

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.

▼ M Boite d M B C ៤ colab.research.google.com/drive/13tVEkTlksQsVyFbPXotAGmq7NHOXqHXP#scrollTo=hgMYpVoc2rP2 ☆ Ď S 🔡 🔀 x-kom.pl - Sklep ko... 🔪 Gmail 🔼 YouTube 💡 Mapy 😚 Adobe Acrobat 😚 Adobe Acrobat 🔇 🚱 🔼 Adobe Acrobat △ Untitled2.ipynb ☆ △ **&** Udostępnij Gemini Plik Edytuj Widok Wstaw Środowisko wykonawcze Narzędzia Pomoc Informacje o wersjach Nie, dziękuję Please follow our blog to tips and ↑ ↓ ♦ © 🗏 💠 🗓 🔟 : # Import required libraries 2025-08-27 import torch.nn as nn • Python runtimes upgraded to Python 3.12. <u>GitHub</u> import torch.optim as optim from torch.utils.data import Dataset, DataLoader ೦ಾ • Julia runtimes upgraded to Julia 1.11. GitHub Launched <u>Interactive Slideshow Mode for lectures</u>. It provides a better viewing experience for presentations making lessons more dynamic. Impore panuas as pu
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split Launched Al toggle per notebook. As requested by instructors, Colab now supports a toggle for disabling Al features at a notebook level to allow instructors and students to focus on learning without assistance. from torchvision import transforms from PIL import Image Python package upgrades import os 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).

Roadmap for our project "Prediction Model":

The first part of the code imports the necessary libraries:

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

Import Libraries

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: 🔡 | 🗙 x-kom.pl - Sklep ko... M Gmail 💶 YouTube 💡 Mapy 🚱 Adobe Acrobat 🚱 Adobe Acrobat 🕞 🚱 🔼 Adobe Acrobat 🔼 Drive Q Rechercher dans Drive 퍒 ② ﴾ ∷∷ 31 + Nouveau Bienvenue dans Drive ①

Raison de la suggestion

Modifié par vous • 01:15

Modifié par vous • 2 sept. 2025

Modifié par vous • 2 sept. 2025

Ouvert par vous • 25 août 2025

Ouvert par vous • 25 août 2025

Ouvert par vous • 25 août 2025

↑ ↓ ♦ 🖘 🗏 🗓 🗓 🗓

Propriétaire

s moi

s moi

s moi

✓ ≡ | :::

÷

:

Informacje o wersjach

Please follow our blog to s

• Python runtimes upgraded to Python 3.12

• Julia runtimes upgraded to Julia 1.11. Gith

• Launched Interactive Slideshow Mode for

2025-08-27

Colab Noteb...

Colab Noteb..

Colab Noteb...

akatechptylt...
A Partagés av...

flowithardwa...
Flow IT Hard...

flowithardwa...
Flow IT Hard...

flowithardwa...
Flow IT Hard...

flowithardwa...
Flow IT Hard...

0

2 Udostę

2,68 Go utilisés sur 15 Go X Game Consoles.xls 🕰 Augmenter l'espace de Current laptops list marc.xls

Suggestions de fichiers

CO Untitled0.ipynb

CO Untitled2.ipynb

Untitled1.ipynb

[] Zacznij kodować lub <u>generować</u> kod za pomocą AI.

from torch.utils.data import Dataset, DataLoader

Import required libraries

import pandas as pd

import torch.optim as optim

import torch

Load the CSV files

Load the CSV files

3. Merge Dataframes

X AkaTechnologies-Inventory.xlsx 🚓

martwatches wholesale list.xls 4

Accueil

• **□** Ordinateurs

☆ Suivis

Spam

a

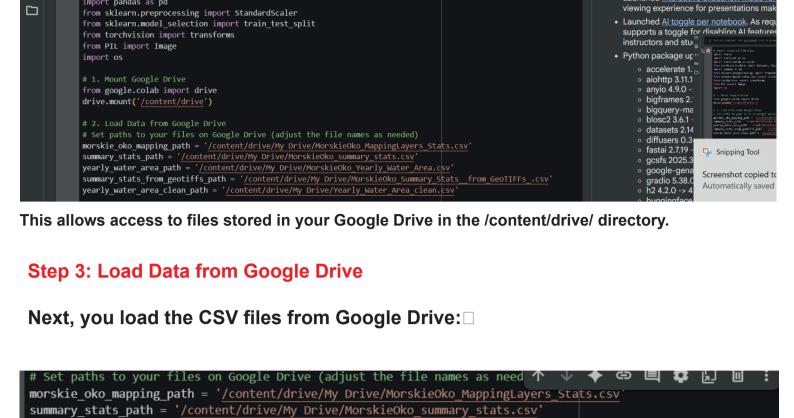
<>

☞

A Partagés avec moi

Espace de stockage

△ Untitled2.ipynb ☆ △ Plik Edytuj Widok Wstaw Środowisko wykonawcze Narzędzia Pomoc



summary_stats_from_geotiffs_path = '/content/drive/My Drive/MorskieOko_Summary_Stats__from_GeoTIFFs_.csv'

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:

yearly_water_area_path = '/content/drive/My Drive/MorskieOko_Yearly_Water_Area.csv'

yearly_water_area_clean_path = '/content/drive/My Drive/Yearly_Water_Area_clean.csv

summary_stats_from_geotiffs = pd.read_csv(summary_stats_from_geotiffs_path)

yearly_water_area_clean = pd.read_csv(yearly_water_area_clean_path)

morskie_oko_mapping = pd.read_csv(morskie_oko_mapping_path)

yearly_water_area = pd.read_csv(yearly_water_area_path)

morskie_oko_mapping = pd.read_csv(morskie_oko_mapping_path)

summary_stats_from_geotiffs = pd.read_csv(summary_stats_from_geotiffs_path)

yearly_water_area_clean = pd.read_csv(yearly_water_area_clean_path)

column is used to merge the data across multiple files.

variable (i.e., the label, which is pollution level):

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

yearly_water_area = pd.read_csv(yearly_water_area_path)

summary_stats = pd.read_csv(summary_stats_path)

summary stats = pd.read_csv(summary_stats_path)

data = pd.merge(morskie_oko_mapping, summary_stats, on='year', how='inner') data = pd.merge(data, yearly_water_area, on='year', how='inner')

change_abs_mean, etc.).

Step 6: Data Preprocessing

features for training: □

4. Data Preprocessing

Normalize the features scaler = StandardScaler()

Step 5: Select Relevant Features and Labels The next step is to select features for the model (i.e., the inputs) and the target

Merging ensures that you have the required features combined into one dataframe. The 'year'

```
# Select relevant features for prediction
features = data[['occurrence_mean', 'change_abs_mean', 'water_area_mean']]
labels = data['pollution_level'] # Assume 'pollution_level' is the target
    Data Preprocessing
```

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

You handle missing values by filling them with the mean and normalize the

Filling missing values: You replace any missing data with the mean value of the column.

labels = torch.tensor(labels.values, dtype=torch.float32).view(-1, 1) # Reshape for regression

labels: The shape of labels is adjusted to match the output format (one value per sample).

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

This is crucial because neural networks perform better when the data is standardized.

features = features.fillna(features.mean()) labels = labels.fillna(labels.mean())

features = scaler.fit transform(features)

Step 7: Convert Data to PyTorch Tensors

features = torch.tensor(features, dtype=torch.float32)

transforms.Resize((224, 224)), # Resize to 224x224

Handle missing values by filling with the mean

Normalization: The features are normalized using StandardScaler.

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

features: Converted into a tensor.

Step 8: Image Preprocessing

Define transformations for the images transform = transforms.Compose([

Step 9: Dataset Class for PyTorch

def __init__(self, features, labels, image_dir, transform=None):

the images:

6. Dataset Class for PyTorch

class WaterPollutionDataset(Dataset):

self.features = features

Image Preprocessing

transforms.ToTensor(),

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.

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

transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]) | # Normalize for pre-trained mc

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 Testing 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: for epoch in range(epochs): model.train() running loss = 0.0 for inputs_tabular, inputs_image, targets in train_loader: # Zero gradients optimizer.zero_grad() # Forward pass outputs = model(inputs_tabular, inputs_image) # Calculate loss loss = criterion(outputs, targets) # Backward pass and optimization loss.backward() optimizer.step() running_loss += loss.item()

self.labels = labels self.image_dir = image_dir self.transform = transform def __len__(self): return len(self.features) def __getitem__(self, idx): # Get tabular features tabular data = self.features[idx] label = self.labels[idx] # Get corresponding image (assuming filenames are indexed by 'idx') image = Image.open(image_path) # Apply transformations to image if self.transform: image = self.transform(image) return tabular_data, image, label 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 for the tabular data. Both are concatenated before the final output: # 8. Define the Model class WaterPollutionNN(nn.Module): def init (self): super(WaterPollutionNN, self).__init__() # Image processing (CNN part) = nn.Conv2d(3,32, kernei self.conv2 = nn.Conv2d(32, 64, kernel_size=3, padding=1)

image_path = os.path.join(self.image_dir, f"{idx + 1}.png") # Images are named 1.png, 2.png, etc. You define a simple CNN model for image data and a fully connected layer self.pool = nn.MaxPool2d(kernel_size=2, stride=2) # Fully connected part for tabular data self.fc_tabular = nn.Linear(3, 32) # 3 input features # Final fully connected layer self.fc1 = nn.Linear(64*56*56 + 32, 128) # 64*56*56 is the output of the CNN, plus the tabular features self.fc2 = nn.Linear(128, 1) # Output one value (pollution level) def forward(self, x_tabular, x_image): # Image processing x_image = self.pool(torch.relu(self.conv1(x_image))) x_image = self.pool(torch.relu(self.conv2(x_image))) $x_{image} = x_{image.view}(x_{image.size}(0), -1)$ # Flatten # Tabular processing x_tabular = torch.relu(self.fc_tabular(x_tabular)) # Concatenate image and tabular features x = torch.cat((x_image, x_tabular), dim=1) # Fully connected layers x = torch.relu(self.fc1(x)) x = self.fc2(x)return x

print(f"Epoch {epoch+1}/{epochs}, Loss: {running_loss/len(train_loader)}") # 11. Testing the Model model.eval() predictions = [] true_labels = [] with torch.no grad(): for inputs_tabular, inputs_image, targets in test_loader: outputs = model(inputs_tabular, inputs_image) predictions.append(outputs) true_labels.append(targets) # Flatten the predictions and true labels for comparison predictions = torch.cat(predictions).numpy() true_labels = torch.cat(true_labels).numpy() # You can print or visualize the predictions and true labels print("Predictions: ", predictions) print("True Labels: ", true_labels) # 12. Save the Model torch.save(model.state_dict(), '/content/drive/My Drive/water_pollution_model_wi