FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERIG
Department of Computer Engineering

Experiment 4 - Bio Medial Image Analytics

Course Details:

Academic Year	**2023 - 24**	**Estimated Time**	**Experiment No. 4 – 02 Hours**
Course & Semester	**B.E. (COMP) – Sem. VII**	**Subject Name**	**Data Science for Healthand Social Care Lab**
Experiment Type	**Software Performance**	**Subject Code**	**HDSSBL701**

Name of Student	Atharva Pawar	**Roll No.**	9427
Date of Performance.:		**Date of Submission.:**	
CO Mapping	**HDSSBL701.4 - Use algorithms and develop models for healthcare data analytics.**		

Aim: To perform Bio Medial Image Analytics.
Objective: To detect patterns, anomalies, and features within medical images to gain insights and understand the characteristics of data.

Tools :
Jupyter Notebook/MATLAB

Libraries:
OpenCV, Numpy, Pandas, Tensor Flow

Dataset:
Various types of medical imaging modalities, such as X-rays, computed tomography (CT) scans, magnetic resonance imaging (MRI), ultrasound, and microscopy images. Datasets can be obtained from publicly available sources

Procedure:
Step 1: Data Collection and Preparation

Step 2: Feature Extraction

Image Segmentation: If the prognosis requires segmenting specific regions of interest within the images, apply image segmentation techniques to isolate these regions accurately.

Step 3: Model Selection and Training

Choose a Model: Select an appropriate machine learning or deep learning model based on the task (classification, regression) and the nature of the medical image data.

Data Splitting: Divide the dataset into training, validation, and testing subsets.

Model Architecture: Design the architecture of the chosen model, which may include convolutional neural networks (CNNs) for image-based tasks.

Training: Train the model using the training data and monitor its performance on the validation set. Use appropriate loss functions and optimization algorithms.

Step 5: Model Evaluation and Interpretation

Testing: Evaluate the trained model on the testing dataset to assess its performance on unseen data.
Metrics: Calculate relevant metrics such as accuracy, sensitivity, specificity.

Interpretation: Analyze the model's predictions and visually inspect its segmentation results to gain insights into its performance and potential areas for improvement.

Step 6: Documentation and Reporting

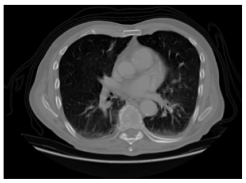
Results: Summarize the experiment's results, including model performance metrics, visualizations, and interpretations.

DSHC : Chest Cancer Detection

Sample Images | Documentation | GITHUB

Choose File | No file chosen

Upload and Predict



Uploaded Image: t1-large.cell.carcinoma.png

Predicted Class: **Large cell carcinoma**

Documentation

```
# Chest Cancer Detection - Documentation


## Project Overview

- Project Name: Chest Cancer Detection
- Dataset: [Chest CTScan Images on Kaggle](https://www.kaggle.com/datasets/mohamedhanyyy/chest-ctscan-images)
- Model Development Code: [Kaggle Notebook](https://www.kaggle.com/code/mrappplg/chest-cancer-detection/notebook)

## Team Members

- Atharva Pawar
- Aditya Vyas
- Harsh Trivedi

##  Classes for Prediction
    Adenocarcinoma: color-Red
    Large Cell Carcinoma: color-Blue
    Squamous Cell Carcinoma: color-Green
    Normal: color-Gray

## Project Description

    This project, "Chest Cancer Detection," is a mini-project in the field of data science applied to healthcare.
    The objective is to develop a deep learning model that can classify chest CT scan images into four categories:
    Adenocarcinoma, Large cell carcinoma, Squamous cell carcinoma, and Normal.
    The project aims to assist in early cancer detection and improve healthcare diagnostics.

## Problem Statement:

    - Problem:
        Chest cancer, including adenocarcinoma, large cell carcinoma, and squamous cell carcinoma, is a major health concern globally.
        Early detection is critical for effective treatment and improved patient outcomes.

    - Objective:
        Develop an automated system that can accurately classify chest CT scan images into four categories: Adenocarcinoma,
        Large cell carcinoma, Squamous cell carcinoma, and Normal, to assist medical professionals in early cancer detection.

## Abstract:

    The Chest Cancer Detection project addresses the pressing need for early cancer diagnosis through the application of deep learning in healthcare.
    The project leverages a Convolutional Neural Network (CNN) model trained on a dataset of chest CT scan images.
    This model classifies the images into four distinct categories, allowing for efficient and accurate detection of chest cancer types.
    The project has been developed and deployed as a web application for easy access and usability.

## Installations:

    ```
    pip install Flask
    pip install tensorflow
    pip install pillow
    pip install numpy

    ```
```

Usage:

    - Access the Web Application:
        Visit the deployed website (e.g., https://dshc-chest-cancer-detection.atharvapawar.repl.co/) where the Chest Cancer Detection system is hosted.

    - Upload an Image:
        On the web application, there should be a file upload form. Use it to upload a chest CT scan image that you want to classify.

    - Prediction:
        After uploading the image, the system will process it using the trained deep learning model.

    - View Results:
        The web application will display the predicted class for the uploaded image, along with the uploaded image itself.

    - Interpretation:
        Interpret the prediction result. If the model predicts a class (e.g., "Adenocarcinoma"), it means the input image is classified as that
        type of chest cancer.

    - Medical Decision Support:
        Medical professionals can use the prediction results as additional information to aid in their diagnoses. However, please note that the
        model's predictions should be considered as supportive information and not a replacement for clinical judgment.

Applications:

    - Early Cancer Detection: The primary application of the project is to assist medical professionals in detecting chest
        cancers at an early stage, which can significantly improve treatment outcomes.

    - Medical Decision Support: The model can serve as a decision support tool for radiologists and oncologists, providing
        them with additional information to make more accurate diagnoses.

    - Reduced Human Error: Automation reduces the risk of human error in the interpretation of medical images, leading to more consistent results.

    - Efficient Triage: The system can help prioritize patients, ensuring that those with a higher likelihood of cancer are assessed more urgently.

    - Telemedicine: The web application can be integrated into telemedicine platforms, enabling remote diagnosis and
        consultation for patients in underserved areas.

Limitations:

    - Data Quality: The accuracy of the model heavily depends on the quality and diversity of the training data.
        Insufficient or biased data may result in misclassifications.

    - Interpretability: Deep learning models are often considered "black boxes," making it challenging to interpret the
        rationale behind specific predictions.

    - Resource Intensive: Running deep learning models can be computationally intensive and may require substantial hardware resources,
        limiting deployment in resource-constrained environments.

Future Scope:

    - Integration with Electronic Health Records (EHR): Integrate the system with electronic health records to provide a comprehensive
        patient history, aiding in more accurate diagnoses.

    - Enhanced Explainability: Develop techniques for better model interpretability to provide insights into why a particular classification
        was made.

    - Real-time Analysis: Enable real-time analysis of streaming medical imaging data for immediate detection and response.

    - Multi-modal Data: Extend the system to handle multiple types of medical imaging data, such as X-rays and MRIs, for a broader
        range of applications.

Model Development and Training

Model Architecture

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Activation, Dropout, Flatten, Dense
from tensorflow.keras.optimizers import Adam

model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=(img_width, img_height, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4, activation='softmax'))  # Four classes

# Compile the model
```

```
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(learning_rate=0.001),
              metrics=['accuracy'])

# Model Training

history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // batch_size,
    epochs=121,  # You can adjust the number of epochs
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // batch_size)

```

CNN Layers

- Input Layer (Conv2D):
  Conv2D(32, (3, 3), input_shape=(img_width, img_height, 3))
    Explanation:
      This is the first layer of your CNN and is responsible for processing the input images.
      32 represents the number of filters or convolutional kernels to be applied in this layer.
      It means there are 32 different feature maps generated.
      (3, 3) defines the size of the convolutional kernel, which is a 3x3 filter in this case.
      input_shape=(img_width, img_height, 3) specifies the shape of the input data.
      In your case, it's images with a width and height of img_width and img_height pixels,
      and 3 channels (RGB color images).
      The Activation('relu') activation function is used to introduce non-linearity into the model.
      ReLU (Rectified Linear Unit) is a commonly used activation function in CNNs.

- MaxPooling Layer (MaxPooling2D):
  MaxPooling2D(pool_size=(2, 2))
    Explanation:
      Max pooling is a downsampling operation used to reduce the spatial dimensions of the feature maps and capture the most
      important information.
      pool_size=(2, 2) specifies that for each 2x2 region in the input, only the maximum value will be retained, reducing the
      spatial dimensions by a factor of 2.

- Convolutional Layer (Conv2D):
  Conv2D(64, (3, 3))
    Explanation:
      This is another convolutional layer with 64 filters and a 3x3 kernel size.
      The Activation('relu') activation function is applied again after this convolution to introduce non-linearity.

- MaxPooling Layer (MaxPooling2D):
  MaxPooling2D(pool_size=(2, 2))
    Explanation:
      Similar to the previous MaxPooling layer, this one further reduces the spatial dimensions of the feature maps.

- Convolutional Layer (Conv2D):
  Conv2D(128, (3, 3))
    Explanation:
    Another convolutional layer with 128 filters and a 3x3 kernel size.
    The Activation('relu') activation function is applied.

- MaxPooling Layer (MaxPooling2D):
  MaxPooling2D(pool_size=(2, 2))
    Explanation:
    Yet another MaxPooling layer to further downsample the feature maps.

- Flatten Layer (Flatten):
  Flatten()
    Explanation:
    This layer is used to flatten the 2D feature maps from the previous layers into a 1D vector. It prepares the data for the
    fully connected layers.

- Dense Layer (Fully Connected):
  Dense(128)
    Explanation:
    This is a fully connected layer with 128 neurons.
    The Activation('relu') activation function is applied to introduce non-linearity.

- Dropout Layer (Dropout):
  Dropout(0.5)
    Explanation:
    Dropout is a regularization technique used to prevent overfitting.
    It randomly sets a fraction of input units to zero during each update, in this case, 50% (0.5).

- Output Layer (Dense):
  Dense(4, activation='softmax')
    Explanation:
    This is the final output layer with 4 neurons, one for each class in your classification task.
    The activation='softmax' activation function is used for multi-class classification.
    It computes the probabilities of each class for a given input.

Deployment
  The trained model is deployed on a website accessible at https://dshc-chest-cancer-detection.atharvapawar.repl.co/.
  Users can upload chest CT scan images, and the website provides predictions for the uploaded images.

Conclusion
  The Chest Cancer Detection project demonstrates the application of deep learning in healthcare for early cancer detection.
  The model is capable of classifying chest CT scan images into four distinct categories,
```
```

providing valuable diagnostic assistance to medical professionals.

For more details, refer to the **Kaggle Notebook** containing the code used in model development.