Beam Prediction Based on Multi-Modal Fusion (in Python)

!!! In-Progress for a journal paper heading to IEEE Transactions on Vehicular and Technology.

[Paper] | [Code] | [Video]

Overview

This project starts Oct, 2023, featured as a Python-coded deep learning project, developed as my bachelor thesis and a journal paper heading to IEEE Transactions mentored by Prof. Xuanheng Li. Over 2 weeks, I focused on integrating radar and image data using PyTorch, designing neural network architectures, and implementing efficient data preprocessing routines. My role extended to tuning hyperparameters aiming to receive better accuracy of our proposed scheme for multi-modal data. For more insights into the project, visit the repository here.

@timeFrame: Nov 13 - 27, 2023 (2 wks)
@role: Developer
@skills: Python, PyTorch, PyCharm, Data Preprocessing, DeepSense6G Dataset
@team: Prof. Xuanheng Li, **Yanhao Bai**, Dongxu Ge, Sike Cheng

The Problem

The future of vehicular communication networks relies on mmWave massive multi-input-multi-output antenna arrays for intensive data transfer and massive vehicle access; however, reliable V2I links require narrow beam alignment, which traditionally involves excessive signaling overhead. To address this issue, existing methods rely solely on communication processing; my team noticed a need to improve beam alignment accuracy, by obtaining comprehensive environmental features.

The Goal

For this 2-week development sprint, my team's solution was a novel proactive beam prediction scheme that integrates multi-modal sensing and communications via Multi-Modal Fusion utilizing deep learning approaches, which is composed of multiple neural network components with distinct functions. Our proposed Multi-Modal Fusion scheme achieves more accurate and stable beam prediction; even in complex dynamic scenarios, robust prediction results can be guaranteed (for this, we enlarged the existing dataset by synthesizing adverse weather conditions to RGB data, for more details, see my project: Adverse Weather Synthetization on Images Based on GPT-4), demonstrating the feasibility and practicality of our proposed proactive beam prediction approach.

Features

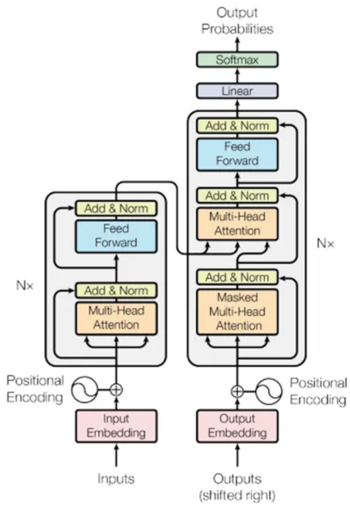
1. Preprocessing: Utilizes pandas for data manipulation, torchvision transforms for img preprocessing.
2. Architecture: Implements custom neural network models of encoder and attention classes, integrating multi-head attention and positional encoding to process sequential data effectively.
3. Data Handling: Employs PyTorch's DataLoader and custom dataset class for efficient data loading and batching, enabling streamlined handling of large datasets.
4. Extraction: Utilizes pre-trained ResNet model for image feature extraction and Fourier transforms for radar data processing.
5. Training: Orchestrates the training process with backpropagation and validation via cross-entropy loss and accuracy metrics, while efficiently utilizing GPU resources for computation.

How it works

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

```
Epoch 49/50, Train Loss: 0.0174, Validation Loss: 2.3977,ACC: 0.2669
Top-3 Accuracy: 55.8277027027027
Top-5 Accuracy: 71.19932432432432
```

```
Epoch 4/50, Train Loss: 0.0076, Validation Loss: 1.1905,ACC: 0.5794
Top-3 Accuracy: 90.70945945945945
Top-5 Accuracy: 97.55067567567568
```



Sample Code

Code snippet, featured as core of positional encoding module in a transformer model, crucial for understanding the sequential aspect of input data

```
class PositionalEncoding(nn.Module):
    def __init__(self, d_model, dropout, max_len=5000):
        super(PositionalEncoding, self).__init__()
        self.dropout = nn.Dropout(p=dropout)
        # Initialize positional encoding with shape (max_len, d_model)
        pe = torch.zeros(max_len, d_model)
        position = torch.arange(0, max_len).unsqueeze(1)
        div_term = torch.exp(torch.arange(0, d_model, 2) * -(math.log(10000.0) / d_model))
        pe[:, 0::2] = torch.sin(position * div_term)
        pe[:, 1::2] = torch.cos(position * div_term)
        pe = pe.unsqueeze(0)
        self.register_buffer("pe", pe)

    def forward(self, x):
        x = x + self.pe[:, : x.size(1)].requires_grad_(False)
        return self.dropout(x)
```

Adverse Weather Synthetization on Images Based on GPT-4 (in Prompts)

!!! Successfully deployed GPT-4 API for image processing, totally Cycle/WeatherGAN-free.
[Paper] | [Code] | [Video]

Overview

This project starts Nov, 2023, featured as blended-language-coded image augmentation project, deploying most advanced GPT-4 API and prompts. Over 2 weeks, I focused on encompassing existing scripts and models that manipulate images to simulate various adverse weather conditions on images serving as enlarged dataset for our proposed beam prediction scheme mentored by Prof. Xuanheng Li. My role extended to writing prompts for GPT-4 to learn and deploy the above image augmentation. For more insights into the project, visit the repository here.

@timeFrame: Oct 16 - 30, 2023 (2 wks)
@role: Developer
@skills: Python, MATLAB, OpenCV, TensorFlow, GPT-4, Image Processing and Augmentation
@team: Prof. Xuanheng Li, **Yanhao Bai**

The Problem

The task of realistically simulating different adverse weather conditions (rain, snow, fog, dusk, etc.) on images can be quite complex, by utilizing image processing methods, such as CGAN, CycleGAN, WeatherGAN. To address this issue, my team noticed a need to adopt a generative scheme with the innovative use of GPT-4 prompts and APIs for intelligent automation in image processing.

The Goal

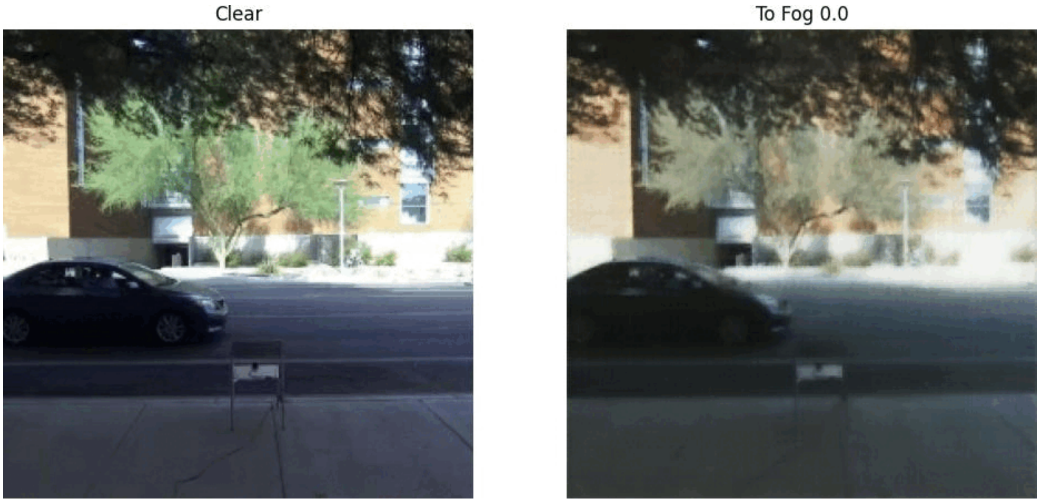
For this 2-week development sprint, my team's solution was a comprehensive multi-tool framework then prompts for worsening images with an array of adverse weather conditions and color alterations. We ended up with successful experiments that demonstrated the effectiveness of our emphasis on deploying GPT-4 to scale up capabilities of image augmentation and processing while minimizing manual intervention, thus offering a novel, automated solution to creating diverse and enhanced datasets for AI-driven, ML-aided applications.

Features

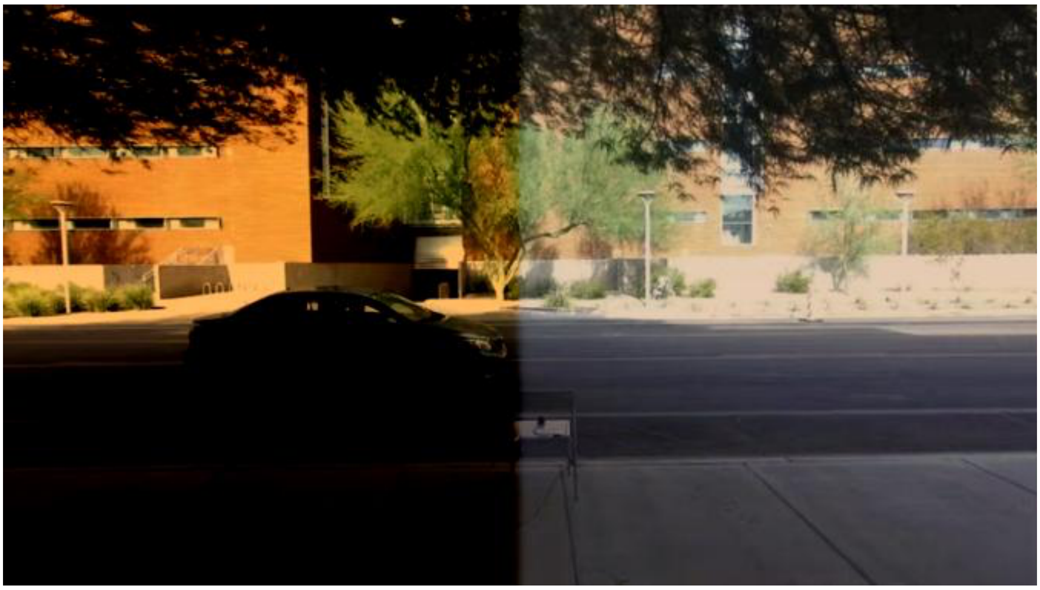
1. Image Augmentation: Advanced python scripts that apply various effects like fog, rain, shadows, sun flare, motion blur, and autumnal changes to images.
2. CycleGAN: TensorFlow-based fog simulation.
3. WeatherGAN: MATLAB-based color transfer for diverse visual effects.
4. GPT-4: Deployment of GPT-4 for interpreting and executing the above image processing tasks based on textual prompts and APIs, showcasing an LLM-assisted approach to cv.
5. Tool Versatility: Use of Python, MATLAB, OpenCV, TensorFlow, and most significantly, GPT-4, demonstrating multifaceted ways and a novel approach to image processing.

How it works

CycleGAN foggy:



WeatherGAN dusky:



GPT-4 all effects:

Sample framework

3. Raindrop Image Formation

We model a raindrop degraded image as the combination of a background image and effect of the raindrops:

$$\mathbf{I} = (1 - \mathbf{M}) \odot \mathbf{B} + \mathbf{R} \tag{1}$$

where $\mathbf{I}$ is the colored input image and $\mathbf{M}$ is the binary mask. In the mask, $\mathbf{M}(\mathbf{x}) = 1$ means the pixel $\mathbf{x}$ is part of a raindrop region, and otherwise means it is part of background regions. $\mathbf{B}$ is the background image and $\mathbf{R}$ is the effect brought by the raindrops, representing the complex mixture of the background information and the light reflected by the environment and passing through the raindrops adhered to a lens or windscreen. Operator $\odot$ means element-wise multiplication.

Raindrops are in fact transparent. However, due to their shapes and refractive index, a pixel in a raindrop region is not only influenced by one point in the real world but by the whole environment [25], making most part of raindrops seem to have their own imagery different from the background scene. Moreover, since our camera is assumed to focus on the background scene, this imagery inside a raindrop region is mostly blur. Some parts of the raindrops, particularly at the periphery and transparent regions, convey some information about the background. We notice that the information can be revealed and used by our network.

Based on the model (Eq. (1)), our goal is to obtain the background image $\mathbf{B}$ from a given input $\mathbf{I}$. To accomplish this, we create an attention map guided by the binary mask $\mathbf{M}$. Note that, for our training data, as shown in Fig. 5, to obtain the mask we simply subtract the image degraded by raindrops $\mathbf{I}$ with its corresponding clean image $\mathbf{B}$. We use a threshold to determine whether a pixel is part of a raindrop region. In practice, we set the threshold to 30 for all images in our training dataset. This simple thresholding is sufficient for our purpose of generating the attention map.

Sample prompt

PROMPT: learn the following code (alleffects.py), it adds adverse weather conditions on the imgs

GPT-4: Thank you for sharing this Python code. It appears to be a comprehensive script for applying various weather and environmental effects to images using the OpenCV library. The code includes functions for simulating rain, snow, fog, gravel, sun flare, motion blur, autumn effects, and more. These functions modify image properties like brightness, color saturation, and overlay patterns to create realistic effects.

PROMPT: you’re totally right! It adds rain, snow, fog, gravel, sun flare, motion blur, autumn effects on imgs. Now apply what you have learnt on my image (test.jpg), i.e., add rain, snow, fog, gravel, sun flare, motion blur, autumn effects on my original image. Note: simply add all effects on my original image, instead of creating any fake, imaginary images.

GPT-4: I have applied all effects to your original image. You can download the processed image with the adverse weather conditions from the following link: Download the processed image.

Multi-Modal 6G Sensing and Communication Dataset Walkthrough (in Python)

!!! A large-scale real-world multi-modal isac dataset for 6G deep learning research.

[Slides] | [Code] | [Video]

Overview

This project starts Sep, 2023, featured as a Python-coded dataset walkthrough project, to perform the essential data loading and visualization tasks on all modalities - Power, RGB images, GPS positions, Radar, and Lidar, in the DeepSense6G Dataset. Besides tasks on the main modality data, the project has acquired other relevant information such as timestamp, number of connected satellites, etc., and performed annotations such as sequence index, direction of movement, obj detection bounding boxes. The project also shows how to access and leverage the metadata and annotations to visualize and understand propagation patterns. For more insights into the project, visit the repository here.

The Problem

In the burgeoning field of 6G research, a critical challenge lies in analyzing and visualizing complex, multi-modal datasets that are quintessential for understanding real-world sensing and communication dynamics. The intricacies of such datasets include diverse data types and formats, necessitating a versatile approach to data processing and interpretation. Traditional data handling methods fall short in providing the necessary depth and clarity for insights essential in 6G technology development.

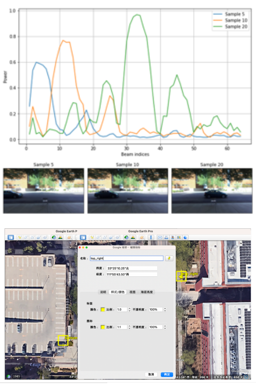
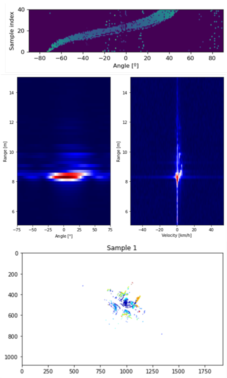
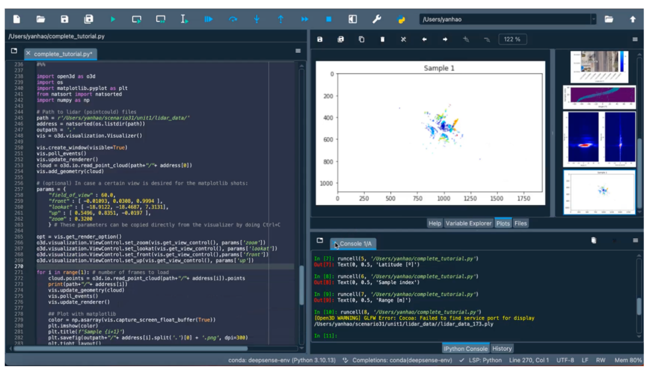
The Goal

This project aims to develop a robust framework for the comprehensive analysis and visualization of multi-modal datasets pertinent to 6G deep learning research. By leveraging advanced py libraries and visualization tools, the project aspires to not only streamline the processing of data of Power, RGB, GPS, Radar, and Lidar but also enhance understanding of data through intuitive visualizations.

Features

1. Data Processing: Uses Python library Pandas for efficient data manipulation and management.
2. Visualization: Employs Matplotlib and Open3D for visual representation of various data types.
3. Multi-Modal Data Handling: Integrates various data forms including Power, RGB images, GPS, Radar, and Lidar data, ensuring a holistic approach to data analysis.
4. Interactive Analysis: Provides tools for dynamic data exploration, enabling users to interact with and derive insights from complex datasets.

How it works

Sample Code

```
# Code snippet, featured as utilizing open3d library to load and visualize point cloud data, by setting up a 3D visualization window, loading point cloud files, and adjusting the display for custom view
import open3d as o3d
import os
from natsort import natsorted
import numpy as np

# Path to lidar (point cloud) files
path = '/Users/yanhao/scenario31/unit1/lidar_data/'
address = natsorted(os.listdir(path))
vis = o3d.visualization.Visualizer()
vis.create_window(visible=True)
vis.poll_events()
vis.update_renderer()
cloud = o3d.io.read_point_cloud(path + "/" + address[0])
vis.add_geometry(cloud)

# Setting custom view parameters
params = {
    "field_of_view": 60.0,
    "front": [-0.01093, 0.0308, 0.9994],
    "lookat": [-18.9122, -18.4687, 7.3131],
    "up": [0.5496, 0.8351, -0.0197],
    "zoom": 0.3200
}

vis.get_view_control().set_front(params['front'])
vis.get_view_control().set_lookat(params['lookat'])
vis.get_view_control().set_up(params['up'])
vis.get_view_control().set_zoom(params['zoom'])

for file_name in address:
    if file_name.endswith('.ply'):
        pcd = o3d.io.read_point_cloud(os.path.join(path, file_name))
        vis.add_geometry(pcd)
        vis.poll_events()
        vis.update_renderer()

vis.destroy_window()
```

UAV Machine Learning Aided Integrated Sensing & Communication (in MATLAB)

!!! Published a China National Patent on Jun 12, 2023; awarded as China National Innov. Project.
[Paper] | [Slides] | [Code] | [Video]

This project starts Dec, 2021, featured as a Matlab-coded integrated sensing and communication (isac) project on SDR devices, developed as my research project for China Innovation & Entrepreneurship Program for College Students in Dec, 2022, mentored by Prof. Xuanheng Li. Over 1 year, I focused on developing a unified system capable of executing communication and sensing tasks simultaneously, leveraging Matlab for programming and gnuradio for modular programming. My role extended to proposing an innovative method for UAV intelligent trajectory planning and resource allocation, using reinforcement learning (deep q-learning) to further demonstrate the effectiveness of our built platform.

The Problem (Part 1)

The future of 6G systems relies on performing wireless communication and environmental sensing simultaneously, bridging gaps between the separation of the two modalities; such separation between sensing and communication mainly attributes to the differing requirements of the two (sensing looks for accuracy and resolution, while communication aims for max. of spectral efficient and data rates). With existing advancements of massive antenna arrays and high-frequency bands, my team noticed a pressing need to merge the two fields of functionalities into one single, efficient isac system.

The Problem (Part 2)

After building up an isac platform resolving the problem, how to ensure efficient resource allocation and trajectory planning for UAVs in dynamic and complex isac networks is another key issue. For this, my team noticed an urging need for adaptable strategies in response to the changing isac demands.

The Goal

For this 1-year development marathon, my team' solution was a novel and comprehensive system that can send and receive ofdm signals while detecting environmental changes using doppler radar at the same time, which fully utilized the inherent properties of the two modalities, achieving high data rates and accurate sensing, demonstrating the feasibility of integrated sensing and communication on a single platform, providing a foundation for enabling UAVs to autonomously and intelligently formulate optimal strategies for trajectory planning and resource allocation under varying environs.

Features

1. Communication (OFDM): Uses Matlab to develop a robust ofdm-based communication system, incorporating frame synchron, frequency offset correction, channel equalization, while using signal processing for data modulation, demodulation, error correction, to ensure reliable data transmission.
2. ISAC: Integrates the communication and sensing systems on the Pluto-SDR platform, showcasing the potential of isac technology in real-world applications.
3. Reinforcement Learning (DQN): Implements a deep q-learning algorithm for UAVs, enabling them to autonomously adapt trajectories and resource distribution based on real-time environmental data.
4. Trajectory Planning / Dynamic Resource Allocation: Allows drones to optimize their flight paths to balance communication needs and sensing accuracy; allows drones to allocate bandwidth and power resources dyna-mically, enhancing the overall efficiency of the isac network.

5. Off/Online Learning: Incorporates the capability for drones to learn from historical or simulated data offline and refine strategies online in real environments.
6. Adaptive Strategy: Contributes to drones' ability to formulate, update complex coupling strategies for power and bandwidth allocation, adapting to the specific demands of the isac environment.

Sample framework

Initialization:

- Set size of experience pool as G, size of mini-batch sampling as S, the update period as J, the discount factor as γ, the learning rate as α, the greedy factor as ε= 0.9, the parameters of the Q-main-network as θ, the parameters of the Q-target-network as $\hat{\theta}$, the initial state as s_0= 1

Repeat:

- Formulate a shared strategy α_t based on the *ε-greedy* policy in the current state s_t
- Move to the next state s_{t+1} and calculate the reward value γ_{t+1}
- Collect experience tuple info: current state s_t, action α_t, next state s_{t+1}, reward value γ_{t+1}, then store this experience tuple info in the memory unit
- $t \leftarrow t+1$
- $\varepsilon \leftarrow 0.9 \times 0.1^{t/10000}$
- If $\varepsilon <= 0.1$

  $\varepsilon = 0.1$
- If $t > G$

  Update the experience tuple
  Sample randomly S numbers of experience tuples to train the network
  Calculate loss function $L(\theta)$, perform a gradient descent, update Q-main-network's θ
  If $t - G \bmod J = 0$

    Copy Q-main-network's parameters θ to Q-target-network's parameters $\hat{\theta}$