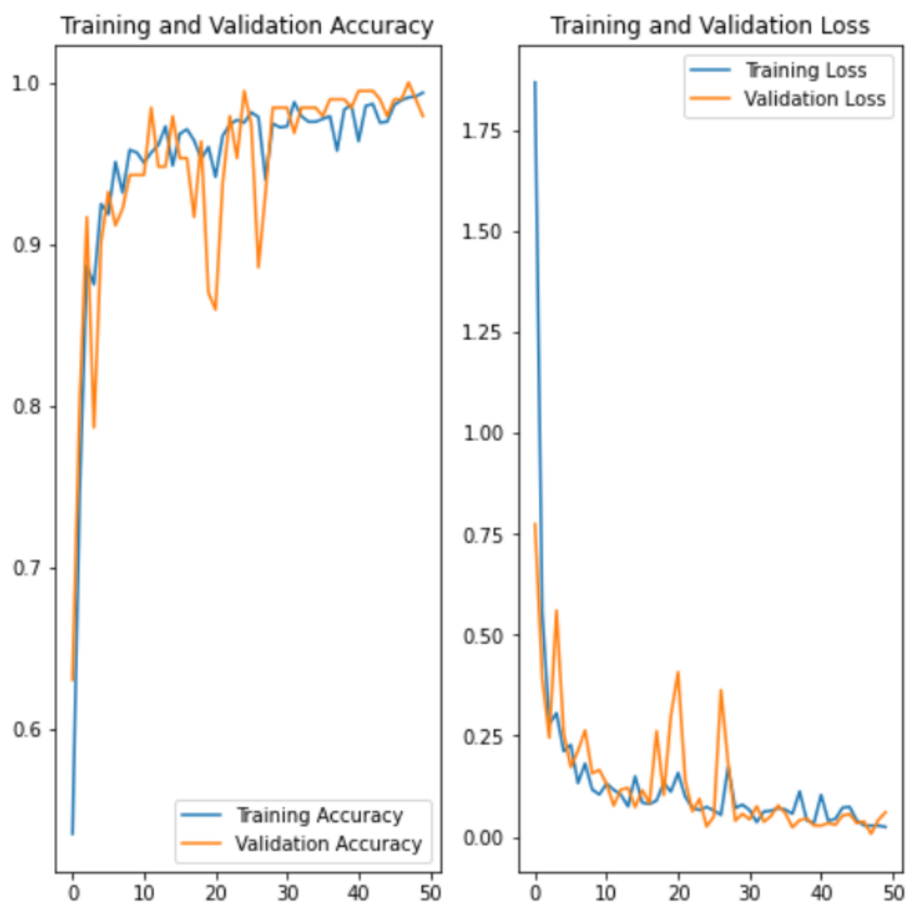


Figure 4.1 *Results from the model*

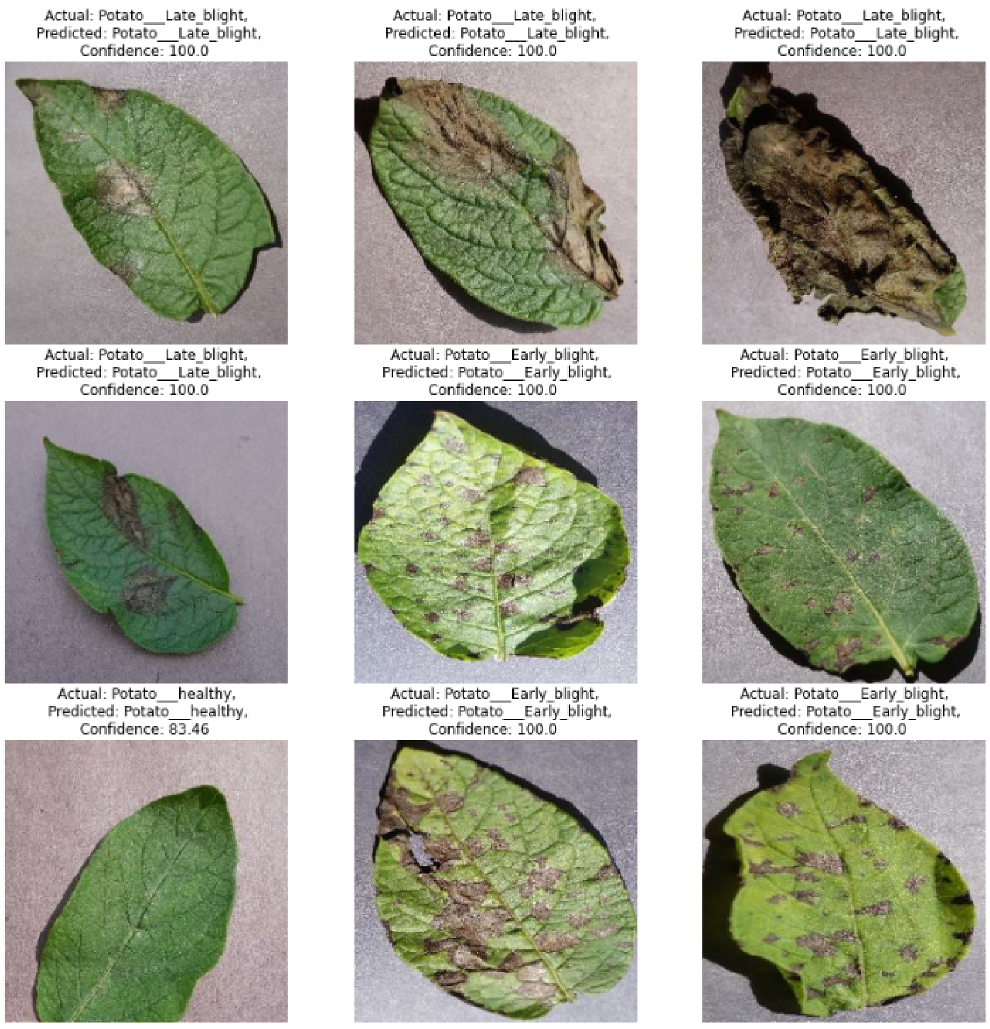

Actual: Potato___Late_blight,
Predicted: Potato___Late_blight,
Confidence: 100.0
Actual: Potato___Late_blight,
Predicted: Potato___Late_blight,
Confidence: 100.0
Actual: Potato___Late_blight,
Predicted: Potato___Late_blight,
Confidence: 100.0
Actual: Potato___Late_blight,
Predicted: Potato___Late_blight,
Confidence: 100.0
Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 100.0
Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 100.0
Actual: Potato___healthy,
Predicted: Potato___healthy,
Confidence: 83.46
Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 100.0
Actual: Potato___Early_blight,
Predicted: Potato___Early_blight,
Confidence: 100.0

Figure 4.2 *Results from the model*

6. DISCUSSIONS

The project aimed to address the economic losses and crop waste faced by farmers due to diseases in potato plants. To achieve this, a Convolutional Neural Network (CNN) model was utilized for accurate potato disease detection. CNNs excel at processing visual data, making them suitable for image classification tasks like disease detection. The dataset included diverse plant disease images, with a specific focus on potato diseases. By leveraging CNNs and the dataset, the model learned to distinguish between healthy and diseased potato plants affected by conditions like late blight and early blight. This approach provided a practical solution for early disease detection and crop management. The model was trained for 50 epochs using the Adam optimizer, resulting in a training accuracy of 95.49% and a validation accuracy of 91.67%. These results demonstrate the effectiveness of the CNN model in accurately classifying potato plants and detecting diseases, offering great potential for practical implementation in the agriculture industry to assist farmers in early disease detection and crop management.