powered by
bytewise.

www.bytewiseltd.com

Final Project

Status	Not started
Reviewed	☑

Final Project Report: Image Classification with CNN

Introduction

For the Bytewise ML Fellowship Program, I developed an image classification project using TensorFlow and Streamlit. The goal of this project was to classify images of vegetables and fruits into predefined categories using a Convolutional Neural Network (CNN) model. The project involved several key components, including data preprocessing, model training, and creating a web application for real-time image classification.

1. Data Preparation

1.1. Dataset:

- **Training Data:** Images were divided into three directories: `train`, `test`, and `validation`.
- **Image Size:** The images were resized to 180×180 pixels for uniformity.

1.2. Data Loading:

The dataset was loaded using TensorFlow's `image_dataset_from_directory` utility, which helped streamline the process of image loading and preprocessing.

```
data_train = tf.keras.utils.image_dataset_from_directory(
    data_train_path,
    shuffle=True,
    image_size=(img_width, img_height),
    batch_size=32,
    validation_split=False
)

data_cat = data_train.class_names

data_val = tf.keras.utils.image_dataset_from_directory(data
_val_path,
    image_size=(img_height,img_width),
    batch_size=32,
    shuffle=False,
    validation_split=False
)

data_test = tf.keras.utils.image_dataset_from_directory(
    data_test_path,
    image_size=(img_height,img_width),
    shuffle=False,
    batch_size=32,
    validation_split=False
)
```

1.3. Data Visualization:

Visualized a few images from the training dataset to understand the data distribution and verify correct loading.

```
plt.figure(figsize=(10,10))
for image, labels in data_train.take(1):
    for i in range(9):
        plt.subplot(3,3,i+1)
        plt.imshow(image[i].numpy().astype('uint8'))
        plt.title(data_cat[labels[i]])
        plt.axis('off')
```

2. Model Building

2.1. Model Architecture:

Developed a Convolutional Neural Network (CNN) using TensorFlow and Keras. The architecture included several convolutional layers followed by max-pooling layers, a flattening layer, and dense layers.

```
model = Sequential([
    layers.Rescaling(1./255),
    layers.Conv2D(16, 3, padding='same', activation='rel
u'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='rel
u'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='rel
u'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dropout(0.2),
    layers.Dense(128),
    layers.Dense(len(data_cat))
])
```

2.2. Model Compilation:

Compiled the model with Adam optimizer and SparseCategoricalCrossentropy loss function.

```
model.compile(optimizer='adam', loss=tf.keras.losses.Sparse
CategoricalCrossentropy(from_logits=True), metrics=['accura
cy'])
```

2.3. Model Training:

Trained the model using the training and validation datasets. The training process included 10 epochs, and accuracy and loss metrics were monitored.

```
epochs_size = 10
history = model.fit(data_train, validation_data=data_val, e
```

```
pochs=epochs_size)
```

2.4. Evaluation:

Visualized the training and validation accuracy and loss to assess model performance.

```
epochs_range = range(epochs_size)
plt.figure(figsize=(8,8))
plt.subplot(1,2,1)
plt.plot(epochs_range, history.history['accuracy'], label
='Training Accuracy')
plt.plot(epochs_range, history.history['val_accuracy'], lab
el='Validation Accuracy')
plt.title('Accuracy')

plt.subplot(1,2,2)
plt.plot(epochs_range, history.history['loss'], label='Trai
ning Loss')
plt.plot(epochs_range, history.history['val_loss'], label
='Validation Loss')
plt.title('Loss')
```

2.5. Prediction:

Tested the model with a sample image to verify its prediction accuracy.

```
image = 'carrot.jpg'
image = tf.keras.utils.load_img(image, target_size=(img_hei
ght,img_width))
img_arr = tf.keras.utils.array_to_img(image)
img_bat = tf.expand_dims(img_arr, 0)

predict = model.predict(img_bat)
score = tf.nn.softmax(predict)

print('Veg/Fruit in image is {} with accuracy of {:0.2f}'.f
ormat(data_cat[np.argmax(score)], np.max(score) * 100))
```

2.6. Model Saving:

Saved the trained model for later use in the Streamlit application.

```
model.save('Image_classify.keras')
```

---

3. Web Application Development

3.1. Setting Up Streamlit:

Developed a web application using Streamlit to allow users to upload images and get real-time predictions from the model.

```
import streamlit as st
import numpy as np
from PIL import Image
import tensorflow as tf
from tensorflow.keras.models import load_model

# Custom styles with CSS
st.markdown(
    """
    <style>
    body {
        background-color: #F0F8FF;
        font-family: 'Arial', sans-serif;
    }
    .header {
        font-size: 45px;
        font-weight: bold;
        color: #FF4500;
        text-align: center;
    }
    .subheader {
        font-size: 20px;
        font-weight: bold;
        color: #2E8B57;
        text-align: center;
        margin-bottom: 30px;
    }
```

```
    .result-box {
        font-size: 25px;
        color: #2F4F4F;
        background-color: #DCDCDC;
        padding: 10px;
        border-radius: 10px;
        text-align: center;
        margin-top: 20px;
    }
    </style>
    """,
    unsafe_allow_html=True
)

# App Header
st.markdown('<p class="header">Image Classification Model</
p>', unsafe_allow_html=True)
st.markdown('<p class="subheader">Upload a Vegetable or Fru
it Image to Classify</p>', unsafe_allow_html=True)

# Load model
model = load_model('Image_classify.keras')
data_cat = [
    'Bean', 'Bitter Gourd', 'Bottle Gourd', 'Brinjal', 'Bro
ccoli', 'Cabbage',
    'Capsicum', 'Carrot', 'Cauliflower', 'Cucumber', 'Papay
a', 'Potato',
    'Pumpkin', 'Radish', 'Tomato'
]

# Image input section
img_height = 180
img_width = 180

# Image upload section
uploaded_file = st.file_uploader("Upload an Image", type=
["jpg", "png", "jpeg"])
if uploaded_file is not None:
```

```
    # Load image and display
    image_load = Image.open(uploaded_file)
    st.image(image_load, caption='Uploaded Image', width=200)

    # Preprocess image
    image_resized = image_load.resize((img_height, img_width))
    img_arr = tf.keras.preprocessing.image.img_to_array(image_resized)
    img_bat = tf.expand_dims(img_arr, 0)

    # Prediction
    predict = model.predict(img_bat)
    score = tf.nn.softmax(predict[0])

    # Display result in a visually appealing box
    st.markdown(
        f'<div class="result-box">Veg/Fruit in the image is <b>{data_cat[np.argmax(score)]}</b> '
        f'with an accuracy of <b>{np.max(score) * 100:.2f}%</b></div>',
        unsafe_allow_html=True
    )

    # Progress bar for accuracy score
    st.progress(float(np.max(score)))
else:
    st.write("Please upload an image to classify.")
```

3.2. User Interface:

- **File Upload:** Users can upload images in JPG, PNG, or JPEG formats.
- **Prediction Display:** Results are shown with accuracy scores and are styled to be visually appealing.
- **Progress Bar:** Displays the confidence level of the prediction.

Conclusion

The project successfully demonstrated the ability to classify images of vegetables and fruits using a CNN model trained on TensorFlow and deployed via Streamlit. The end-to-end solution from data preprocessing to model deployment provides a practical example of implementing machine learning in real-world applications.