Learning Geographic Embeddings from Satellite Imagery (Computer Vision)

David Chen
Stanford Center for Professional Development
Stanford University, Stanford, CA, 94305
dchen11@stanford.edu

Neha Konakalla
Department of Computer Science
Stanford University, Stanford, CA, 94305
nkona@stanford.edu

Avrum Noor
Department of Electrical Engineering
Department of Computer Science
Stanford University, Stanford, CA, 94305
avrum@stanford.edu

1 Introduction

Across a variety of fields, there is always a need for data about any given location. For instance, to track population and demographic shifts, there is a need to have accurate population and demographic data by time and location. Nowadays, this is often collected through census data. To track deforestation and climate impact on forests, there is a need for forest cover data. Survey based methods on the ground are often too expensive to perform frequently, and some countries may be too poor or may not have practical means to do them at all.

Satellite imagery can be a potential solution. Recently, there have been many gains in using machine learning to make quantitative estimates about many tasks from satellite images (SIML). However, going through the whole training process for every single different task can often be prohibitive in terms of the amount of compute needed. An alternative is to first pre-compute a general set of embeddings, or “feature vectors", for a set of images. Then, we would use these embeddings as the input for all the downstream models for each application.

For this project, we wanted to explore the viability of using a multi-task learning setup to train a ResNet18 on a set of tasks (nightlights, average income, average road length, and average housing prices, etc.). The input of this setup would be our satellite images, and the output would be the scalar predictions for our set of tasks. Then using this model, we would run a set of images through the forward pass, stopping before the last layer, and grabbing that set of values as our general embeddings to test on other tasks. (Note that our results for the multi-task setup were not very impressive, so we also explored some other alternatives, like pre-training models on individual tasks, and concatenation of features from said models). For these “downstream” tasks, the input would be the generated feature vectors, and the output would be the predictions for a given task. The model used at this step can be anything that fits. We base all comparisons on ridge regression, as it is simple and fast and since we want our generated feature vectors to be easy to use.

In addition, we wanted to test the viability and generalizability of this approach to a new downstream task not yet seen before in the context of satellite imagery: prediction of self-storage facility pricing. Prediction for this application would help in determining optimal locations for new facilities, and fair pricing for existing facilities. Our results show that adding features produced by MOSAIKS [1] to specific unit tabular features boosts the prediction capability using ridge regression.

2 Related work

MOSAIKS is a state-of-the-art technique that produces general task-agnostic feature vectors from satellite imagery for use in downstream tasks [1]. MOSAIKS was able to produce results that were actually competitive with traditional end-to-end satellite imagery to individual task setups. Much of our project was attempting to reach the level of MOSAIKS on the same set of downstream tasks.

For multi-task learning on satellite images, we consider two existing papers. One is regarding learning geographic elevation and semantics at the same time. Semantics can be described as providing a class classification at the pixel level [3]. The other paper is regarding multi-task learning on pansharpening and semantics, where pansharpening is the process of combining a high-res but low channel image with a low-res but colored image to create a high-res, colored image [4]. Note that both [3] and [4] employ the use of an encoder-decoder network architecture, while our novel approach uses a ResNet18.

For multi-task learning, there has also been work relating to the proper weighting of the tasks. In [6], uncertainty weighting is explored, while in [5], a naive random loss weighting is explored. The second paper is considerably less complex and requires less computation.

Finally, tile2vec from Stanford provides a solution for low dimensional embeddings of remote imagery using unsupervised representation learning [2]. The underlying architecture is also a ResNet, but they benchmark on classification tasks. Nevertheless in their results, they are able to beat out other unsupervised methods and even supervised trained models. There has yet to be a paper on multi-task learning to generate embeddings for satellite imagery, so we decided to take this approach over the self-supervised approach already explored by tile2vec.

3 Dataset and Features

For the satellite images, we sourced 1 km by 1 km daytime images from the Google Static Maps API at a resolution of 640px by 640px. There were two sampling methods to determine what images to collect. One was selecting locations uniformly at random over the continental United States, and the other was sampling proportional to population over the continental United States. In total, there were 200,000 images downloaded for the embedding generation work. We employed a 64%-16%-20% train-dev-test split. Before use, the images were downsized to 224px by 224 px. During training for the multi-task learning, data augmentation in the form of random cropping, flipping, and random rotation was used.

For the labels for all downstream tasks (forest cover, elevation, population density, nighttime lights, income, road length, and housing prices), see the appendix.

The self-storage dataset was scraped data off of various self-storage websites. The data consists of 33,085 units from 2,271 facilities, with the final label being the price for the unit, and various tabular features including square footage, presence of air conditioning, power outlets, stair access, etc. 22,101 of those units were used as the train dataset, 5,555 were used as an “in-distribution" test set, and 5,429 were used as an “out-of-distribution" test set, containing units from self-storage facilities not present in the training set.

Thus, an extra 2,296 images were required to be downloaded from Google Maps Static API. An example of an images can be seen in the appendix.

4 Methods

4.1 Baselines + Comparisons

The two baselines we used for embedding generation and evaluation were the MOSAIKS random patch method, as well as extracting the last layer of a ResNet152 pretrained on natural imagery (not satellite imagery). The ResNet152 will give us a 2048-dimensional vector for every image. The pretrained model weights were pulled from PyTorch’s model zoo. The MOSAIKS method consists of randomly selecting 8192 3x3x3 patches from the image dataset. Next, for a given patch, it is convolved across all images. The results are then passed through a ReLU, and then averaged down to a single scalar value for each image. This process is repeated for all 8192 patches for each image, resulting in a 8192-dimensional feature vector corresponding to a satellite image.

Another comparison was with separate end-to-end ResNet18s each trained on a given application, with MSE loss and no additional regularization. All were trained with stochastic gradient descent.

4.2 Multitask model

The base of the multitask model was a ResNet18. (A ResNet is a convolutional neural network with layers augmented to learn residual functions to help in the learning process). The last layer of the ResNet18 was replaced with a linear layer taking in the 512-dimensional flattened output of the previous layer, and outputting to a T-dimensional vector, where T is the number of tasks.

There was an alternative head architecture explored, where at the end, we appended two linear layers with a ReLU activation after the first. Dropout was also applied to the first linear layer. The number of hidden units in the second layer, as well as the dropout percentage, were hyperparameters that were evaluated.

In either case, the loss function applied to the output was MSE loss, along with L1 regularization applied to the weights of the final linear layer. In addition, the MSE loss for each task was weighted differently according to two explored methods: random weighting, and uncertainty weighting.

$$loss = \frac{1}{m}\sum_{i=1}^{m}(\sum_{j=1}^{T} w_j(y^{(i)} - \hat{y}^{(i)})^2 + v_j) + \lambda \sum_{k} |\theta_k|$$

For random weighting, The w_j are sampled from $\mathcal{N}(0,1)$, and then passed through a softmax to ensure positive values. Also, v_j is 0. For uncertainty weighting, $w_j = e^{-v_j}$, where v_j is now a learnable parameter.

This training involved stochastic gradient descent, which applies gradients after every new training example instead of in batches. More information about training with the two differently sampled image datasets is explained in the appendix.

To extract features for each image, we sent the images through a forward pass and utilized the 512-dimensional output before the final linear layers as the feature vector.

4.3 Feature concatenation

As revealed in the results section, the downstream performance of the multitask generated features was lackluster, so an alternative direction was also explored. The images were passed through the individual task trained ResNet18s, and 512 features were extracted per task for every image. In downstream evaluation, all the features from all tasks except for the 512 from the task currently being evaluated were used, leading to a final feature vector of dimension 3072.

4.4 Downstream tasks

For all downstream tasks, we employ a 64%, 16%, 20%, train, dev, and test split. The model is simple ridge regression, with the L2 penalization parameter tuned by 5-fold cross validation on the 80% combined validation set. k-fold cross validation is a technique where we split the validation set into k groups, and train k separate models where for each model, a different group is selected as the “test" set. The final evaluation of the downstream tasks is with the R^2 value, since we are evaluation regression tasks.

4.5 Self-Storage

The model used for the self-storage downstream task was also ridge regression. However, since the data is not price for a location, but rather price for a variety of units within a location, we must also use the unit specific features. Again, optimal L2 penalization was found using 5-fold cross validation.

There was also an exploration of a 3-layer model, but during training it consistently performed worse.

5 Experiments/Results/Discussion

5.1 Multitask and feature generation

The hyperparameters we tuned were the L1 regularization factor, as well as the milestone at which we shrank our learning rate from 0.001 to 0.0001. The tuning process involved training for a shortened number of epochs (around 10) and selecting the hyperparameters with the best R^2 results on the tasks. This was also the process by which we tested the different architecture choices. The reason for this rather than a more rigorous thorough exploration of hyperparameters was the time needed for training. A full 50 epochs takes at least 15 hours to run on just one dataset of 80k images. The best performing multi-task model uses a L1 penalization parameter of 0.1, and a learning rate milestone at 10 epochs.

Our primary evaluation metric was simply the R^2 value. (Equations in appendix).

In the following table, “Multitask", “Combined", “ResNet152", and “MOSAIKS" represent the R^2 on the test sets for all the downstream tasks, with the ridge regression model taking as input generated embeddings from said model/technique. “Multitask end-to-end" and “ResNet18 end-to-end" represent the R^2 on the test set for each task, with no concept of embeddings or additional regression, but directly from image to prediction. The new models we were responsible for building are bolded.

Tasks	**Multitask**	**Combined**	ResNet152	MOSAIKS	**Multitask e2e**	ResNet18 e2e
Forest cover	0.62	0.63	0.66	0.91	0.88	0.94
Elevation	0.26	**0.33**	0.32	0.68	0.66	0.80
Population density	0.26	**0.33**	0.29	0.72	0.68	0.80
Nighttime lights	0.43	**0.55**	0.47	0.85	0.77	0.89
Income	0.01	0.07	0.07	0.45	0.40	0.47
Road Length	0.14	**0.18**	0.16	0.53	0.44	0.57
Housing Price	-0.01	**0.07**	0.01	0.52	0.46	0.50

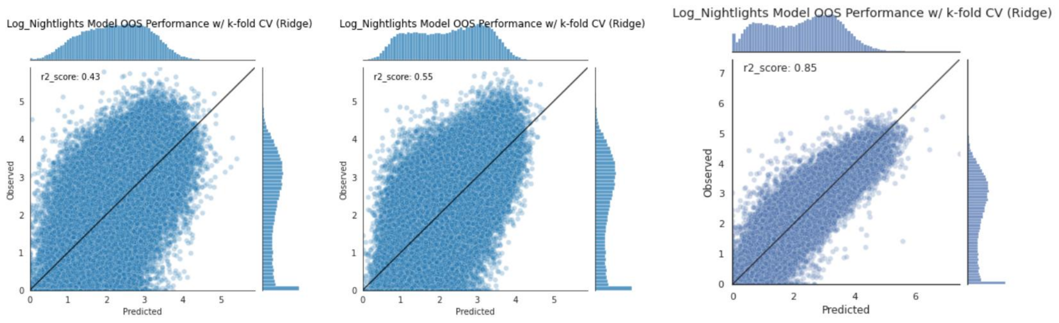

Figure 1: KDE Scatterplots on nighttime lights task. From left to right: Multitask features, Combined ResNet18 features, and MOSAIKS features

5.2 Analysis of multitask and feature generation performance

Ridge regression on features produced from the multitask setup did not perform as well as we had hoped. Some R^2 on the tasks came close to the ResNet152 baseline, but none exceeded the baseline. The regression on the combined features generated from the ResNet18s performed better, to the point where R^2 on all tasks except for forest cover matched or exceeded the ResNet152 baseline. Yet, they still did not reach the R^2 attained by MOSAIKS features. In the worst case, the R^2 differed by 0.45 points for the housing price application.

However, we can see from Figure 1. that the combined features are able to provide enough information such that the shape of the scatterplot starts to match that of the one generated by MOSAIKS. For example, it reveals the bimodel distribution of the data also shown in MOSAIKS. This suggests to me that there is enough information captured in the feature vector, but it may be too noisy.

A possible explanation for the performance discrepancies is the bandwidth of information available given the dimensions of the feature vectors. Since our multitask approach was based on a ResNet18's last hidden layer, we were constricted down to 512 dimensions. In the combined feature vectors, even though the individual ResNet18s were not trained on multiple tasks, it could embed more and a wider variety of information with 3072 dimensions. Similar logic applies for the ResNet152's 2048, and MOSAIKSs' 8192. In fact, more features continues to aid performance on downstream tasks at the cost of computation [1].

On the other hand, after training, our end-to-end multitask models performed competitively with MOSAIKS, only losing out by 0.09 R^2 points at the most. Even with our limited amount of model iteration, it performed better than expected, considering many of the tasks were shown to not be very correlated [1]. Although it did not beat the MOSAIKS or individual ResNet18s baselines, with further architecture exploration and hyperparameter search, it is very possible that this performance could increase higher still.

In terms of overfitting, L2 reg was applied for all downstream tasks, and for multitask learning, data augmentation and L1 reg was explicitly implemented to prevent overfitting. In fact, with the addition of data augmentation, the R^2 on validation data always hovered slightly above the R^2 on the training data, so there was no blatent overfitting present.

5.3 Self-storage

Distribution	**Multitask**	**Combined**	ResNet152	MOSAIKS	Only unit features
In	0.5663	0.6001	0.5092	**0.6245**	0.6121
Out	0.5546	**0.5943**	0.5014	**0.6260**	0.5926

The best value of the L2 penalization parameter found using 5-fold cross validation was 0.01. Note that without any embeddings added, using just the unit features, the R^2 was found to be 0.6121 on the in-distribution test set, and 0.5926 on the out-of-distribution test set. Any improvement in this value shows that the image embeddings provide enough value about the location, while any decrease indicates some level of overfitting. Indeed, when I increase the L2 penalization parameter manually on the other three types of embeddings, it does reach a point eventually where the R^2 of "only unit features" will decrease to the point of being much closer, and in some cases, below the R^2 with the embeddings.

What we find is that MOSAIKS does add valuable information about the surrounding location into the price prediction task. This makes sense, as things like population density and income should definitely have some amount of correlation with self-storage unit prices, and if the embeddings are able to contain enough information to perform well on those tasks, it makes sense that it would add value in this case. However, much like the other downstream tasks, we find that the other embedding types do not add as much value, and in most cases, serve only to increase overfitting.

6 Conclusion/Future Work

The end-to-end multitask model itself performed well, but not better than the alternatives. With future iteration, it may very well match or potentially outperform the baselines. The embeddings generated from this multitask model were not as impressive, not even meeting the features from the baseline ResNet152. However, an alternate approach of concatenating features from individually trained ResNet18s beat the ResNet152 baseline. None reached the performance of MOSAIKS, however. In the future, exploration of a different architecture could be valuable, especially one that allows for higher dimensional embeddings. Or, a model such as encoder-decoder or a vision transformer (ViT) could produce higher quality embeddings, especially in conjunction with training techniques such as self-supervised learning as seen in tile2vec.

The self-storage unit price prediction results demonstrate that it is possible for satellite image embeddings to provide useful information in the prediction task. With more time and resources, exploration of using only embeddings to predict prices would be insightful. Another possible direction is to build a model where the input is satellite imagery or a lower dimensional embedding, and the output is a vector of coefficients that we can use to take the dot product with the other unit features for final unit price prediction. This would be greatly beneficial in terms of being interpretable, and being able to see how much a unit feature like air conditioning contributes to final unit pricing.

7 Contributions

David worked on data collection, image downloading, and pre-processing for all downstream tasks except for self-storage. He also worked on replication and running of all baselines and alternative models. David also worked on the implementation and iteration of the multitask model, as well as the combined concatenated features. David was also responsible for the results for all downstream tasks, including the self-storage application. David was also the main author of the report.

Neha was responsible for self-storage data preprocessing, image downloading, and iteration on the 3-layer model for the self-storage task. She also helped edit the report, and contributed to the video.

Avrum was responsible for self-storage data preprocessing, image downloading, setting up AWS architecture and git repository, finding the optimal training epochs for the models, as well as setting up dataloaders and running various MLP models (3-layer and 5-layer MLP networks) for making predictions from feature embeddings produced from MOSAIKS. He also co-authored the report and contributed to the video.

Scott Lanyon Fleming was responsible for providing mentorship, additional datasets, including all self-storage related data, and baseline code examples.

8 Appendix

8.1 Getting labels for all downstream tasks

The labels for all downstream tasks (forest cover, elevation, population density, nighttime lights, income, road length, and housing prices) were pulled from various online sources, and preprocessed individually with help from provided MOSAIKS scripts. The labels are then paired with images based on latitude and longitude. For forest cover, the raw data is available from The Global Land Analysis and Discovery Group at the University of Maryland. For elevation, the raw data is available from the Mapzen elevation tiles on AWS. For population density, it is sourced from the GPW dataset. The nighttime lights raw data can be retrieved from The Earth Observation Group at the Colorado School of Mines. Road length is available from the USGS National Transportation Dataset. Housing prices are originally from Zillow, but since the API is discontinued, we skip directly to the intermediate label dataset downloaded from the orignal MOSAIKS codebase. Income data is downloaded from the US Census Bureau's website.

8.2 Satellite image example

This example corresponds to the latitude longitude pair of (48.97, -122.51). Note the Google watermark, which is illegal to remove. The images are processed within Google to be during the daytime, and to not have clouds.

Google
Imagery ©2022 CNES / Airbus, Maxar Technologies, U.S. Geological Survey, USDA/FPAC/GEO

8.3 Training on UAR and POP image datasets

Training the model involved training on images sampled based on population, or uniformly random across the US, and the relevant tasks for those. For example, a model was trained for the “UAR" uniform images, with forest cover, elevation, and population density as the relevant tasks. There was an attempt to train on both datasets simultaneously, but various performance factors led us to abandon that approach. See the next section in the appendix for more info.

8.4 R^2 equation

R^2, also known as the coefficient of determination, is often used to judge performance on regression tasks.

The equation is as follows:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(\hat{y}^{[i]} - y^{[i]})^2}{\sum_{i=1}^{m}(\hat{y}^{[i]} - \bar{y})^2}$$

8.5 Using MOSAIKS Embeddings for MLP

The exploration of various MLP models (3-layer and 5-layer MLP networks) for making predictions from feature embeddings produced from MOSAIKS can be found here: `https://github.com/avrumnoor/cs230_final_project/tree/main/MLP`.

References

[1] Rolf, E. & Proctor, J. & Carleton, *T. et al.* (2021) A generalizable and accessible approach to machine learning with global satellite imagery. *Nat Commun 12*, 4392. https://doi.org/10.1038/s41467-021-24638-z

[2] Jean, N., Wang, S., Samar, A., Azzari, G., Lobell, D., & Ermon, S. (2019). Tile2Vec: Unsupervised Representation Learning for Spatially Distributed Data. *Proceedings of the AAAI Conference on Artificial Intelligence, 33(01)*, 3967-3974. https://doi.org/10.1609/aaai.v33i01.33013967

[3] M. Carvalho, B. Le Saux, P. Trouvé-Peloux, F. Champagnat and A. Almansa (2020). Multitask Learning of Height and Semantics From Aerial Images, in *IEEE Geoscience and Remote Sensing Letters, vol. 17, no. 8*, pp. 1391-1395, Aug. 2020, doi: 10.1109/LGRS.2019.2947783.

[4] A. Khalel, O. Tasar, G. Charpiat and Y. Tarabalka (2019). Multi-Task Deep Learning for Satellite Image Pansharpening and Segmentation, *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, 2019*, pp. 4869-4872, doi: 10.1109/IGARSS.2019.8899851.

[5] Lin, B., Ye, F., & Zhang, Y. (2021). A Closer Look at Loss Weighting in Multi-Task Learning. ArXiv, abs/2111.10603.

[6] Kendall, A., Gal, Y., & Cipolla, R. (2018). Multi-task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 7482-7491.

Code:

The MOSAIKS codebase, from: github.com/Global-Policy-Lab/mosaiks-paper

PyTorch, from: pytorch.org

Various others, including numpy, pandas, and other commonly used libraries.