Assignment 7

Name: Vaishnavi Gosavi
Roll No: 71
Class: TY CSAI-A
Batch: 2

Title: Convolutional neural network (CNN) Use any dataset of plant disease and design a plant disease detection system using CNN.

Description:-

- **Convolutional Neural Network (CNN)**
  A Convolutional Neural Network (CNN) is a class of deep neural networks most commonly used in analyzing visual imagery. CNNs are inspired by the visual cortex of the human brain, which processes visual data hierarchically.

- **Architecture Components:**

1. **Input Layer**
   - Accepts images of shape (28x28x1) for grayscale MNIST data.

2. **Convolutional Layer**
   - Applies filters (kernels) to extract low-level features like edges, textures, and patterns.
   - Operation: Convolution between input and filter.
   - Output is called a feature map.

3. **Activation Layer (ReLU)**
   - Applies the ReLU (Rectified Linear Unit) function: f(x) = max(0, x)
   - Introduces non-linearity and prevents vanishing gradients.

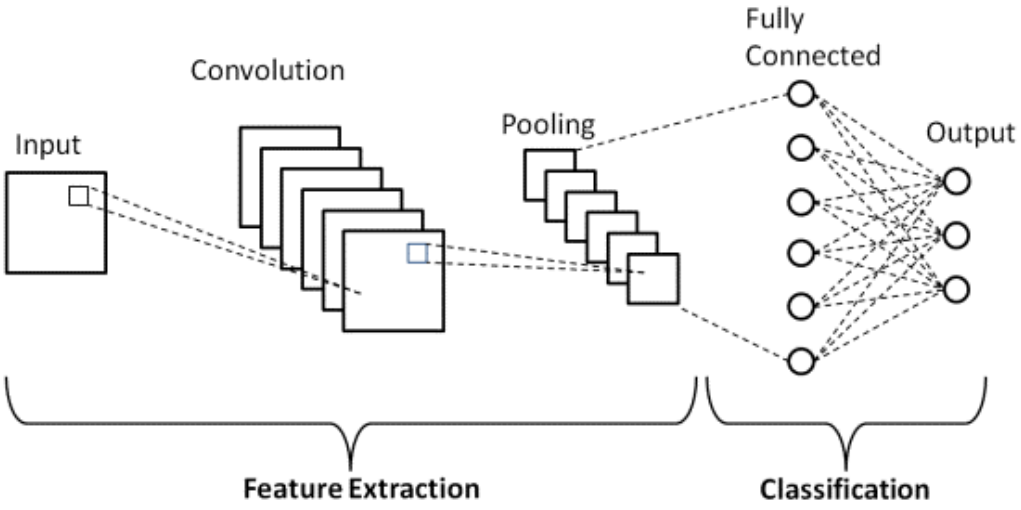
4. **Pooling Layer (Max Pooling)**
   - Reduces spatial dimensions (width and height) of feature maps.
   - Helps in making the model more efficient and invariant to minor translations.
   - E.g., 2x2 MaxPooling reduces 28x28 → 14x14.

5. **Flatten Layer**
   - Converts the 2D feature maps into a 1D vector to feed into the Dense (fully connected) layer.

6. **Dense Layer**
   - Fully connected layer where each neuron is connected to all outputs from the previous layer.
   - Learns high-level global patterns.
   - Plot actual vs. predicted stock prices.

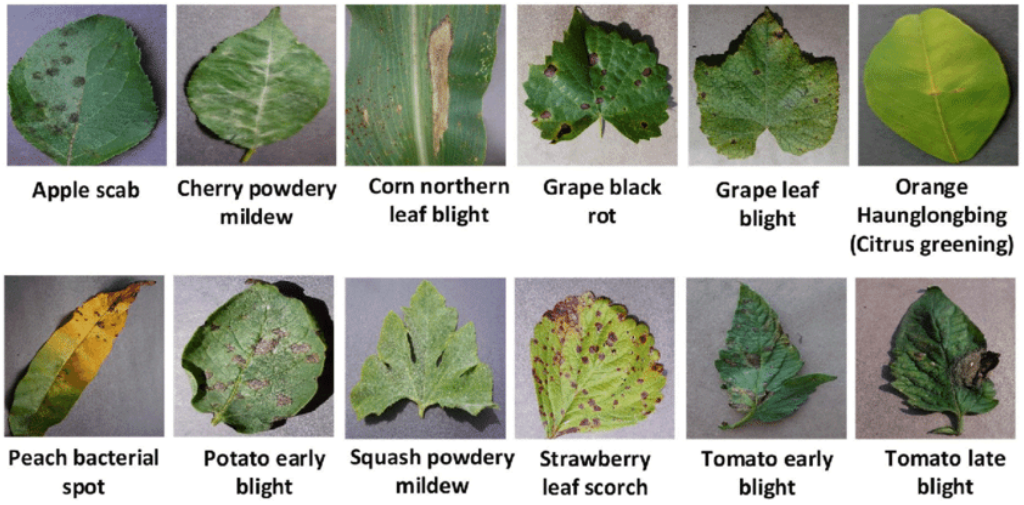
7. **Output Layer**
   - Uses Softmax activation function to output probabilities for each of the 10 classes.
   - Softmax ensures the outputs sum to 1 and represents confidence for each class.

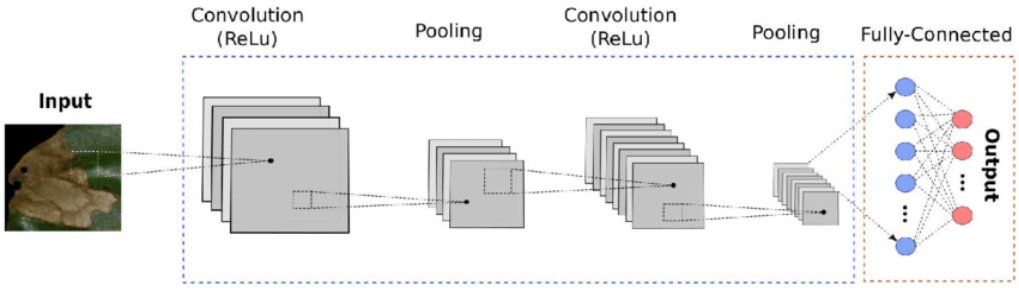
7. **Output Layer**
   - Uses Softmax activation function to output probabilities for each of the 10 classes.
   - Softmax ensures the outputs sum to 1 and represents confidence for each class.



- **Working Mechanism of CNN on MNIST**
  1. Input image (28x28) is fed to the CNN.
  2. First Conv layer extracts edge-level features (horizontal, vertical lines).
  3. Pooling layer reduces dimensions, keeps the most important features.
  4. Second Conv + Pooling layers extract more abstract features like loops, curves.
  5. Flattening makes the data suitable for the Dense layer.
  6. Dense layers learn complex relationships and finally classify the digit using Softmax.

- **New Plant Diseases Dataset**
  - The New Plant Diseases Dataset from Kaggle (vipoooool/new-plant-diseases-dataset) is a large collection of images used for multiclass classification of plant diseases using deep learning models like CNN.
    - It contains ~87,000+ RGB images across 38 classes, each representing a specific plant-disease or healthy status (e.g., Tomato___Leaf_Mold, Potato___Early_blight, Apple___healthy).
    - Images are organized into folders by class, making it easy to load using libraries like Keras or PyTorch.
    - Each image is typically resized (e.g., 128x128) and normalized before training.
    - It is widely used to build AI tools for crop disease detection in smart farming and agriculture automation.
  - Example Class Labels:
    - Tomato___Late_blight
    - Grape___Black_rot
    - Apple___Black_rot
    - Tomato___healthy





- **Algorithm**

1. Import necessary libraries
2. Download dataset from KaggleHub
3. Set up data directories (train, test)
4. Preprocess data:
   - Resize images to (128, 128)
   - Normalize pixel values (divide by 255)
   - Apply data augmentation
5. Create training and validation generators using ImageDataGenerator
6. Define CNN model:
   - Conv2D -> ReLU -> MaxPooling
   - Conv2D -> ReLU -> MaxPooling
   - Flatten -> Dense -> Dropout -> Output(Softmax)
7. Compile model with:
   - Optimizer: Adam
   - Loss: Categorical Crossentropy
   - Metric: Accuracy
8. Train the model on training data
9. Evaluate model on test data
10. Predict labels for test set
11. Generate and display confusion matrix

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("vipoooool/new-plant-diseases-dataset")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/vipoooool/new-plant-diseases-dataset?dataset_version_number=2...

100%|██████████| 2.70G/2.70G [00:26<00:00, 108MB/s]

Extracting files...


Path to dataset files: /root/.cache/kagglehub/datasets/vipoooool/new-plant-diseases-dataset/versions/2
```

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

training_set = train_datagen.flow_from_directory(
    '/root/.cache/kagglehub/datasets/vipoooool/new-plant-diseases-dataset/versions/2/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/train',
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical'
)

test_set = test_datagen.flow_from_directory(
    '/root/.cache/kagglehub/datasets/vipoooool/new-plant-diseases-dataset/versions/2/new plant diseases dataset(augmented)/New Plant Diseases Dataset(Augmented)/valid',
    target_size=(64, 64),
```

```
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 70295 images belonging to 38 classes.
Found 17572 images belonging to 38 classes.
```

```
num_classes=38

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64,
3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

```
/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 layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)
```

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

```
model.fit(
    training_set,
    steps_per_epoch=len(training_set),
    epochs=10,
    validation_data=test_set,
    validation_steps=len(test_set)
)
```

```
/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
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 496s 225ms/step - accuracy: 0.4938 -
loss: 1.7903 - val_accuracy: 0.6850 - val_loss: 1.0673
Epoch 2/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 482s 219ms/step - accuracy: 0.8128 -
```

```
loss: 0.5967 - val_accuracy: 0.8273 - val_loss: 0.5435
Epoch 3/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 484s 220ms/step - accuracy: 0.8609 -
loss: 0.4295 - val_accuracy: 0.8286 - val_loss: 0.5665
Epoch 4/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 505s 222ms/step - accuracy: 0.8896 -
loss: 0.3413 - val_accuracy: 0.8787 - val_loss: 0.3802
Epoch 5/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 495s 218ms/step - accuracy: 0.9083 -
loss: 0.2834 - val_accuracy: 0.8865 - val_loss: 0.3609
Epoch 6/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 477s 217ms/step - accuracy: 0.9200 -
loss: 0.2411 - val_accuracy: 0.8587 - val_loss: 0.5298
Epoch 7/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 475s 216ms/step - accuracy: 0.9290 -
loss: 0.2165 - val_accuracy: 0.8671 - val_loss: 0.4513
Epoch 8/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 502s 216ms/step - accuracy: 0.9375 -
loss: 0.1901 - val_accuracy: 0.9079 - val_loss: 0.2921
Epoch 9/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 481s 219ms/step - accuracy: 0.9420 -
loss: 0.1715 - val_accuracy: 0.8940 - val_loss: 0.3563
Epoch 10/10
2197/2197 ━━━━━━━━━━━━━━━━━━━━ 496s 216ms/step - accuracy: 0.9450 -
loss: 0.1687 - val_accuracy: 0.9175 - val_loss: 0.2729

<keras.src.callbacks.history.History at 0x7c57c04f5350>

model.save('plant_disease_model.h5')

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

def preprocess_image(image_path):
  img = image.load_img(image_path, target_size=(64, 64))  # Adjust
target_size if needed
  img = image.img_to_array(img)
  img = np.expand_dims(img, axis=0)
  img = img / 255.0  # Rescale pixel values
  return img
def predict_disease(image_path):
  processed_image = preprocess_image(image_path)
  prediction = model.predict(processed_image)
  predicted_class_index = np.argmax(prediction)

  # Get class labels (assuming you have a list of class names)
  class_labels = list(training_set.class_indices.keys())  # Get class
```

```
labels from training_set

  predicted_class_label = class_labels[predicted_class_index]
  return predicted_class_label

import numpy as np
from tensorflow.keras.preprocessing import image
image_path = '/root/.cache/kagglehub/datasets/vipoooool/new-plant-
diseases-dataset/versions/2/new plant diseases dataset(augmented)/New
Plant Diseases Dataset(Augmented)/valid/Apple___Apple_scab/00075aa8-
d81a-4184-8541-b692b78d398a___FREC_Scab 3335_270deg.JPG'
predicted_label = predict_disease(image_path)
print("Predicted disease:", predicted_label)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 274ms/step
Predicted disease: Apple___Apple_scab
```