Final Report: Human-machine Collaborated Economic Development Measurement via Satellite Imagery

[1]Shukai Gong (龚舒凯) [1]Churui Zheng (郑楚睿) [1]Hang Xie (谢航)

[1]Applied Economics & Data Science, School of Applied Economics

Renmin University of China

December 27, 2024

Outline

Background and Motivation

- Reliable measures of economic activity are hard to collect in developing countries, limiting economic research as well as policy analysis.
- Can we predict fine-grained economic development level using **only satellite imagery** and a few human annotation?
- **Our Task:** Evaluate the development level of a region in **China** using this method.

We have reproduced the method in ***A human-machine collaborative approach measures economic development using satellite imagery*** published by *Nature Communications*.

Our Achievement

- Constructed a light-weighted, semi-automatic and **highly-parallelized** pipeline that realizes the annotation of the satellite data and the training of the model.
- Successfully applied the scoring model (from North Korea, Laos, ...) to a **larger and more diverse** terrain.
- Achieved a more **fine-grained** development level evaluation metric than existing metrics. (Population density map, VIIRS Map)
- **Redesigned** the training method (learning task, loss function, backbone encoder) to fit for our data.
- Exhibited good **generalization performance** on unseen satellite data (e.g. coastal regions).

Review of Stage 1

Our place of interest: *Sichuan Province, China*

- **Diverse Terrain:** Basins, Mountains and Plateaus. Unbalanced population density and economic development level.
- **Data Inaccessibility:** Statistics for remote or inaccessible areas are few and old.
- **Scaling Law:** Measuring the economic level of a large area with a small amount of high-quality annotated data.

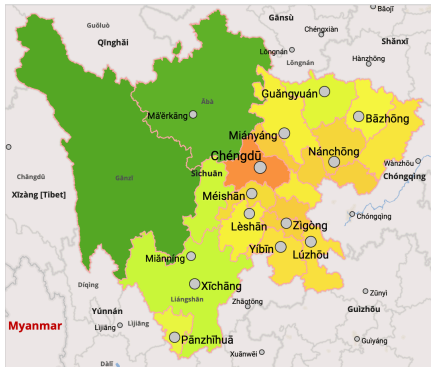


Figure 1: Population distribution of Sichuan Province

Review of Stage 1: Dataset

Data Collection & Processing: Sentinel-2 L2A high-resolution satellite images of Sichuan

- **Web-Crawling:** through **Planetary Computer API** powered by Microsoft.
- Crawled GeoTiff images of **any granule** with **any cloud cover**, during **any time period** by Python multiprocessing/threading/serial downloading.

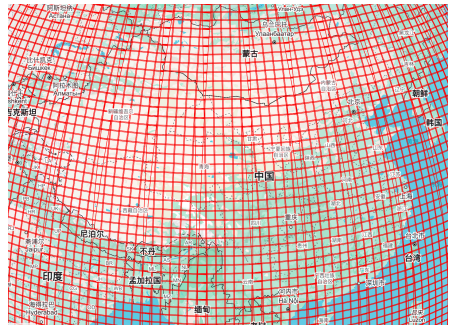


Figure 2: Illustration of Sentinel-2 granules

Review of Stage 1: Dataset

- Our dataset: $60 \times 1024 = 61440$ satellite images covering 60 granules of Sichuan Province.
  - For each granule in Sichuan, we crawl the best image (lowest cloud cover, lowest blank ratio, from 2020-2024) and segment it into 32×32 patches.
  - 900 images have annotations like $[p_{\text{Urban}}, p_{\text{Rural}}, p_{\text{Mountain}}, p_{\text{Highland}}]$.
- Test of web-crawling performance: Tested on granule 51RUQ, 2020-2023, cloud cover $<1\%$

Method	Mean	Std
Python Threading	22.11	4.84
Python Multiprocessing	27.21	3.51
Python Serial	39.78	5.94

- Test of image segmentation performance: Tested on 41 images

Method	Time usage
Python Threading	9.30
MPI (np $= 4$)	6.26

Review of Stage 1

We first train a classifier using 900 annotated images to classify images into four categories (Urban, Rural, Mountain, Highland), and then do within-category clustering:

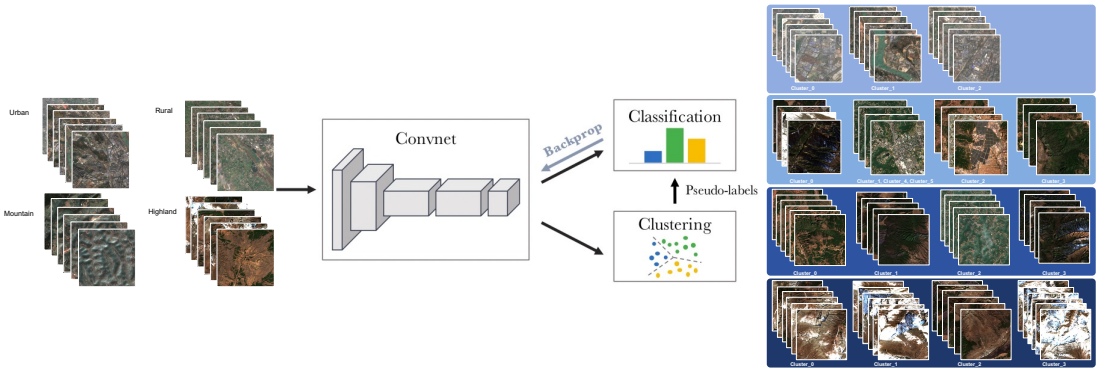


Figure 3: Category classification and within-category clustering

Review of Stage 1

Within-category Clustering: $K = 3$ for urban, $K = 11$ for rural, $K = 8$ for mountain, and set $K = 1$ for highland.

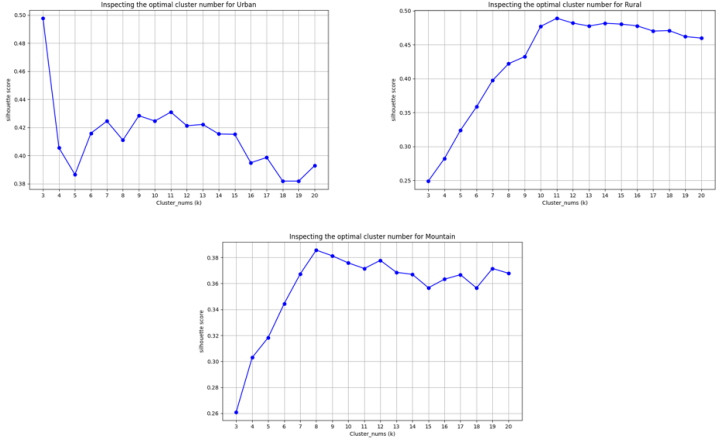


Figure 4: Silhoutte score of each category for $K \in [3, 20]$.

Review of Stage 1

What we have done **before midterm**:

- **Stage 1:** i) Obtain satellite imagery of Sichuan ii) Image Classification iii) Within-category clustering

What we have done **now**:

- **Stage 2:** POG annotation and ensemble.
- **Stage 3:** Training a development-level scoring model based on our aggregated POG.

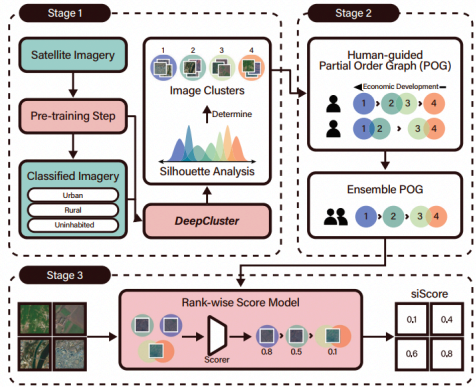


Figure 5: The proposed 3-Stage model

Stage 2: Generation of POG

We have obtained 23 clusters from Stage 1 ($|C_U| = 3, |C_R| = 11, |C_M| = 8, |C_H| = 1$). We generate POG in a **LLM-assisted** way. At each round of POG generation, we

1. Randomly sample 3 images from each cluster, i.e. 23×3 images in total.
2. Design a text prompt to demand VLLM to score those images on a scale from 0-100.
3. **Asynchronously and concurrently** send the images and prompts to VLLM via API. For each image, we require VLLM to generate 3 responses.
4. Average the API responses to get the mean score for each image.
5. Average the scores of images from each cluster to get the mean score for each cluster.
6. Rank the clusters.

$$\text{Cluster } A = \text{Cluster } B \qquad \text{if } |\text{Score}_A - \text{Score}_B| \leq 10$$
$$\text{Cluster } A > \text{Cluster } B \qquad \text{if } \text{Score}_A - \text{Score}_B > 10$$

Stage 2: Generation of POG

We run **10** rounds of POG generation, using 4 different prompts:

- "你是一位卫星遥感图像专家，以下是四川省不同地区的卫星图像，我要求你根据每张图片所代表地区的经济发展水平，对这张图片进行打分..."
- "你是一个四川本地人，以下是你的家乡不同地区的卫星图像，我要求你根据每张图片所代表地区的经济发展水平，对这张图片进行打分..."
- "你是一位发展经济学教授，以下是四川省不同地区的卫星图像，我要求你根据你对四川省经济开发现状和对发展经济学的理解，根据每张图片所代表地区的经济发展水平，对这张图片进行打分..."
- "You are an Economics student from the U.S., I want you to score the development level of the following satellite images based on your common sense..."

VLLM: Qwen2-VL-72B-Instruct, deepseek-vl2

Stage 2: Generation of POG

We have discovered that running 3-5 rounds at a time will be more efficient given the **rate limit** of most APIs.

runs	Total Time	Average Time per Run
1	9.98	9.98
3	27.84	**9.28**
5	57.24	11.45
10	353.81	35.38

Table 1: POG generation efficiency under different settings

Stage 2: Aggregation of POG

We follow the steps in the original paper with minor modifications to aggregate POGs

1. Convert discrete rankings $\mathcal{R}_i = (\text{rank}(c_1), \cdots, \text{rank}(c_{|\mathcal{C}|})$ into continous ranking vector $\mathbf{R}_i = (r_1, \cdots, r_{|\mathcal{C}|})$ via

$$r_i = \text{rank}(c_i) + \frac{1}{2} \sum_{c \in \mathcal{C} \setminus \{c_i\}} \mathbf{1}(\text{if rank}(c) = \text{rank}(c_i),\ c_i \in \mathcal{C} \tag{1}$$

   where $\mathcal{C}$ is the set of all clusters. E.g. the ranking of $(1, 2, 3, 3)$ is converted into $(1, 2, 3.5, 3.5)$

2. Initialize the aggregated ranking vector as $\mathbf{R}^* = \frac{1}{10} \sum_{i=1}^{10} \mathbf{R}_i$, and repeat the following steps until convergence ($\epsilon < 10^{-6}$):

   2.1 $\alpha_m = 1 - \exp\left(-\frac{\|\mathbf{R}_m - \mathbf{R}*\|^2}{\sigma^2}\right),\ m = 1, 2, \cdots, M\ ,\ \ w_m = \frac{\alpha_m}{\sum_{i=1}^{M} \alpha_i}\ ,$

   $\epsilon = \|\mathbf{R}^* - \sum_{i=1}^{m} w_m \mathbf{R}_m\|_2$

3. Conduct 1D K-means clustering on $\mathbf{R}^*$ to get POG.

Stage 2: Aggregation of POG

We accelerate the aggregation of POG **by matrix multiplication**, which compute $\alpha_m, w_m, \|\mathbf{R}^* - \sum_{i=1}^{m} w_m \mathbf{R}_m\|$ in parallel.

# of POGs	# of Total Clusters	Cluster Size	Matrix Ensemble	Loop Ensemble
10	22	7	0.038	**0.005**
10000	22	7	**0.048**	0.384
100000	22	7	**0.146**	3.784
# of POGs	# of Total Clusters	Cluster Size	Matrix Ensemble	Loop Ensemble
10	22	7	0.038	**0.005**
10	1000	7	0.084	**0.069**
10	10000	7	**0.184**	1.914
# of POGs	# of Total Clusters	Cluster Size	Matrix Ensemble	Loop Ensemble
10	10000	7	**0.084**	0.190
10	10000	1000	**0.188**	0.324
10	10000	5000	**0.348**	0.598

Table 2: POG aggregation efficiency under different settings

Stage 2: Aggregation of POG

Illustration of POG aggregation

$$\mathbf{R}^* = (3.8, 3.35, 3.25, 2.45, 3.05, 5.1, 2.75, 3.75, 5.25, 4.75, 5.75,$$
$$10.7, 10.2, 10.1, 7.6, 8.5, 7.7, 9.65, 10.15, 10.8, 8.7, 9.45)$$

Aggregated POG:

$$3 = 4 = 6 > 0 = 1 = 2 = 7 > 5 = 8 = 9 = 10 > 14 = 16$$
$$> 15 = 20 = 21 > 11 = 12 = 13 = 17 = 18 = 19$$

Stage 2: Aggregation of POG

Our final aggregated POG for training is

$$\underbrace{0 > 1 > 2}_{\text{Urban}} > \underbrace{4 = 3 = 6 > 9 = 12 > 17 = 11 > 13 = 10}_{\text{Rural}}$$
$$\underbrace{> 7 = 20 = 19 = 21 = 5 = 14 = 15 = 16 = 18 = 8 = 22}_{\text{Mountain and Highland}}$$

Stage 3: Training development-level scoring model

We simplify the training methods in the original paper:

- Assign pseudo labels based on the aggregated POG before.
- **Training:** Optimizing through **Multi-task Learning**.

$$\mathcal{L} = \mathcal{L}_{\text{reg}} + \lambda \mathcal{L}_{\text{linear}}$$

  where $\mathcal{L}_{\text{reg}}$ refers to the regression loss between model outputs and pseudo labels, and $\mathcal{L}_{\text{linear}}$ compares the model's prediction for a linear combination of two images with a weighted average of the model's predictions for each individual image.
- **Evaluation metrics:** If $|\text{output} - \text{pseudo label}| < \epsilon$, the output is considered "correct". (We set $\epsilon = 0.07$)

Stage 3: Training a development-level scoring model

The workflow of $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}_{\text{linear}}$ is shown in Figure 6.

- Backbone Encoder: ResNet18
- Loss function for $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}_{\text{linear}}$: L1-Loss
- Trained on 2 NVIDIA GeForce RTX 3090 with **nn.DataParallel** for 4 hours.

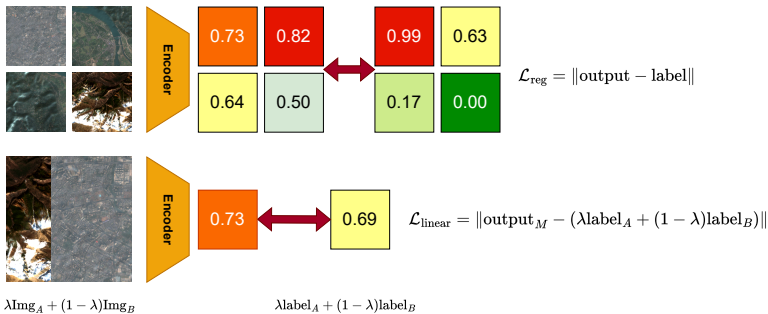


Figure 6: The workflow of scoring model

Stage 3: Training a development-level scoring model

As shown in Figure 10, we need to put together all 32×32 small chunks (343×343) to recover the original satellite image (10976×10976)

Figure 7: Putting together all 1024 chunks to recover the original satellite image

Stage 3: Training a development-level scoring model

This task is suitable for **parallel computing with GPUs**. We use **SourceModule in PyCuda to write GPU kernel functions.** The design is as follows:

- **Thread:** Each thread is responsible for filling the scores into their corresponding 343×343 matrices.
- **Thread block:** Each thread block is responsible for concatenating a large image, containing 32×32 small blocks.
- **Grid:** Each grid contains n thread blocks, which are responsible for putting together n 10976×10976 large images.

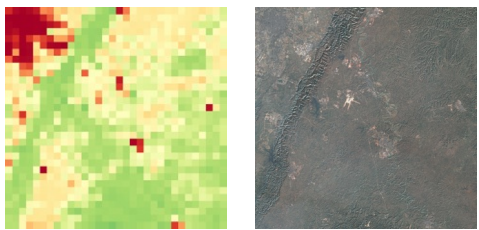


Figure 8: The development-level map (Left) and satellite image (Right) of 48RVU(**成都**)

Stage 3: Training a development-level scoring model

	# of images	Total Time	Averaged Image I/O time	Average Concatenation Time
Serial	1	39.60	2.27	37.33
	6	235.24	2.13	37.08
	15	591.71	2.22	37.23
GPU Parallel	1	2.58	2.23	0.35
	6	13.97	2.13	0.2
	15	34.27	2.10	0.18

Table 3: Comparison between serial and parallel image concatenation

Stage 3: Training a development-level scoring model

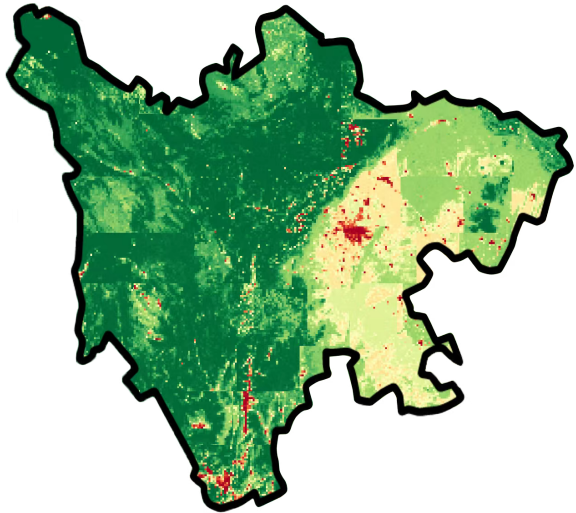


Figure 9: The development-level map of the whole Sichuan province

Stage 3: Training a development-level scoring model

Generalizability: we also deploy our model on 51RUQ(上海), 50RQP(福州), and 50QMM(潮汕). The results indicate that our model exhibits **good generalization performance** even on unseen terrain (such as waters).

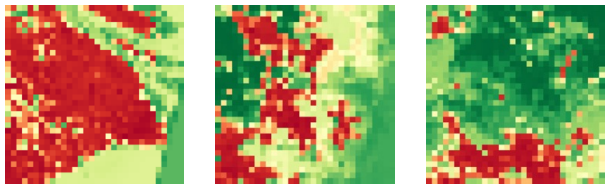
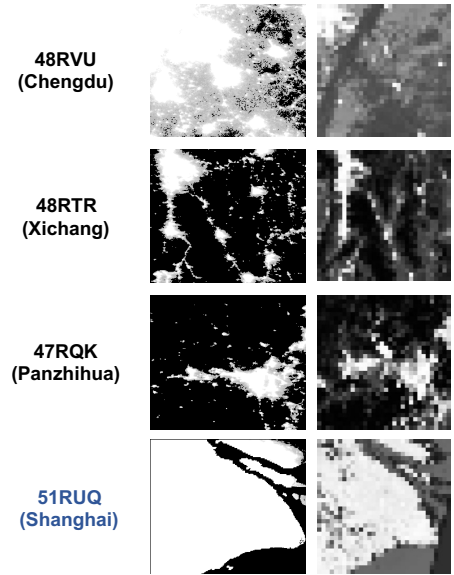


Figure 10: The development-level map of Shanghai(Left), Fuzhou(Middle) and Chaoshan(Right)

Stage 3: Training a development-level scoring model

Feasibility Evaluation: We compute the correlation between our development-level scoring model and the VIIRS night light map (2023).

Granule	Region	Pearson Correlation	Spearman Correlation
48RVU	成都	0.18	0.16
48RTR	西昌	0.29	0.21
47RQK	攀枝花	0.42	0.35
51RUQ	上海	0.56	0.52
50RQP	福州	0.27	0.26
50QMM	潮汕	0.45	0.61



Summary

Room for improvement:

- Hallucination of the trained model still exists. (Between urban & highland)
- Error in initial classification and annotation (Stage 1).
- More complete feasibility evaluation.

Thank you!