Multi-View Geo-Localization

Prepared for: Dr. Ali Ghandour

By:
Ahmad Sidani

Maria Hussien

Mariah Sandakli

Project Overview

Multi-view geo-localization is a key research area focused on determining the geographic location of a place captured from one viewpoint—such as a drone or ground-level image—by matching it against a reference database of satellite or aerial imagery. This project aims to bridge the visual gap between different perspectives by leveraging drones as a rich source of imagery and incorporating large-scale, multi-source datasets. Through this approach, it enables the development of robust and scalable geo-localization systems capable of accurately operating in diverse and dynamic real-world environments.

Adopted Solution: MBEG from ACMMM 2023

This project adopts and builds upon the solution presented in the following GitHub repository:
https://github.com/Reza-Zhu/ACMMM23-Solution-MBEG/

Explanation:

The adopted method, known as **MBEG (Multi-Branch Embedding Guidance)**, was introduced at ACM Multimedia 2023. It addresses cross-view geo-localization by proposing a framework that separates and leverages multi-modal cues across different branches. MBEG combines multiple visual perspectives (e.g., satellite, drone, and ground views) into a unified embedding space using a series of parallel branches and shared projection heads.

Key components of MBEG include:

- **Multi-Branch Design**: Each branch processes a specific modality (e.g., drone-view or satellite-view) and captures modality-specific patterns.
- **Shared Embedding Space**: Features from different modalities are projected into a shared latent space using contrastive learning objectives, enabling robust retrieval across views.
- **Global-Local Matching**: The model performs both global and local matching by incorporating attention mechanisms and feature alignment, enhancing its ability to match partial or occluded views.
- **Contrastive and Instance Loss**: MBEG optimizes its embeddings using a combination of losses to reinforce discriminative learning.

By adapting the MBEG framework, this project benefits from its ability to generalize across environments and views, while customizing it to drone-to-satellite localization

using simulated environmental conditions and large-scale data from the University-1652 dataset.

Original Codebase

Preprocessing

- Reads **`settings.yaml`** for parameters including `classes`, `batch_size`, `dataset_path`, and `image_size`.
- Defines **data transforms**:
  - **Training (drone)**: Resize → Pad → RandomCrop → RandomPerspective → RandomHorizontalFlip → ToTensor → Normalize.
  - **Satellite**: Similar pipeline but includes `RandomAffine(90)` to augment orientation.
- Loads folders `train/drone` and `train/satellite` using `torchvision.datasets.ImageFolder` and returns dataloaders for both, with shuffling on training.

Model

- Implements a class (e.g., `EVA`) combining **two branches** via **weight-sharing** ViT-based backbones (`eva02_large_patch14_448`) from the `timm` library.
- If `share_weight = True`, both drone and satellite branches use the same backbone. A classification "head" (`ClassBlock`) maps final embeddings to geo-tag classes.

Train

- Trains the model using data from both `drone` and `satellite` domains via the two-branch architecture.
- Employs a **classification loss** (instance loss) using a shared classifier to enforce embedding alignment.
- Weights the backbone learning rates separately: a typical setup uses a lower learning rate for pre-trained visual models and a higher rate for the classification head. (While code-specific hyperparameters weren't retrieved, README and

issues imply an optimized learning schedule aligned with published results.)

Test and Evaluate

- Loads the trained model and evaluates on **University-1652 test split**.
- Uses an **`extract_feature`** function similar to the prediction script:
  - Inputs are flipped horizontally for augmentation before processing in the appropriate branch.
  - Features are **ℓ_2-normalized** (optionally block-wise if using LPN).
- Computes **pairwise cosine similarity** between drone query and satellite gallery embeddings.
- Evaluates **Recall@K** and **mean Average Precision (mAP)** using the `compute_mAP()` helper, following a standard retrieval evaluation: sort scores, filter out junk samples, build CMC curves, and integrate PR area.

Modification Progress

1. Configuration file (settings.yaml)

The primary modifications were made in the `setting.yaml` configuration file to suit our experimental setup. Specifically, we updated the dataset path to point to our custom dataset location, changed the directory where model weights are saved to better organize outputs, adjusted the batch size and training epochs to fit our hardware constraints, and renamed the model identifier to reflect our specific experiment.

---

2. Creation of a Training Subset from the U1652-University Dataset

To tailor the dataset for efficient training and evaluation, we created a subset of the original U1652-University dataset. This subset was generated by randomly selecting 100 classes that are common to both drone and satellite image modalities. For each selected class, we then randomly sampled 20 drone images, ensuring balanced representation per class. The matching satellite image corresponding to those classes were also included entirely.

---

3. Data Preprocessing (Modified_Preprocessing.py)

The original preprocessing script was designed for handling multi-view geo-localization data. To accommodate the subset dataset we created—where each class contains one satellite image paired with multiple drone images—we developed a customized preprocessing class in PyTorch called `PairedU1652Dataset`.

Features and Modifications:

- **Paired Multi-View Data Handling**:

  Unlike a standard dataset where each sample is a single image, our dataset consists of pairs of images from different modalities: one satellite image and multiple drone images per class. The dataset class loads these pairs accordingly. It assumes that each class folder under the satellite directory contains exactly one satellite image (verified by an assertion), and the corresponding drone directory contains multiple drone images for that same class.

- **Directory Structure Traversal:**

  The dataset constructor (__init__) traverses both the satellite and drone root directories. It lists all class folders and maps each class name to an integer label (class_to_idx). For each class, it collects the satellite image path and all drone image paths. Since there is only one satellite image per class, its path is repeated for every drone image in that class to maintain paired indexing.

- **Data Lists for Efficient Access:**

  Three parallel lists are created:
  self.sat_images: repeated satellite image paths for each drone image
  self.drone_images: drone image paths
  self.labels: class labels corresponding to each pair

- **On-the-Fly Image Loading and Transformation:**

  The __getitem__ method loads the paired images using the Python Imaging Library (PIL) and converts them to RGB. Finally, it returns a tuple (sat_img, drone_img, label) for training or evaluation.

---

4. Model definition (dualresnet.py)

The original model used in the ACMMM23 solution was based on the EVA (Efficient Vision Architecture) backbone, a transformer-based architecture that processes satellite and drone images through separate but coordinated branches. Due to hardware limitations, this was replaced with a lightweight Dual ResNet-18 architecture. Both models share key similarities in their overall approach: they employ a dual-stream design where satellite and drone images are processed independently, extract modality-specific features, and use separate classification heads. This setup allows the models to learn view-specific representations and facilitates applications such as cross-view geolocalization and classification. However, the main difference lies in the backbone choice. While the EVA model leverages the high-capacity, transformer-based EVA or EVA-CLIP backbone—requiring substantial compute resources and enabling flexible input resolution—the Dual ResNet-18 model uses two parallel CNN-based ResNet-18 backbones, which are far more efficient and suitable for constrained environments. Each ResNet branch outputs a 512-dimensional feature vector, which is then passed to a simple linear classifier. The model returns both class logits and intermediate features, supporting both classification and metric learning objectives. This dual-stream CNN architecture retains the core functionality of the original EVA-based approach while significantly reducing the computational cost, making it suitable for training and experimentation on modest hardware.

Model Architecture of Dual-ResNet18:

🔷 ***Backbone Networks***

- Two parallel ResNet-18 models are used:

  backbone1 processes satellite images.

  backbone2 processes drone images.

- The final fully connected (fc) layers of both backbones are replaced with identity layers (nn.Identity()) to output raw 512-dimensional feature embeddings.

🔷 ***Classification Heads***

- Each backbone is followed by a separate linear classifier:

  classifier1 predicts the class label based on satellite features.

  classifier2 predicts the class label based on drone features.

🔷 ***Output***

- The model returns four values during the forward pass:

  out1: class prediction for the satellite image.

  out2: class prediction for the drone image.

  f1: feature embedding from the satellite backbone.

  f2: feature embedding from the drone backbone.

---

5. Adaptation of Training Pipeline to DualResNet Architecture

Overview

The original training pipeline, as implemented in the official MBEG repository, was adapted to accommodate the architectural and resource constraints introduced by the DualResNet model. While the core training strategy remained consistent—supervised classification using paired satellite and drone views—the implementation was refined to suit the modified network and simplified hardware setup.

Key Adaptations and Design Consistency

- **Preservation of Dual-Modality Supervision**
  The dual-branch training structure remains intact: satellite and drone image pairs are processed in parallel, with both branches supervised using cross-entropy loss against shared ground truth class labels.
- **Modified Backbone Architecture**
  The original EVA-02 transformer backbone was replaced by two lightweight ResNet-18 models—one for satellite images and another for drone images. This change significantly reduced computational demand while maintaining the representational power needed for learning modality-specific features.
- **Feature Extraction and Classification**
  Each ResNet-18 model has its final fully connected (fc) layer replaced with an identity layer to extract 512-dimensional feature embeddings. These embeddings are passed through separate linear layers (classifiers) to produce classification logits for each modality.

Dataset and Preprocessing Pipeline

- **Custom Paired Dataset**
  A new `PairedU1652Dataset` class was implemented to load synchronized satellite-drone image pairs. This dataset structure mirrors the organization of the original U1652 dataset and ensures balanced exposure to both modalities during training.
- **Image Transformations**
  Both satellite and drone images undergo standard preprocessing steps: resizing to 224×224 pixels, normalization using ImageNet mean and standard deviation, and conversion to PyTorch tensors using `torchvision.transforms`.

Training Procedure

- **Loss Computation and Optimization**
  The model computes predictions from both branches (`out1` for satellite and `out2` for drone), and computes cross-entropy loss separately for each. The average of both losses is backpropagated in a single optimization step. The Adam optimizer is used for parameter updates.
- **Accuracy Monitoring and Model Checkpointing**

  For each epoch, classification accuracy was tracked independently for both the satellite and drone branches. The final average accuracy per epoch was computed as the mean of both accuracies. The model achieving the highest average accuracy across epochs was saved as a .pth file in the designated weights directory. In the current training run, the best model achieved an average classification accuracy of 75%, reflecting strong modality-consistent performance.

- **Hardware Consideration**
  The modified pipeline is designed to operate in CPU-only environments (e.g., Google Colab without GPU), making it more accessible for constrained training scenarios.

---

6. Creating a subset of U1652 testing dataset

To evaluate the modified DualResNet model, a **test subset** was systematically created from the original U1652 testing dataset. This subset consisted of **100 randomly selected classes** that were present in both modalities — **query_satellite** and **query_drone** — ensuring that each selected class had images from both views. The creation process involved:

- Identifying the intersection of classes available in both satellite and drone query folders to maintain consistency.
- Randomly sampling 100 classes (or fewer if less than 100 were available) from this shared set.
- For each selected class, copying **all images** from both satellite and drone query folders into a new structured test subset directory, preserving the class-wise folder hierarchy.

This preparation ensured the test dataset mirrored the training data's dual-modality structure, allowing the model to be evaluated in a consistent and realistic setting.

---

7. Testing and Evaluation Methodology

The testing and evaluation procedure was adapted from the original codebase to align with the custom DualResNet architecture and the specific paired dataset structure used in this study. Several modifications were implemented to accommodate these changes while preserving the fundamental evaluation principles.

Dataset Preparation and Loading:

The test dataset consists of two modalities: satellite images and drone images, organized into separate folders but containing the same classes. A subset of 100 classes was carefully sampled to ensure balanced representation and to match the paired input requirement of the DualResNet model. For each modality, images are loaded using PyTorch's `ImageFolder` dataset wrapper, which reads images grouped by class folders. The images undergo a standardized preprocessing pipeline involving resizing to 224×224 pixels, normalization with ImageNet mean and standard deviation values, and tensor conversion to ensure consistency with the training data format.

Two independent dataloaders are created for the satellite and drone test sets respectively, each providing batches of images without shuffling to maintain order during evaluation.

Feature Extraction:

During testing, the DualResNet model operates in evaluation mode with its learned weights loaded from the best checkpoint saved during training. The model extracts feature embeddings from each modality branch independently:

- The satellite images pass through the first backbone branch, producing a set of normalized feature vectors.
- The drone images pass through the second backbone branch, similarly producing normalized features.

Normalization of these feature vectors to unit length (L2 normalization) is critical to ensure that cosine similarity calculations are meaningful and comparable across features.

Similarity Computation and Retrieval:

The core evaluation task is framed as a retrieval problem, where the goal is to find, for each satellite query image, the most similar drone images belonging to the same class. This setup reflects the real-world application of cross-modality matching between satellite and drone imagery.

For each satellite query feature, cosine similarity scores are computed against all drone gallery features. These scores are then sorted to create a ranked retrieval list of drone images, ordered from most to least similar.

Evaluation Metrics:

Two widely accepted metrics from retrieval and re-identification literature are employed:

1. **Mean Average Precision (mAP):**
   mAP quantifies the overall quality of the ranked retrieval list by averaging the precision scores obtained at positions where relevant (correct-class) images are found. It effectively balances precision and recall, reflecting how well the model retrieves correct matches throughout the list.
2. **Cumulative Matching Characteristic (CMC):**
   CMC focuses on recall at specific ranks, reporting the probability that at least one correct match is retrieved within the top-k candidates. For example, Recall@1 indicates the chance of the correct match being the topmost retrieval, while Recall@5 extends this to the top five.

The evaluation function computes these metrics by identifying "good" indices (correct class matches) and excluding "junk" indices (irrelevant images or images marked for exclusion). It then assesses the retrieval list accordingly.

Significance and Robustness:

This testing and evaluation framework rigorously assesses the DualResNet's ability to generalize to unseen test data, particularly its capacity to learn shared embeddings that bridge satellite and drone imagery domains. By treating the problem as cross-modality retrieval rather than simple classification, the evaluation aligns with practical use cases where matching images across different sensors and viewpoints is critical.

Testing results

The evaluation metrics reported were:

Recall@1: 2.00%

Recall@5: 7.00%

Mean Average Precision (mAP): 2.68%

While these metrics indicate that the current model has limited retrieval accuracy in this cross-modality setting, this outcome provides valuable insights into the complexity of the task and highlights areas for potential future improvements, such as enhanced feature learning, more robust domain alignment, or additional data augmentation.