Akademia WSB
Dąbrowa Górnicza, Kraków, Cieszyn, Żywiec, Olkusz, Gliwice, Tychy
WSB University

University WSB

Interdisciplinary Project

Neural Network part :
Using: Python-PyTorch-NumPy

Student: Meghraoui slimane 54401

Introduction to the Project

In this project, we aim to build a machine learning model that predicts water pollution levels based on a combination of tabular data (such as water occurrence and area) and satellite image data (showing surface water characteristics). We will use the powerful tools of Python, PyTorch, and NumPy to preprocess the data, build and train a predictive model, and evaluate its performance. The model will integrate both Convolutional Neural Networks (CNNs) for processing image data and fully connected layers for processing tabular data to make accurate predictions.

Why PyTorch, NumPy, and Python?

PyTorch is an ideal choice for this project because it allows for easy and flexible deep learning model development. PyTorch's dynamic computation graph and extensive library of pre-trained models make it a powerful tool for this type of problem.

NumPy is used to efficiently handle array operations and numerical tasks, which is essential when working with large datasets and performing transformations (like scaling and reshaping).

Python is the foundation of all the machine learning tasks due to its rich ecosystem and ease of use. With libraries like Pandas, Matplotlib, and NumPy, Python offers the tools necessary for data processing, model building, and evaluation.

Key Components of the Project

Data Collection and Preprocessing:

The project involves loading and combining different data sources related to water bodies. This includes:

Satellite image data (e.g., showing the occurrence of water and changes over time).

Tabular data such as historical water occurrence, water body area, and pollution levels.

We'll use Google Drive to store the data, which includes both CSV files for tabular data and PNG images for satellite imagery.

Data preprocessing involves handling missing values, normalizing numerical features, and resizing images to a consistent format for input to our machine learning model.

Modeling with PyTorch:

We build a neural network using PyTorch to predict water pollution levels based on both the image data (processed by a CNN) and tabular data (processed by fully connected layers).

The model uses the CNN architecture to extract features from the images and the fully connected network (FCN) to handle the tabular data.

After training the model, we evaluate its performance using Mean Squared Error (MSE) for regression tasks and save the trained model for future use.

Integration of Data Types:

One of the key aspects of this project is the integration of two types of data: images and tabular data. The model effectively combines these inputs to make predictions about water pollution levels.

Model Training and Evaluation:

Training involves feeding both the image data and tabular data to the model, updating the weights using the Adam optimizer, and minimizing the Mean Squared Error loss function.

After training, we evaluate the model's performance by comparing the predicted pollution levels with the actual values in the test set.

Tools and Libraries:

Python is the core programming language used for this project. It provides an excellent ecosystem of libraries for machine learning and data analysis.

PyTorch is used for building and training the neural network. It provides efficient tools for creating deep learning models and working with tensors.

NumPy is used for handling array operations and numerical computations, especially when working with the tabular data (e.g., for feature scaling and matrix operations).

Pandas is used for loading and manipulating tabular data, and Matplotlib is used for visualizing results like the annual water area and histograms.

Project Workflow

Mount Google Drive: To load the data from Google Drive where the files are stored.

Load CSV Files: These files contain historical data such as water occurrence, area, and pollution levels.

Image Preprocessing: Resize images from satellite data and normalize them to be ready for input into the CNN model.

Create a Custom Dataset Class: This class will handle the loading of both image data and tabular data efficiently.

Define the Model Architecture: Build a model with CNN layers for image data and fully connected layers for tabular data.

Training the Model: Use the Adam optimizer and MSE loss function to train the model on the data.

Evaluate the Model: Assess model performance by comparing predictions with actual labels.

Save the Model: Save the trained model to Google Drive for future use.

Conclusion

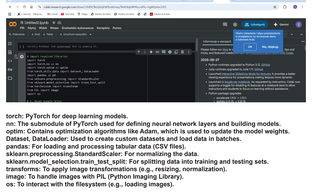
This project combines deep learning with environmental science to create a predictive model for water pollution levels using PyTorch, NumPy, and Python. By utilizing satellite imagery and historical data, we aim to provide insights into water body changes and help monitor environmental factors.

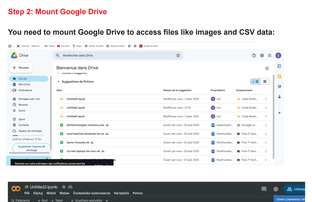
Roadmap for our project " Prediction Model":

Step 1: using google colab "gpu" + google drive :

- Import Libraries

The first part of the code imports the necessary libraries:

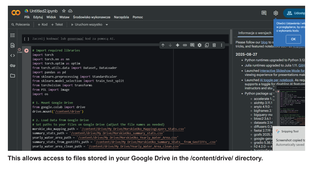




torch: PyTorch for deep learning models.
nn: The submodule of PyTorch used for defining neural network layers and building models.
optim: Contains optimization algorithms like Adam, which is used to update the model weights.
Dataset, DataLoader: Used to create custom datasets and load data in batches.
pandas: For loading and processing tabular data (CSV files).
sklearn.preprocessing.StandardScaler: For normalizing the data.
sklearn.model_selection.train_test_split: For splitting data into training and testing sets.
transforms: To apply image transformations (e.g., resizing, normalization).
image: To handle images with PIL (Python Imaging Library).
os: To interact with the filesystem (e.g., loading images).

Step 2: Mount Google Drive

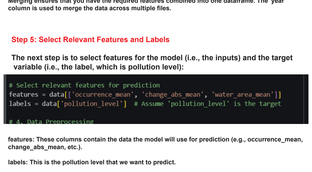
You need to mount Google Drive to access files like images and CSV data:



This allows access to files stored in your Google Drive in the /content/drive/ directory.

Step 3: Load Data from Google Drive

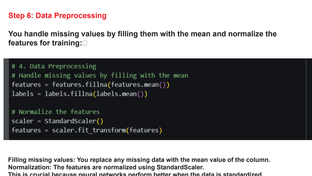
Next, you load the CSV files from Google Drive:



CSV files contain the tabular data that will be used to train the model. These files are assumed to contain various features (e.g., water occurrence, water change, etc.).

Step 4: Merge Dataframes

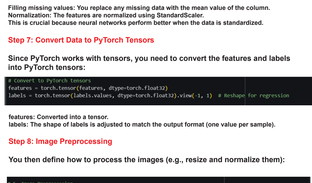
After loading the CSV files, you merge the data into a single dataframe:



Merging ensures that you have the required features combined into one dataframe. The 'year' column is used to merge the data across multiple files.

Step 5: Select Relevant Features and Labels

The next step is to select features for the model (i.e., the inputs) and the target variable (i.e., the label, which is pollution level):

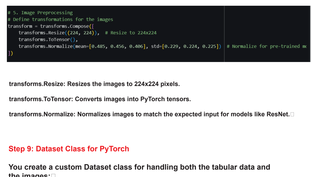


features: These columns contain the data the model will use for prediction (e.g., occurrence_mean, change_abs_mean, etc.).

labels: This is the pollution level that we want to predict.

Step 6: Data Preprocessing

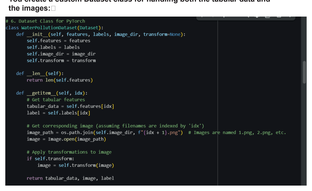
You handle missing values by filling them with the mean and normalize the features for training:



Filling missing values: You replace any missing data with the mean value of the column.
Normalization: The features are normalized using StandardScaler.
This is crucial because neural networks perform better when the data is standardized.

Step 7: Convert Data to PyTorch Tensors

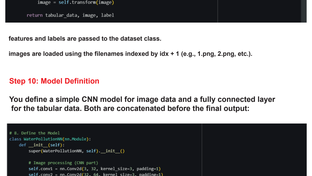
Since PyTorch works with tensors, you need to convert the features and labels into PyTorch tensors:



features: Converted into a tensor.
labels: The shape of labels is adjusted to match the output format (one value per sample).

Step 8: Image Preprocessing

You then define how to process the images (e.g., resize and normalize them):



transforms.Resize: Resizes the images to 224x224 pixels.

transforms.ToTensor: Converts images into PyTorch tensors.

transforms.Normalize: Normalizes images to match the expected input for models like ResNet.

Step 9: Dataset Class for PyTorch

You create a custom Dataset class for handling both the tabular data and the images:

features and labels are passed to the dataset class.

images are loaded using the filenames indexed by idx + 1 (e.g., 1.png, 2.png, etc.).

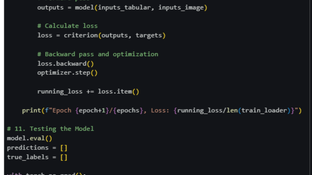
Step 10: Model Definition

You define a simple CNN model for image data and a fully connected layer for the tabular data. Both are concatenated before the final output:

CNN layers: These process the image data.

Fully connected layers: These process the tabular data and concatenate the image features before making a final prediction..

Step 11:
Model Training
Finding the Model
Save the Model
Finally, the trained model is saved to Google Drive for future use:
After training, the model is evaluated on the test set:
You train the model using MSELoss (for regression) and the Adam optimizer:



Step 12. Save the Model