Poverty Prediction by Selected Remote Sensing CNN Features
Final Report

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

Abstract

Remote sensing images with the convolutional neural network (CNN) model are proved to be an alternative approach to night-lights in poverty prediction. However, CNN model creates a high dimensional dataset, which may cause overfitting, and limit its performance. In this paper, we select on remote sensing CNN features to improve its performance. We leverage five feature selection methods: forward search, correlation search, principle components analysis (PCA), variational autoencoder (VAE) and lasso. All these features are then evaluated via linear, ridge and lasso regression method as well as XGBoost gradient boosting methods. We obtain robust feature sets that outperform night-lights intensities in poverty prediction.

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 approach(Jean et al. 2016) 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 levels 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 it has no closed form solutions. Moreover, overfitting would be a considerable problem. Another issue is that, CNN features cannot outperform night-lights data by itself in poverty prediction (Xie et al. 2016). 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 can 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 selecting 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 and boosting models used in our work. Finally, we will present our contribution in feature selections.

Fully Convolutional Neural Network

Convolutional Neural Networks (CNN) are powerful models with application widely spread in image processing tasks such as classification, semantic segmentation and object detection. The multiple convolution layers provide massive feature assets while the pooling layers select on that. CNN model such as VGG net (Simonyan and Zisserman 2014) facilitate further research on the use of deep visual representations in computer vision.

In our work, we use a convolutional model converted from the VGG F model pre-trained on ImageNet as **Figure 1**. We train this VGG model by using remote sensing images with corresponding night-lights intensities as labels. The input to our VGG model is 400×400 pixels and the 4096 fully connected layer features produced by our model are selected as our CNN features. We use TFlearn framework built on Tensorflow to implement VGG model.

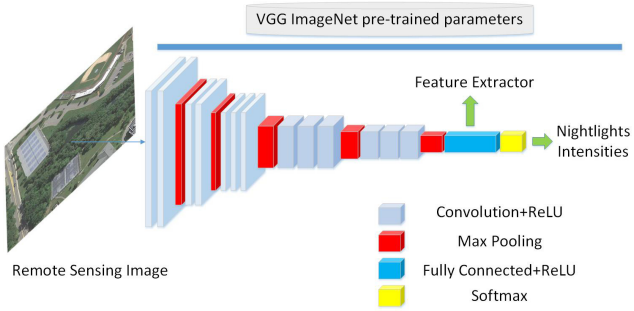


Figure 1

Regression Model

- Linear Regression
  Linear regression is 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.

$$minimize \qquad (X\theta - y)^T (X\theta - y)$$
$$subject\ to \qquad \sum_{i=1}^{n} \theta_i \leq t$$

- 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.

$$minimize \qquad (X\theta - y)^T (X\theta - y)$$
$$subject\ to \qquad \sum_{i=1}^{n} |\theta_i| \leq t$$

- XGBoost
  XGBoost is an open-source software library which provides the gradient boosting framework for machine learning. XGBoost is widely used in Kaggle competitions and other machine learning objectives due to its scalable, portable and distributed features.
  In our work, we leverage the regression model API in open source python XGBoost library on various input feature combinations. We set our regressor in the gradient boosting mode and then tune on parameters including maximum depth of a tree, minimum sum of instance weight (hessian) needed in a child, subsample ratio of the training instance, etc.

Feature Selection Methods

The number of CNN features is well above 4000, larger than the number of samples which is around 3000. Therefore, we may experience serious overfitting if we use all these features directly in the regression. In the experiments, we try possible feature-selection approaches as below:

- 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 makes lasso regression a valid feature selection method.
- Variational Autoencoders
  Variational Autoencoder (VAE) has emerged as one of the most popular approaches to unsupervised learning of complicated distributions. VAEs are built on top of standard function approximators (neural networks), and can be trained with stochastic gradient descent. First, we construct a encoder-decoder pairs. The encoder encodes input 4096 CNN features and output 2 dimension compressed features, then the decoder uses them to recover input. Both encoder and decoder are neural network with 256 intermediate latent variables and ReLU activation function.
- Forward Search
  The forward feature selection procedure begins by evaluating all input attributes feature separately. Then it extends to feature subsets. In other words, we start by measuring the leave-one-out cross validation error of the one-component subsets, so that we can find the best individual feature. We then repeat this process over the remaining and generate a sorted list of features.

Experiments

Dataset and Benchmark

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 nightlights 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. R^2 score is widely used as our performance indicator.

Meanwhile, to compare our selected features with night-lights intensives. We apply linear regression on night-lights intensities, and get 0.5147 R^2 score in both train and test set. This score can be treated as our baseline.

Feature Selection

First, we run straightforward linear regression using all 4096 CNN features and get 0.995 training R^2 score, but the test R^2 is below zero. This clearly is overfitting.

To address the problem, we first try 2 basic feature selection methods. First, we simply select features starting from the beginning of the 4096 CNN features. 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 is based on the linear dependency of different feature assets, but our CNN features are not linear dependent for the non-linear activation function used in VGG net architecture. Meanwhile, PCA loses information of the original features. Both train and test R^2 scores are shown in the appendix.

Then, we applied correlation-based feature selection use features with the highest correlation with the labels, the result is similar to selecting features from the beginning. When we use more than 200 features, overfitting becomes obvious. We also find that the correlation is not an excellent metric in selecting features. We then use forward-search feature selection, i.e. starting from an empty feature set, each time adding one single feature that raises the test performance the most. Specifically, we have two loops. In the outer loop, we start from an empty feature set, and loop until overfittting occurs (e.g. 300 epochs). In the inner loop, we traverse through all features, each time add one feature into the current feature set, test the performance, and record which feature gives the most performance increase. We add this feature into the current feature set. We periodically pause to check overfitting. In the forward search, the performance is evaluated using ridge regression. We notice that the test performance begins to stay around certain level when the number of features exceeds 180. We then take the first 300 features selected by forward-search, and train our prediction models using linear regression, ridge regression and Lasso regression. Again, overfitting becomes obvious in linear regression and ridge regression when the number of features is larger than 180.

Then we turned to Lasso regression. As shown in **Figure 2, Appendix A**, 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 2, Appendix A**, shows that as the number of features gets large, Lasso almost shows 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.

Finally, we check the feature indices and show the overlapped features in **Figure 2**. In the figure we observe that the three effective methods selected features with few overlapping. We select all 63 features in the overlapped areas and form a new combination called common features for further investigation.

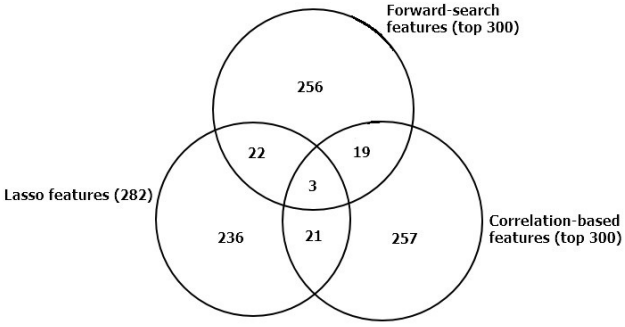


Figure 2: Common features shared by selection methods

Regression

In this section, we test our selected features on different regression models. As shown in **Table 1**, we can see that VAE features are inadequate for our task. Then reason behind this may come from the high loss in the decoder when trying to recover the original 4096 features. Nonetheless, we see that the features selected by forward-search perform well in all three regression experiments, meaning we have found a set of features that is robust and gives stable performance. The common features (those appearing in at least two feature sets in **Figure 2**) and lasso features preform well, but do not increase the performance by a lot compared to our baseline.

We also try XGBoost to train our model of predicting. XGBoost introduces a number of parameters to tune for the model to best fit the data. After carefully researching the parameters, we find that:

1. Its almost impossible to find the global minimum through manually tuning the parameters;
2. For different sets of features, we need different sets of parameters for the model to perform better;
3. The model is vulnerable to overfitting, in the sense that controlling overfitting sometimes reduces test performance at the same time.

For the experiment, we try to find the best set of parameters for different sets of features through grid searching. And we show the result in **Figure 3**. The model trained by

Table 1: R^2 score of Selected Features on Regression Models

	Model(Train|Test)			
Features	Linear Reg	Ridge Reg	Lasso Reg	XGBoost
Forward Search	0.6749|0.6416	0.6747|0.6438	0.6724|0.6426	0.8748|0.5094
Common Features	0.5606|0.5271	0.5605|0.5283	0.5552|0.5311	0.7591|0.3493
Lasso Features	0.6433|0.5027	0.6419|0.5170	0.6681|0.5863	0.8787|0.4970
VAE Features	0.5331|0.3042	0.6559|0.2918	0.6900|0.3199	0.5677|0.3391

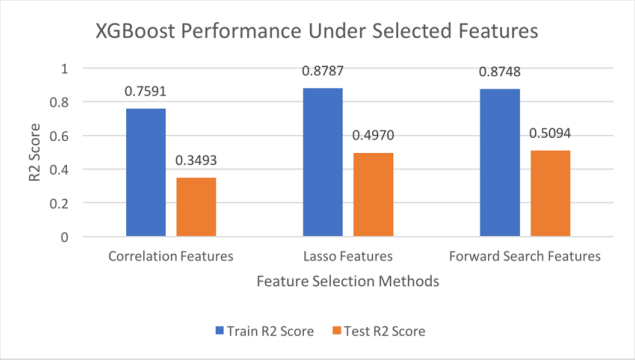


Figure 3: XGBoost performance

forward search features is providing the best performance, which is consistent with our previous experiment. However, all three models are subject to overfitting to some extent. As stated above, its hard to control the overfitting while keeping the test accuracy at the same time. Overall, we conclude that XGBoost is not the most stable model to perform different sets of model fitting, because changing the features means that we have to tune the parameters again. Further research needs to be done to better utilize the XGBoost.

As forward select features has the best results in regression model, we conduct experiments and plot figures to evaluate how regression R^2 score changes as we increase the features selected. As **Figure 4 and 5** shows, test performance begins to go down when the number of features exceeds 180. We can choose the first 180 selected features as our final decision for application.

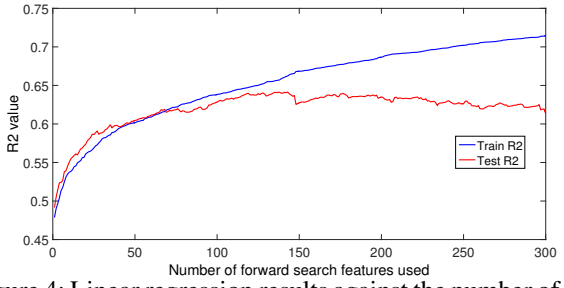


Figure 4: Linear regression results against the number of features in forward search method used

Conclusion

In this paper, we present a method that leverage CNN as an feature extractor for poverty prediction in African countries. Compared to previous works, we improve CNN features' strength by selecting CNN features to obtain the best combination for regression models. Through experiments on various feature-regressor combinations, we find that forward search is best feature selection method for this project, and the top 180 searched features are the most robust features for the regression model. Our features outperform night-lights intensities in poverty predictions, which is a step forward to provide validated information for tackling economic issues.

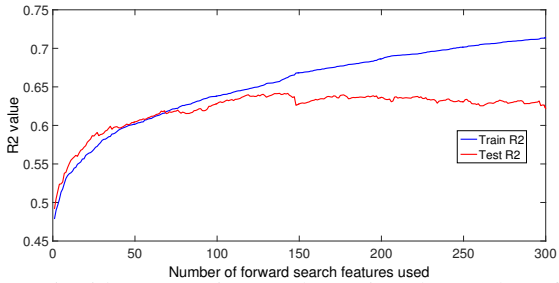


Figure 5: Ridge regression results against the number of features in forward search method used

Contribution

Zhihan Jiang works on XGBoost setup and tuning, poster design while Yicheng Li works on feature extraction and regression model on MATLAB. Zhaozhuo Xu takes CNN features extraction, VAE and report writeup.

Acknowledgements

We thank Neal Jean and Anthony Perez for their generous advice during our research.

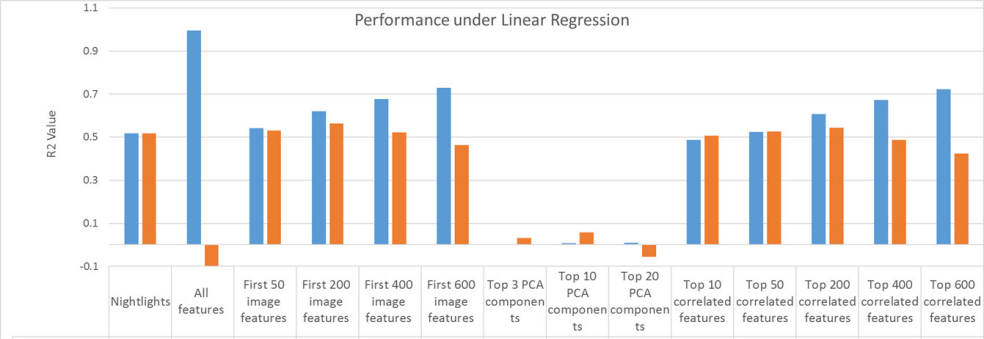
References

[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.

[Simonyan and Zisserman 2014] Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

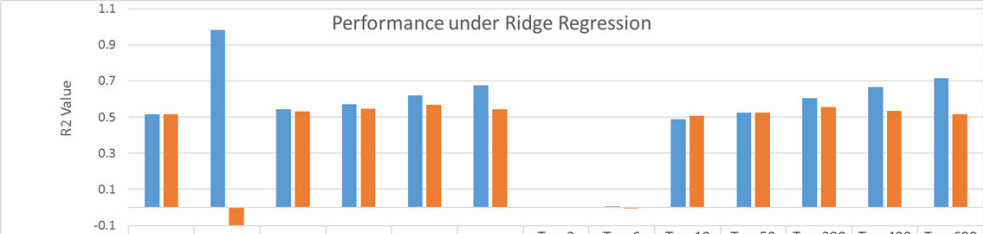
[Xie et al. 2016] Xie, M.; Jean, N.; Burke, M.; Lobell, D.; and Ermon, S. 2016. Transfer learning from deep features for remote sensing and poverty mapping. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 3929–3935. AAAI Press.

Appendix A



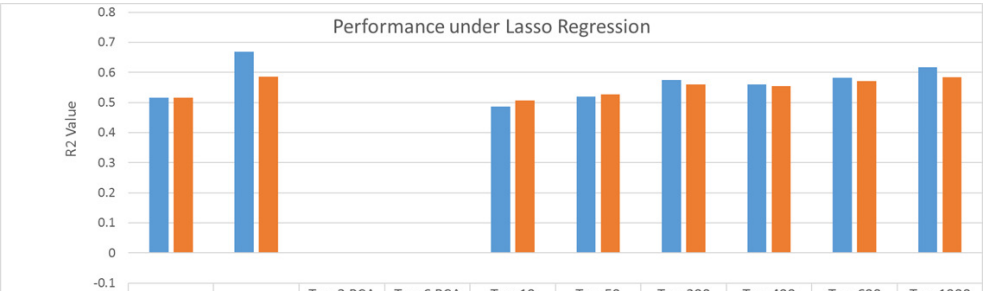
	Nightlights	All features	First 50 image features	First 200 image features	First 400 image features	First 600 image features	Top 3 PCA components	Top 10 PCA components	Top 20 PCA components	Top 10 correlated features	Top 50 correlated features	Top 200 correlated features	Top 400 correlated features	Top 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 1



	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 2



	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 3