# Outperforming Night-lights in Poverty Prediction by Selected Remote Sensing CNN Features
*Milestone Report*

**Zhaozhuo Xu, Zhihan Jiang and Yicheng Li**
Department of Electrical Engineering, Stanford University

## Introduction

Accurate poverty measurements of certain area critically shape the decisions of local governments about how to allocate scarce resources, and to track progress toward improving human livelihoods. However, ground truth data from conducted surveys such as the Demographic and Health Surveys (DHS) are far from adequate in such regions. Closing this data gap is prohibitively costly and institutionally difficult, due to governments' reluctance to show their lackluster economic performance. An alternative path is proposed to measure poverty by leveraging satellite images of luminosity at night (night-lights). This technique shows promising results in improving the predictions of urban area's economic status, but has trouble distinguishing differences of economic activity between areas with populations living near or below the international poverty line ($1.90 per capita per day). In these impoverished areas, luminosity levels are generally very low and show little variation. To remedy this issue, a novel machine learning(Jean et al. 2016) approach of extracting socioeconomic data from high-resolution daytime satellite imagery is proposed. In this approach, a convolutional neural network (CNN) model pre-trained on ImageNet is fine-tuned to predict nighttime light intensities based on the input daytime satellite imagery. We obtained a large labeled training dataset using CNN as a feature extractor. The features are then used to predict poverty assets via linear and ridge regression. This approach does not depend on night-lights data, which enables it to distinguish poor, densely populated areas from wealthy, sparsely populated areas.

However, this CNN feature extraction method creates a dataset with size smaller than the feature dimensions, thus overfitting would be a potential problem. Another issue is that, CNN features cannot outperform night-lights data by itself in poverty prediction. To tackle these issues, we conduct feature selections on CNN features and try various regression models to find a feature-regressor combination that outperforms night-lights data in poverty prediction. Through our work, a fine-grained poverty and wealth estimator will be produced using only the data available to the public domain. Our major contribution lies in these three parts:

- Train CNN as a feature extractor rather than a classifier.
- Reduce overfitting by wisely choosing CNN features.
- Improve prediction performance by using stronger regression models.

## Methodology

We begin by introducing the CNN model as a feature extractor, and then we will introduce the regression models used in our work. Finally, we will present our contribution in feature selections and boosting models.

### Fully Convolutional Neural Network

In our work, we use a fully convolutional model converted from the VGG F model pre-trained on ImageNet. We train this VGG model by using remote sensing images with corresponding night-lights intensities as labels. The input to our VGG model is $400 \times 400$ pixel and the 4096 fully connected layer features produced by our model are selected as our CNN features.

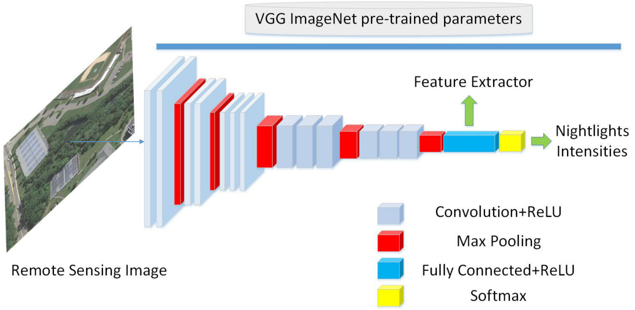


Figure 1

### Regression Model

- Linear Regression
  Linear regression a naive model for CNN feature regression, which models the error as normal distributions. We consider linear regression as our baseline for all features and regression models.
- Ridge Regression
  Ridge regression is a technique for analyzing multiple regression data that suffers from multicollinearity. When

multicollinearity occurs, least-square estimates are unbiased, but their variances are large so they may be far from the true value. By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors. It can also help mitigate overfitting to some extend.

- Lasso Regression
  Lasso (Least Absolute Shrinkage Selector Operator) regression is quite similar to ridge regression, but instead of using L2 regularization, Lasso uses L1 regularization, which causes many fitted coefficients to become zero. In other words, Lasso automatically discards features that are not very helpful, and only uses a small number of features. For this reason, Lasso is even less susceptible to over-fitting than ridge regression.

### Feature Selection Methods

The number of CNN features is well above 4000, larger than the number of samples. Therefore, we may experience serious overfitting if we use all these features directly in the regression. In the experiments, we try some possible feature-selection approaches:

- Correlation-based Feature Selection
  The basic method to select features is to compute each feature's correlation with the label (asset index/poverty level). Then we simply use the features with the highest correlation in our regression. The correlation values are computed using the training set only.

- Principle Component Analysis
  Principal components analysis is often used in unsupervised learning to find a low-dimensional coordinate system that preserves the separation of data points. Here, we try to apply PCA to the 4096 CNN features and see if the principle components would give a good regression performance. We also compare it with other feature selection methods in the experiments.

- Lasso Features
  Lasso regression automatically selects some features to use while reducing the coefficients of others to zero. This property is absent in the case of ridge regression.

## Experiments

### Dataset

In our work, we construct a dataset with the DHS survey containing wealth assets and households data from five African countries, which are Rwanda, Malawi, Uganda, Tanzania and Nigeria. We also collect the corresponding night-lights intensities value in the surveyed areas. Then we fine-tuned the proposed VGG model using the remote sensing images in the surveyed areas and its night-lights intensities to get the CNN features. To test different feature selections and regression models, we split our dataset randomly into $66.6\%$ training set and $33.3\%$ testing set. We train different feature selection methods and regression models on the training set and validate them on testing set.

### Results

First, we run straightforward linear regression using all 4096 CNN features, and compare the result with our baseline: the model trained with night-lights data only. We use R2 value to evaluate the performance of the models. Higher R2 indicates better model performance. As shown in Figure 2, when all features are used, the training R2 is 0.995, but the test R2 is below zero. This clearly is overfitting.

To address the problem, we try 3 feature selection methods. First, we simply select features starting from the beginning of the 4096 CNN features. As shown in Figure 2, using 50 to 400 features gives slightly better results than night-lights. But as the number of features gets large, overfitting becomes obvious. Second, when using PCA components as features in linear regression, the performance is rather unsatisfactory. This is because PCA loses information of the original features. Finally, if we use features with the highest correlation with the labels, the situation is similar to selecting features from the beginning. When we use more than 200 features, overfitting becomes obvious. This also tells us that the correlation is not an excellent metric in selecting features.

Since linear regression does not give satisfactory results, we run ridge regression instead. Figure 3 shows the results using ridge regression. Again, using all 4096 CNN features leads to serious overfitting, and PCA components behave poorly because PCA loses information. When using 50 to 600 features, overfitting still appeared as the number of features gets large, but not as obvious as with linear regression. In comparison with Figure 2, We see that ridge regression reduces overfitting to some extend. Using 200 features now gives some better results than night-lights, but still not satisfactory.

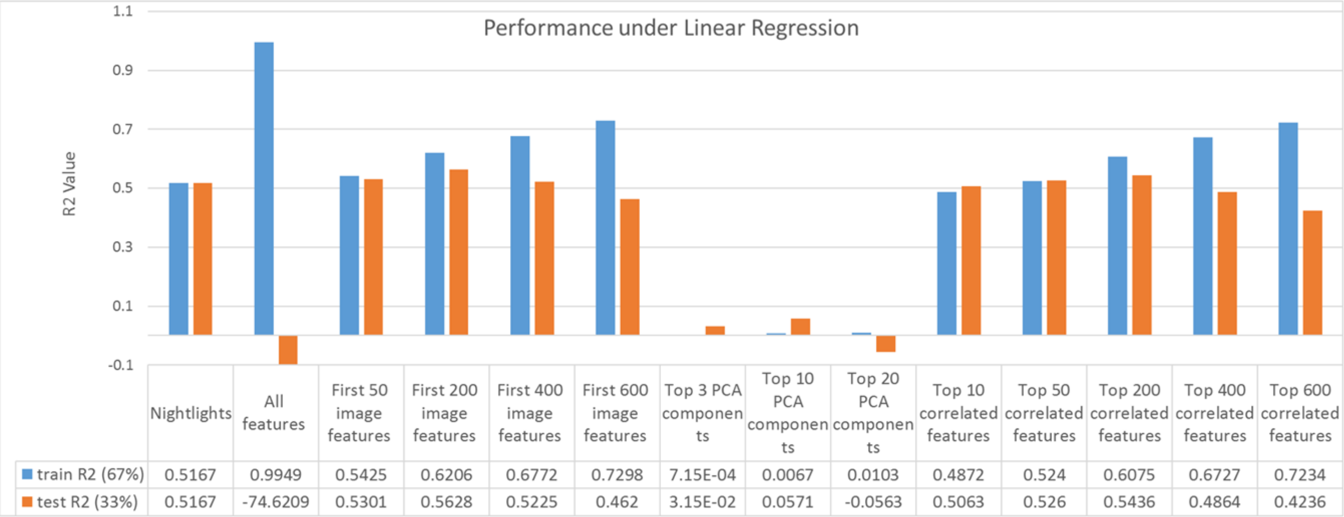
Then we turned to Lasso regression. As shown in Figure 4, feeding all 4096 features to Lasso regression yields promising results. This is because Lasso automatically discards most of the features and only uses a small portion of them. In fact, Lasso selects 282 features out of all 4096 CNN features - all other features have coefficients of zero in Lasso regression. Also, the right-hand part of Figure 4 shows that as the number of features gets large, Lasso almost exhibits no overfitting problem. Therefore, in our situation, where we have a huge number of features to select from, Lasso is a convenient and robust method to use.

## Conclusion & Next Step

For the current phase, we are able to achieve a significantly better prediction of asset index by using Lasso regression and Lasso feature selection of CNN features. In the next phase, we will try to utilize boosting regressions such as XGBoost, and more feature selection methods such as feature forward, to seek any potential better solution for this work.

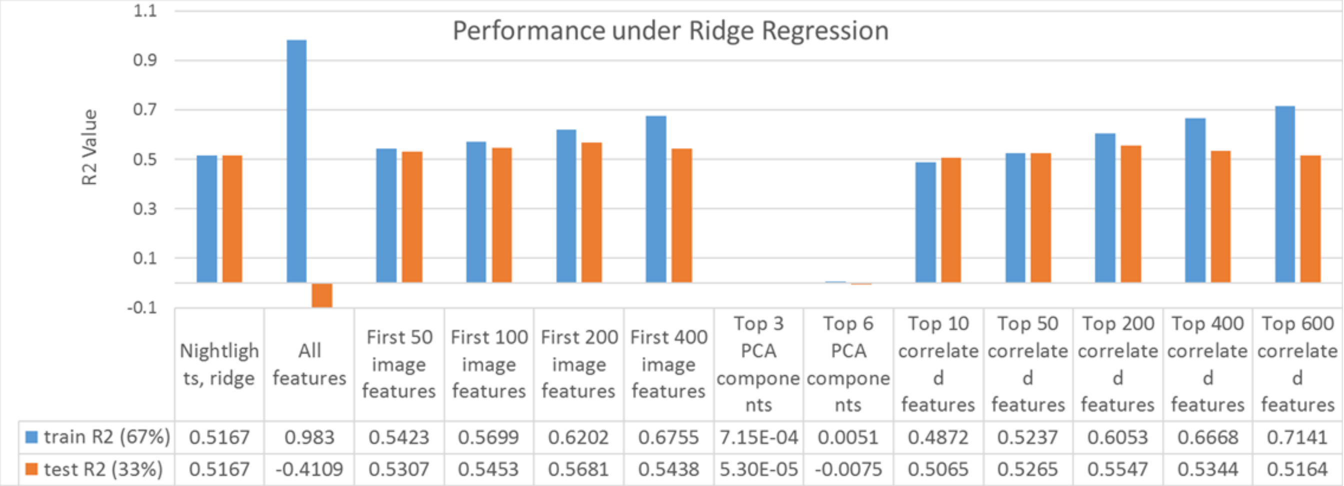
## References

[Jean et al. 2016] Jean, N.; Burke, M.; Xie, M.; Davis, W. M.; Lobell, D. B.; and Ermon, S. 2016. Combining satellite imagery and machine learning to predict poverty. *Science* 353(6301):790–794.



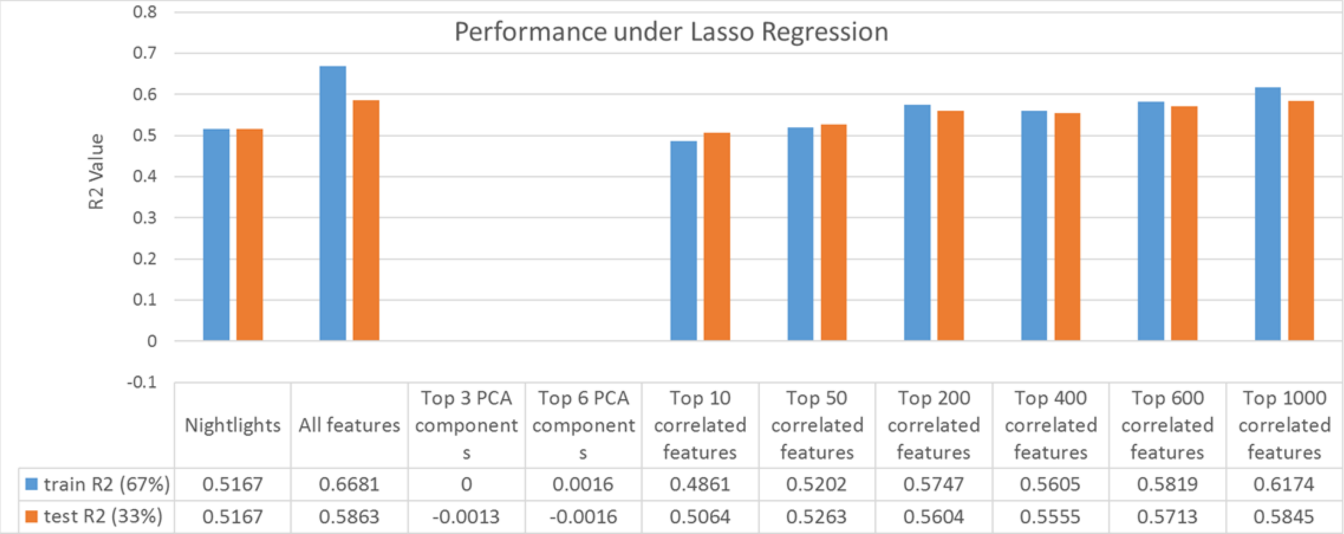
| | Nightlights | All features | First 50 image features | First 200 image features | First 400 image features | First 600 image features | Top 3 PCA components | Top 10 PCA components | Top 20 PCA components | Top 10 correlated features | Top 50 correlated features | Top 200 correlated features | Top 400 correlated features | Top 600 correlated features |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ■ train R2 (67%) | 0.5167 | 0.9949 | 0.5425 | 0.6206 | 0.6772 | 0.7298 | 7.15E-04 | 0.0067 | 0.0103 | 0.4872 | 0.524 | 0.6075 | 0.6727 | 0.7234 |
| ■ test R2 (33%) | 0.5167 | -74.6209 | 0.5301 | 0.5628 | 0.5225 | 0.462 | 3.15E-02 | 0.0571 | -0.0563 | 0.5063 | 0.526 | 0.5436 | 0.4864 | 0.4236 |

Figure 2



| | Nightlights, ridge | All features | First 50 image features | First 100 image features | First 200 image features | First 400 image features | Top 3 PCA components | Top 6 PCA components | Top 10 correlated features | Top 50 correlated features | Top 200 correlated features | Top 400 correlated features | Top 600 correlated features |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ■ train R2 (67%) | 0.5167 | 0.983 | 0.5423 | 0.5699 | 0.6202 | 0.6755 | 7.15E-04 | 0.0051 | 0.4872 | 0.5237 | 0.6053 | 0.6668 | 0.7141 |
| ■ test R2 (33%) | 0.5167 | -0.4109 | 0.5307 | 0.5453 | 0.5681 | 0.5438 | 5.30E-05 | -0.0075 | 0.5065 | 0.5265 | 0.5547 | 0.5344 | 0.5164 |

Figure 3



| | Nightlights | All features | Top 3 PCA components | Top 6 PCA components | Top 10 correlated features | Top 50 correlated features | Top 200 correlated features | Top 400 correlated features | Top 600 correlated features | Top 1000 correlated features |
|---|---|---|---|---|---|---|---|---|---|---|
| ■ train R2 (67%) | 0.5167 | 0.6681 | 0 | 0.0016 | 0.4861 | 0.5202 | 0.5747 | 0.5605 | 0.5819 | 0.6174 |
| ■ test R2 (33%) | 0.5167 | 0.5863 | -0.0013 | -0.0016 | 0.5064 | 0.5263 | 0.5604 | 0.5555 | 0.5713 | 0.5845 |

Figure 4