Predicting Poverty of a Region from Satellite Imagery using CNNs

Master Thesis

Author

Ratih Ngestrini

Supervisors

Dr. R.W. Poppe

Dr. A.J. Feelders

Universiteit Utrecht

Computing Science
Department of Information and Computing Science
Utrecht University

16 January 2019

Abstract

Poverty in a socioeconomic context can be defined as the inability of individuals to meet their basic needs. Measuring poverty is important to target efforts in places that need aids the most and evaluate the effectiveness of government programs. However, it is difficult and expensive as it requires the collection of detailed data from the households. The development of machine learning-based techniques has enabled the use of big data such as social media, mobile phone, and satellites for poverty measurement. In this thesis, Convolutional Neural Network (CNN) models are evaluated to directly predict poverty from daytime satellite imagery. Two approaches, naive and semantic segmentation, are proposed and compared with the multistep learning approach that uses nighttime lights image. We perform experiments using publicly available daytime and nighttime satellite images from Google Maps and NOAA. The best model is achieved by combining the semantic segmentation approach and the night lights data. Moreover, we test the generalizability of the models using higher-level administrative and out-of-country data. The test reveals that we can use the models to estimate poverty in higher level administrative region, but they are not robust to be used to predict poverty in other countries.

Keywords: poverty estimation, machine learning, Convolutional Neural Networks, deep learning, satellite imagery

Contents

Chapter 1

Introduction

1.1 Background

Poverty is one of the fundamental global issues that needs serious attention from governments, especially in developing countries. The world is committed to ending poverty by 2030 as stated in one of the 17 Global Goals in The Sustainable Development Goals (SDGs) [1]. Poverty can be defined in many ways referring to its various dimensions such as social, economics, and politics. However, poverty is often defined in its socioeconomic context. It is described as the inability of individuals or households to meet their basic needs including food, clothing, and shelter to ensure a decent life in a society. Typically, poverty is measured to determine whether they are considered to be poor or not. The policy makers rely on that measurement, either at regional or household level, to direct their efforts in places that need aids the most. Moreover, they need the measurement to monitor and evaluate the effectiveness of the programs designed to improve human livelihoods. A standard measurement is also required to make comparisons between different time periods and regions. Most countries use household income or consumption as a basis for determining the well-being of the poor [2]. A value called poverty line is set and employed as a fixed threshold where those who fall below this value are considered as being poor, and those above it are not [3]. Unfortunately, obtaining a timely and reliable measurement is difficult and expensive as it requires the collection of detailed data directly from the households.

Given the difficulty of traditional data collection, in the past several years, the novel sources of data which can potentially be harnessed to estimate poverty have been explored such as data from social media, mobile phone, or satellites. The recent development of machine learning-based techniques has enabled novel data-intensive approaches to the measurement of poverty [4]. Previous studies have shown that search-engine and social media data can be used to forecast the value of economic indicators in various sectors [5], [6]. Another approach using mobile phone metadata demonstrated that mobile phone history could be used to predict the socioeconomic characteristics of an individual accurately, as well as to reconstruct the distribution of wealth and poverty of an entire nation [7].

1.2 Poverty Estimation from Satellite Imagery

Satellite images for domain-specific analysis such as imagery, geophysics, demographics, climate, and weather are publicly available [8]. It raises opportunities to use these data when there is a lack of resources and infrastructure to produce reliable data. As producing such data is expensive and time-consuming, a number of studies have been conducted to examine the potential use of publicly available satellite data to augment or replace the existing data. The initial work of exploring satellite data for measuring socioeconomic parameters is initiated with a research on night lights data. The data are obtained by measuring the intensity of lights captured passively by satellite and introduced in the fields of Economics as a supplement to the national survey data. The studies show that the brightness of visible lights from satellite views is strongly related to both population density and economic performance of a whole country or region. Furthermore, the luminosity contains useful information that can be utilized to measure Gross domestic product (GDP) per capita at the national and subnational levels [9], [10]. The night lights data are then used extensively in other researches to estimate the economic development and growth [11]–[13], to investigate the economic implications of urban geometry [14], and to assess the relative quality of GDP per capita [15]. The results of these studies tell us the extent in which satellite data can give insights on economic development in regions across the world.

Meanwhile, daytime satellite imagery emerged as a new source of information on economic activity [16]–[21]. Daytime imagery is taken at a much higher resolution than nighttime imagery. It contains visible features such as building areas, roads, cars, crops, and roof types that make it possible to identify the well-being of a region. Jean et al. introduced a novel deep learning approach to extract the landscape features from daytime satellite imagery that are indicative of poverty. In this approach, a Convolutional Neural Network (CNN) algorithm is applied first to learn the relationship between daytime and nighttime satellite images. Then, a ridge regression model uses the extracted image features and survey data to predict the average household consumption of enumeration areas. The resulted model predicts reasonably well the spatial distribution of economic welfare across the five countries it is tested in [16]. While this innovative method improves the performance of the model using the night lights data alone, it is not necessarily optimal for predicting poverty. It can explain an average of 46 percent of the variation of household consumption. Due to the scarcity of labeled survey data, this research applies a multistep learning technique and uses the nighttime lights intensity as an intermediate explanatory variable for estimating the average household consumption and asset wealth. In contrast to the nighttime lights intensity, there is much less information about whether the landscape features in the satellite imagery are also informative enough for estimating poverty. So far, there has been little discussion about the direct application of CNN model to the daytime satellite imagery for this task.

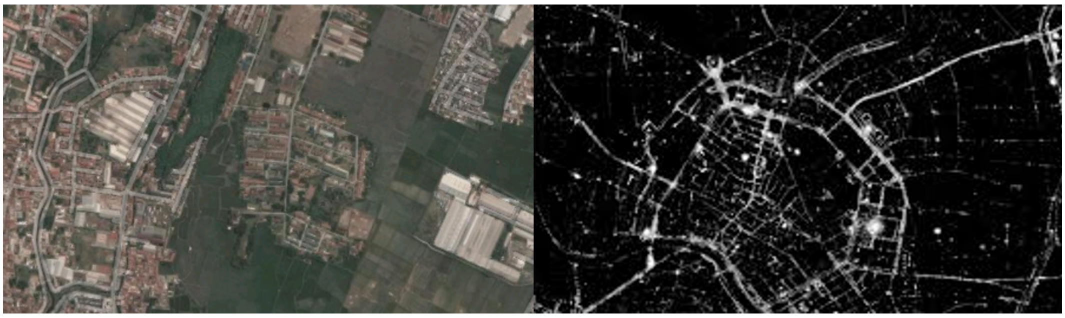


Figure 1.1: Example of daytime satellite image (left) and nighttime lights image (right)

1.3 Research Goal

The goal of this thesis is to design and examine the CNN models to directly predict poverty from daytime satellite imagery. Moreover, we intend to compare the performance of our models with a model utilizing nighttime lights image. The image data required as the input model are publicly available from Google Static Maps API and National Geophysical Data Center - NOAA. We use labeled training poverty data from Indonesia socioeconomic survey data. Indonesia is a developing country with varied geographic, demographic, and socioeconomic characteristics. No previous study has investigated the poverty prediction using satellite imagery data from this country. We replicate the multistep learning approach introduced by Jean et al. using the data from this country. Then, we compare the results with our proposed methods. In the end, this study will provide insights on how to leverage new datasets and machine learning for estimating poverty across Indonesia. The result of poverty measurement with big data could also be used to complement the national survey data and bring new opportunities to capture other socioeconomic conditions. Moreover, to examine the generalization of the fitted model, we test the models using data from two other countries, namely Thailand and Sri Lanka. We want to see whether the models that are trained using data and satellite image features from one country could be used to estimate the well-being of regions in other countries.

1.4 Research Questions

In this thesis, we address the following research questions:

1. *How good is the performance of the model if we estimate the poverty of a region only from the satellite imagery?*

   We build CNN models using the satellite imagery and poverty data from Indonesia and evaluate how well the models can predict poverty directly from daytime satellite images. We propose two approaches to build the models: naive and semantic segmentation approaches. The multistep learning approach by Jean et al. is used as a baseline in our research.

2. *How well does the model perform in comparison to both the model using nighttime lights image alone and the model using a combination of daytime and nighttime lights satellite images?*

We build two other models that utilize nighttime lights satellite images. First, we build a regression model using the night lights intensity as the only predictor. Second, we include that variable as one of the predictors in the semantic segmentation approach. Then, we evaluate and compare the performance of models with the models in *RQ 1*.

3. *Which landscape features are most correlated with the measure of poverty?*

   In the semantic segmentation approach used in *RQ 1*, the landscape features are extracted first from the satellite images and then used as predictors for estimating poverty. We analyze and evaluate those features such as vegetation, ground, road, building, water, and other landscape structures to see which features are highly predictive for poverty.

4. *How does the level of region for training data affect the result of the model implemented in a higher level of the administrative unit? How is the generalizability of the model to predict poverty in other countries?*

   In this experimental study, we use the ground truth data in the municipality level. We also examine if the model fitted using those data can predict well the poverty in a higher level of the administrative unit, namely province. In addition, the models are tested using the satellite imagery of Thailand and Sri Lanka. Since the training and testing countries have different units of poverty, before the testing process, we standardize the ground truth poverty data. We assess whether the performance in the testing set increases or decreases significantly compared to the test results in *RQ 1*.

Different approaches and types of image are used in this research. We investigate whether our proposed approaches (naive and semantic segmentation) will improve the existing approach (multistep learning) and if the daytime satellite imagery is more relevant to the poverty estimation rather than the nighttime lights data. We evaluate the results of each experiment to find the best solution for the described problem.

The remaining part of this thesis proceeds as follows: Chapter 2 provides literature reviews describing the standard poverty measurement, the theoretical background of a CNN model, and the studies that are relevant to this research. Chapter 3 aims to explain the data and methods mentioned above in detail. In this chapter, we also describe what we compare and how we conduct experiments to address each research question. Finally, Chapter 4 comprises the discussion and conclusion of this research.

Chapter 2

Literature Review

In this chapter, we discuss the literature review of this study. First, we present the standard approach for measuring poverty and then followed by the theoretical background of the machine learning models that are used in the proposed methods discussed in the next chapter. Subsequently, we present some related works that are relevant to this research. This information gives more insights on the poverty prediction and the Convolutional Neural Network algorithm used in the experiments.

2.1 Poverty Measurement

A poverty line is calculated in order to determine the well-being of households. Those whose expenditure (or income) falls below the line can be categorized as poor. There are three methods to construct that measurement: the cost of basic needs, food energy intake, and subjective evaluations [2]. The *cost of basic needs* is the most used approach. It first estimates the cost of acquiring enough food for adequate nutrition (usually 2,100 calories per person per day) and then adds the cost of other essential needs such as clothing and housing. When the price information is unavailable for the first approach, the *food energy intake* method can be applied. This method plots expenditure (or income) per capita against food consumption (in calories per person per day) to determine the expenditure (or income) level at which a household obtains enough food. The last method, *subjective evaluations*, is based on asking people about minimum income level that is needed just to make ends meet.

Practically, the construction of this poverty line is the most challenging step in the measurement of poverty. Using the first method, poverty is seen as an economic inability to meet basic needs, including food and nonfood, which are measured by expenditure. Once the consumptions are calculated, we need to determine whether that amount of expenditure can put the household in poverty category or not. Here, the poverty line acts as the decisive measure. It defines the level of expenditure required by an individual to fulfill his basic food and nonfood needs in order to escape poverty. As the cost of living across the world varies, the World Bank set the international poverty line \$1.90 per person per day as a global threshold. Those who have the average spending per capita per day below the threshold are defined as poor [2], [22]. Using the basic needs approach, the poverty line (Z_{BN}) can be formulated as

$$Z_{BN} = Z_F + Z_{NF} \tag{2.1}$$

The food poverty line (Z_F) is the total value of expenditures from basic food commodities that are measured based on the *cost of basic needs* concept. The cost to meet basic needs in each region will undoubtedly varies because it is influenced by the market price and the number of commodities consumed. The poverty line of each region will determine the number of the poor's population and also represent the level of welfare of the region. The formula to compute the food poverty line of a region d (Z_F^d) can be written as follows

$$Z_F^d = \sum_{c=1}^{n} P_d^r Q_i^d = \sum_{c=1}^{n} V_c^d \tag{2.2}$$

where c is the index of food commodity, P_c^d is the average price of food commodity c in region d, Q_c^d is the average quantity of food commodity c consumed in region d, and V_c^d is the expenditure value of food commodity c in region d. Meanwhile, the nonfood poverty line (Z_{NF}) is the sum of the minimum needs of selected nonfood commodities including housing, clothing, education, and health. The selection of nonfood commodities, goods and services, is tailored to the consumption patterns of the residents.

$$Z_{NF}^r = \sum_{c=1}^{n} r_c^d V_c^d \tag{2.3}$$

where c is the index of nonfood commodity, r_c^d is the ratio of commodity expenditure c for region d, and V_c^d is the expenditure value of food commodity c in region d. The average price of commodity and the consumed quantity of those commodities are obtained from the socioeconomic survey, while the ratio of commodity expenditure are obtained from the basic needs commodities survey.

2.2 Machine Learning Methods

Nowadays, data mining is extensively used in diverse areas. Data mining is not a technology push. The rapid growth of data is one of the reasons why data mining is needed in this era of big data. Data mining is used to discover patterns and relationships in the datasets to solve problems through data analysis. There are different approaches to mine those properties of datasets. Machine learning is one of them [23]. Basically, machine learning is a computer algorithm that learns the data without having to be programmed explicitly. Simply put, given a model, or structure, that are defined by some parameters, the algorithm will optimize those parameters using the training data. Then, we can use that model to predict new data [24].

2.2.1 Classification and Regression

Machine learning algorithms are classified into categories according to their purposes such as supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. However, the first two categories are the most common ones. In supervised learning, the data used to train the algorithm are already labeled with correct values. This process requires prior knowledge of what the output values for our samples should be. The algorithm will learn the function that approximates the relationship between input (X) and output (Y) observable in the data. Unsupervised learning, on the other hand, only has the input data (X) and no corresponding output variables. The algorithm attempts to find patterns within a dataset based on the natural structure such as similarity or dissimilarity between data points.

Supervised learning problems can be further grouped into regression and classification tasks. A classification problem is when the output variable is a category, such as Yes/No, True/False, or multi-label cases. The model will estimate the probability of a given input data (X) belonging to each output class (Y). Logistic regression is a well-known technique for classification task. The logistic function can be defined as follows:

$$f(X) = \frac{1}{1 + e^{-\theta X}} \tag{2.4}$$

The function above is also called continuous log-sigmoid function. It takes any input in the range of negative to positive infinity and maps it to output in the range of 0.0 to 1.0. The value can be interpreted as the likelihood of a given example belonging to each class. The predicted probability then can be converted into a class label by selecting the class that has the highest probability $P(Y|X)$.

In contrast, a regression problem is when we want to map the input to a continuous output like weight, value, or price. The model will estimate the mapping function that defines the relationship between the independent variables $X = [X_1, ..., X_p]^T$ and the dependent variable Y. Regression, in general, is about learning model f

$$Y = f(X) + \epsilon \tag{2.5}$$

where ϵ is some noise/error which describes everything that cannot be captured by the model. Specifically, we view ϵ as a random variable that is independent of X, normally distributed and has mean zero $\epsilon \sim \mathcal{N}(0,\, \sigma_\epsilon^2)$.

Linear regression (Ordinary Least Squares), which is a straightforward approach to regression, has a linear (or affine) combination of the input variables X as its model f. Mathematically, the relationship can be written as

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon \tag{2.6}$$

where the coefficients $\beta_0, \beta_1, ..., \beta_p$ are known as the *parameters* in the model. Even though the linear regression is relatively simple, it is still surprisingly useful on its own. Furthermore, it constitutes an important building block in more advanced algorithms such as deep learning or Convolutional Neural Network (CNN) that will be explained in the next section. The task of regression is to estimate the value of *parameters* $\beta_0, \beta_1, ..., \beta_p$ from training datasets $D = \{(x_i, y_i)\}_{i=1}^n$, so we can use the model to make a prediction $\hat{y}$ of some (not yet seen) output y for some test input x. The goal of the algorithm is to find the model that fit the data well. There are a number of strategies to learn the unknown parameters β such that the resulted regression line is as close as possible to all training data points. However, the most common approach involves minimizing the *least squares* criterion. Let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p$ be the prediction of Y based on the value of X. Then $e_i = y_i - \hat{y}_i$ represents the ith residual - the difference between the true value and the predicted value of the ith observed data point by the linear model. The *residual sum of squares* (RSS) can be defined as

$$RSS = e_1^2 + e_2^2 + ... + e_n^2 \tag{2.7}$$

$$RSS = \sum_{i=1}^{n}\left(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_{ij}\right)^2 . \tag{2.8}$$

The least squares approach will choose $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p$ that minimize the RSS.

Even though the linear regression model may seem fairly rigid and non flexible, it is not necessarily so. We can extend the model with nonlinear transformations, for instance, $X, X^2, ..., X^p$ as inputs, and thus obtain a linear model which is a polynomial in X (nonlinear). When a model is too simple, it might be inflexible in learning from the dataset and will produce poor predictions. At the same time, a complex model may not perform well in testing datasets due to *overfitting*. We need to select the right model in between simple and complex model.

To improve the linear model in terms of prediction accuracy and model interpretability, different methods are proposed to find a better model. James et al. classifies three major model selection techniques: subset selection, dimension reduction, and shrinkage [25]. *Subset Selection* builds a model using least squares on the reduced set of variables that are most predictive to the response variable. Meanwhile, the *Dimension Reduction* will transform the predictors and then fit a least squares model using the transformed variables. Instead of combining or reducing the input variables, as an alternative, we can fit the model using a technique that constrains or *regularizes* the coefficient parameters relative to the least squares estimates, or equivalently, that shrinks the estimates $\hat{\beta}$ towards zero. This shrinkage (also known as *regularization*) has the effect of reducing variance, and it helps to handle the overfitting problem. The idea is that the model with small parameter values should be preferred if it fits the data almost as well as a model with larger parameter values [26]. The two best-known regularization techniques are *ridge regression* and *lasso*.

Ridge Regression

Ridge regression (also known as *Tikhonov regularization*, *ℓ_2 regularization*, or *weight decay*) implements minimization technique in a similar way as the least squares approach. While in the least squares we find the coefficient by minimizing the RSS, the ridge regression coefficient parameters ($\hat{\beta}^R$) are the values that minimize

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2, \tag{2.9}$$

where λ is a *tuning parameter*, or also called hyperparameter. The minimization Equation 2.9 consists of two terms. As the least squares, ridge regression tries to find coefficient parameters that fit the data well by making RSS small. The second criteria or also called *shrinkage penalty*, $\lambda\sum_{j=1}^{p}\hat{\beta}_j^2$, is small when $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p$ are close to zero, and so it has the effect of *shrinking* the parameters of $\hat{\beta}_j$ towards zero. The regularization parameter λ is required to control the relative impact of these two terms on the regression coefficient parameters. For $\lambda = 0$ this regularization method will result the same as the original least squares, whereas if we set $\lambda \to \infty$, the impact of the shrinkage penalty grows, and it will force all parameters $\hat{\beta}_j$ to approach zero. Selecting a good value for λ is critical, and it depends on each regression problem. It can either be determined by manual tuning or in a more systematic fashion by using *cross-validation*.

Lasso

As previously mentioned, ridge regression will include all the predictors in the final model. The shrinkage penalty $\lambda \sum_{j=1}^{p} \hat{\beta}_j^2$ will shrink the regression coefficient to zero, but it will not be exactly equal to zero. It improves the prediction accuracy, but it does not help to make the model more interpretable. It can create a challenge to understand the model since the number of predictor variables is quite large. Tibshirani introduced the *lasso (Least Absolute Shrinkage and Selection Operator)* technique, or equivalently *L1 regularization*, as a solution to the ridge regression problem [27]. Lasso is able to achieve both the accuracy and model interpretability by forcing the coefficients of the less important features to zero, which eventually will results in a simpler model that does not include those coefficients. The lasso coefficients ($\hat{\beta}^L$) minimize

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = RSS + \lambda\sum_{j=1}^{p}|\beta_j| \tag{2.10}$$

As can be seen, the lasso and ridge regression have similar minimization formulas. The key difference is that the lasso uses the absolute value of the coefficient as a penalty term to the function which is called ℓ_1 penalty. The ℓ_1 penalty has the effect of forcing some of the coefficients to be set to zero when the tuning parameter λ is sufficiently large. Therefore, much like the subset selection method, the lasso performs *feature selection.* This technique is a great alternative when we are dealing with a large set of features since it will only include a part of those features that are relevant to describe the response variable. In addition to making the model easier to interpret, it enables the algorithm to work faster, and ultimately also decreases overfitting. Moreover, as in ridge regression, selecting a good value of λ for the lasso is also critical.

2.2.2 Convolutional Neural Network (CNN)

In the field of machine learning, artificial neural network is a model that inspired by biological neural networks in the brain. It has two main components that follow the idea of how the brain works. The first is a *neuron* (or *node*) that is like a biological neuron, they are stimulated by inputs. These neurons will pass information they receive to other neurons, often with transformations. The second component is the *signal.* The artificial neurons will be trained to pass forward the useful signals to achieve the larger goals of the brain [28].

The behaviour of a neural network is described by its architecture. The neural network architecture is shaped by the number of neurons, the number of layers, and the types of connection between layers. The most well-known neural network is the *feed-forward multilayer neural network*, also often called *deep feed-forward network* or *multilayer perceptrons* (MLPs). This topology has an input layer, one or many hidden layers, and a single output layer. Each layer can have a different number of neurons, and the neurons in each layer are fully connected to all neurons in the adjacent layer. The paths connecting the neurons contain adaptive *weights* that can be tuned by the learning algorithm in order to improve the model performance [29].

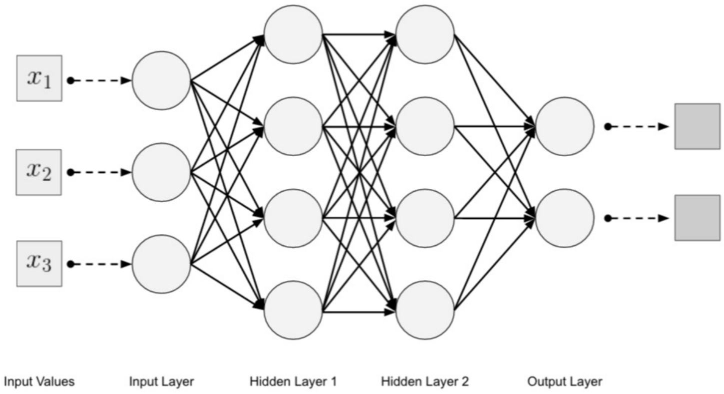


Figure 2.1: Multilayer Neural Network Topology

The *input layer* consists of the neurons that only receive the data and pass it on to the next layer. The number of neurons in the input layer is equal to the number of features in the dataset. The nodes in the *hidden layers* apply transformations to the inputs from previous layer before passing them on. As the network is trained, those nodes that are found to be more predictive if the outcomes are weighted more heavily. The *output layer* consists of a number of nodes depending on the type of model we are building. Neural network can be used for both regression and classification models. Hence, the final output may be a real-valued output (regression) or a set of probabilities (classification). In a classification model, there will be one node for each classification label, while in a regression model there will be only a single node that produces a value [30].

As any other machine learning models, a deep feed-forward network defines a mapping function $\hat{y} = f(x, \theta)$ and learns the value of the parameters θ (or denoted by β in the previous section) that results in the best function estimation of y. The models are called feed-forward because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output $\hat{y}$. The capacity of a neural network is represented by its size. When we increase the size of the input data and the number of layers in the network, the computational power of the model will also improve. The combination of new software, hardware, parallel algorithms, and a lot of training data enables the *deep learning* to be more powerful and makes a significant contribution to the field of machine learning. Deep learning has excelled in many applications, including computer vision, speech recognition, text generation, and language translation [26].

Gradient Descent

Machine learning algorithms train their models by solving optimization problems. Gradient descent is an optimization algorithm used to find the parameters (θ) of a function (f) that minimizes a cost (error) function. As previously explained, we expect the predicted value $\hat{y}$ is close to the actual value of y in the observed data. For example, in regression setting, we can use *mean squared error* (MSE) as the measurement of the cost function ($J(\theta)$).

$$J(\theta) = MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(x_i;\theta))^2 \tag{2.11}$$

Gradient descent is best applied when the parameters cannot be computed analytically using simple linear algebra. A model performs optimization on the training data to find the lowest error function, and we can check the performance of the model on the validation data.

In gradient descent, we imagine the function of the parameters (or called as *weights*) as a landscape. The hills represent locations (parameter or weight values) that give a lot of prediction error (cost) and valleys represent locations with less error (cost). Figure 2.2 below illustrates the cost function with only one parameter (weight), while Figure 2.3 depicts the function with two parameters. The algorithm basically does what we are doing by hand: change the weight value bit by bit, until we hopefully arrive at a minimum cost. We select one point on that landscape to place our initial weight (orange point in Figure 2.2) randomly or based on domain knowledge. The goal of the gradient descent is to move that weight downhill to areas of lower error as quickly as possible [28]. Generally, it is easier to get a better result when applying gradient descent on a convex function rather than on a non-convex function which has one local minimum and one global minimum. It is because once the process falls into a local minimum, it will be not easy to climb out and find the global minimum. If the function is convex, the local minimum is the global one.

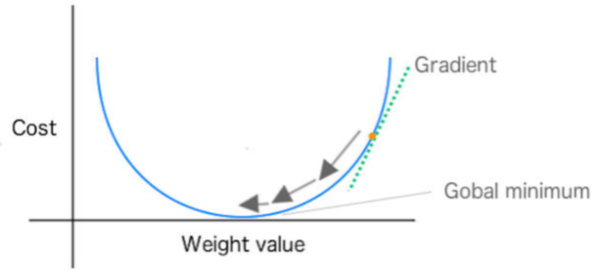


Figure 2.2: Weight (parameter) changes toward global minimum

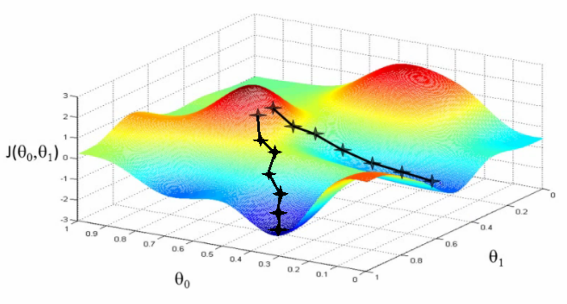


Figure 2.3: The gradient descent to find the *deepest valley* in the cost function with two parameters (θ)

The process will repeatedly tweak that weight value, measure the cost, and select a new weight value that has a lower cost until local or global minimum of the cost function has been reached (convergence). To determine the new weight value in each iteration, gradient descent will compute the *slope* or *gradient* ∇ (the change in error caused by a change in the weight) so that we know the direction (sign +/-) to move the weight value in order to get a lower cost in the next iteration. It does so by taking a derivative of the cost function $J(\theta)$ with respect to the parameters.

$$\nabla^t = \frac{\partial J(\theta)}{\partial \theta_j} \tag{2.12}$$

where t is the iteration, and j is the j-th parameter. Once we have the direction, the new weight value can then be updated. A *learning rate* parameter (η) must be specified to control how much the weight value can change on each update.

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla^t \tag{2.13}$$

The process is repeated until the cost value is converged, or ∇^t is zero or close enough to zero. As mentioned previously, the purpose of gradient descent is to find the optimal weight as quickly as possible. However, if we set the step size or the learning rate too big, we might not be able to find the minimum because we will overshoot it. On the other hand, if it is too small, the process will take too many iterations to get to the minimum, and we can get stuck in a local minimum.

Backpropagation Learning

The multilayer neural network is typically composed of multiple layers, each layer contains multiple neurons (I_i, H_j, O_i) and biases (B_i), and they are connected by the weights (w_i). Bias refers to the constant nodes in a neural network as can be seen in Figure 2.4. Furthermore, activation functions (HA_i, OA_i) are important features of the model. They decide whether a neuron should be activated or not. In other words, they will determine whether the information received by the neuron is relevant for the given information or it should be ignored [28], [31], [32].

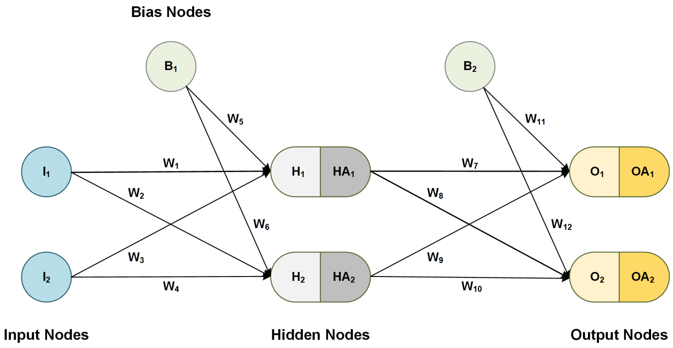


Figure 2.4: Simple architecture of a neural network

A multilayer neural network model learns the data using *backpropagation* process. In the model above, the data flows forward from the input layer to the output layer, and we compare the predicted results with the actual results and calculate the total error by the cost function. Then, the model will use the total error to modify the weights and biases in the network, working from the output neurons through the hidden neurons to the input neurons or going backward. Once the entire data has gone through this process, the final weights and biases can be used for prediction.

We can consider backpropagation as the implementation of gradient descent in multilayer neural network model. Backpropagation is basically an algorithm to compute the gradient that is needed to find the optimal parameters used in the multilayer neural network. The gradient can be calculated by taking its derivative with respect to the parameters (weights and the biases).

The pseudocode of the backpropagation algorithm for updating the weights in the network is presented in Algorithm 1. It starts with initializing the neural network and starts looping through the input examples (until it encounters a stopping condition or a maximum number of *epochs* or iteration). In the neural network terminology, one *epoch* or iteration means one forward pass and one backward pass of all the training examples. First, we compute the output of the current network for the current input example. We compare this output to the true value of output associated with the input and compute the error (*example_err*). Then, it will iteratively compute the weight updates leading to the output layer until the process is terminated [28].

Algorithm 1: Backpropagation algorithm for updating weights

```
Input  : network, training-records, learning-rate
Output: network

network ← initialize weights (randomly)
start loop
foreach example in training-records do
    /* compute the output for this input example */
    network-output ← neural-network-output(network, example)
    /* compute the error and the Δ for neurons in the output layer */
    example-err ← target-output - network-output
    /* update the weights leading to the output layer */
    $w_{j,i} \leftarrow w_{j,i} + \eta \times a_j \times Err_i \times g'(input\_sum_i)$
    foreach subsequent-layer in network do
        /* compute the error at each node */
        $\Delta_j \leftarrow g'(input\_sum_j) \sum_i w_{j,i} \Delta_i$
        /* update the weights leading into the layer */
        $w_{k,j} \leftarrow w_{k,j} + \eta \times a_k \times \Delta_j$
    end
end
end loop when network has converged
return network
```

Notation	Description
i	Index of neuron
j	Index of neuron in previous layer connecting to neuron i
$w_{j,i}$	Weight on the incoming connection from previous layer neuron j to neuron i
$\mathbf{W}_i$	Vector of weights leading into neuron i
b	Bias
g	Activation function
g'	Derivative of the activation function
a_i	Activation value of neuron i (output of neuron i); $= g(input\_sum_i)$
$\mathbf{A}_i$	Vector of activation values for the inputs into neuron i
$input\_sum_i$	Weighted sum of inputs to neuron i; $= \mathbf{W}_i\mathbf{A}_i + b$
$input\_sum_j$	Weighted sum of inputs for neuron j in previous layer (used in backpropagation)
η	Learning rate
Err_i	Difference between the network output and the actual output value for the training example (cost)
Δ_j	Error term for connected neuron j in previous layer
Δ_i	Error term for neuron i; $= Err_i \times g'(input\_sum_i)$

Table 2.1: Neural Network notation

Activation Functions

The activation function is a nonlinear transformation over the input of a neuron to determine if that neuron should be activated or not. It is really important because it gives the neural network its nonlinear capability. Without the function, the model would simply be a linear regression which has limited power, and it would not be able to learn complicated, high dimensional, nonlinear, and big dataset such as images, videos, audio, speech, etc. It basically maps the resulting values in between 0 to 1 or -1 to 1 depending upon the function. This transformed output is then sent as an input to the next layer of neurons. The following are most commonly used activation functions [33].

Sigmoid Function It is usually implemented for classification setting in the output layer since it produces output between 0 and 1. Therefore, we can easily interpret the output as probabilities.

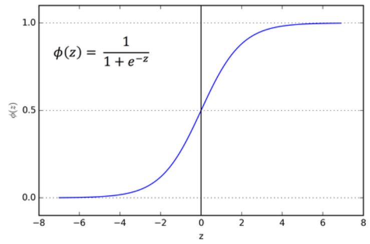


Figure 2.5: Sigmoid Function

Hyperbolic Tangent Function This function is similar to logistic sigmoid but having stronger gradients. The range of the function is from -1 to 1. The negative inputs will be mapped strongly negative, and the zero inputs will be mapped near zero in the tanh graph. The output of the activation units will be around zero.

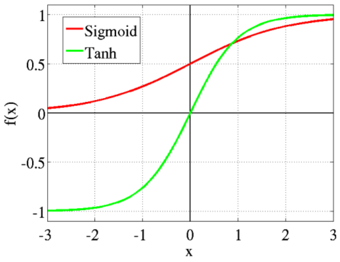


Figure 2.6: Hyperbolic Tangent Function

Rectified Linear Unit (ReLU) The ReLU is the most used activation function, especially in convolutional neural network. Since it has a lot of the properties of linear functions, it tends to work well on most of the problems. Moreover, it was found to greatly accelerate the convergence of gradient descent compared to the sigmoid and tangent functions. It has a range from 0 to ∞. The problem with this function is that the derivative of this function is not defined when $z = 0$.

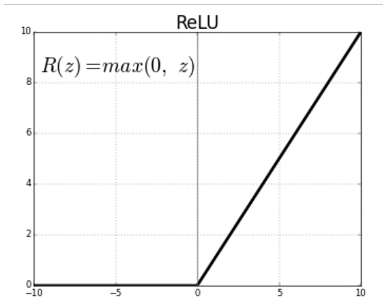


Figure 2.7: Rectified Linear Unit (ReLU)

Leaky Rectified Linear Unit This function overcomes the zero gradient issue from ReLU. Once a ReLU ends up in this state, it is unlikely to recover, and gradient descent learning will not alter the weights. Leaky ReLU with a small positive gradient for negative inputs when $z < 0$ are one attempt to address this problem and give a chance to recover.

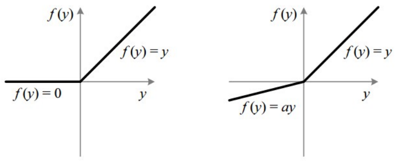


Figure 2.8: Leaky Rectified Linear Unit

Convolutional Network

Convolutional network, also known as *convolutional neural network* (CNN), is introduced to handle large features in image and speech recognition tasks [34]. Those kind of tasks cannot be done only with an ordinary fully connected feed-forward network. The ordinary neural network will understand the mapping between inputs and outputs but not the patterns in the inputs. In image or speech data, the features (or pixels) that are spatially or temporally nearby are highly correlated. Convolutional network forces the extraction of those correlations (local features) to recognize the inputs. The CNN has been tremendously successful in practical applications, especially in machine vision like self-driving cars, robotics, drones, and treatments for the visually impaired. Similar to the ordinary neural network described previously, the CNN is made up of neurons that have learnable weights and biases. It is a variant of neural network specialized for processing data that has a grid-like topology like image and audio that have a specific set of repeating patterns that are related spatially. In this section, we will focus only on the image data.

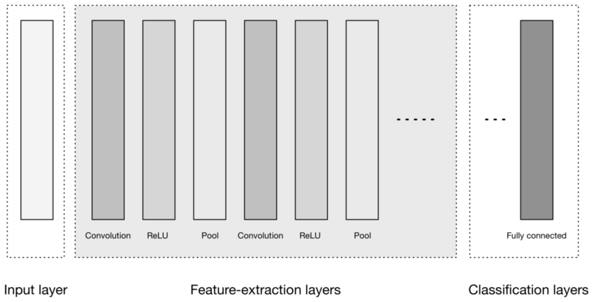


Figure 2.9: High-level architecture of CNN

Figure 2.9 depicts three major parts of the CNN: input layer, feature-extraction (learning) layers, and output layers [28]. In image recognition task, the *input layer* receives three-dimensional input in the form of the size of the image (width and height) and the depth representing the color channels (generally three for RGB color channels). The *feature-extraction layers* are composed of a repeating sequence of convolutional and pooling layers with the Rectified Linear Unit (ReLU) as the activation function. These layers extract a number of features in the images and progressively construct higher-order features. Finally, the last components of CNN, *output layers*, have fully connected layers to take the higher-order features and produce class probabilities for the classification task, or a real value for the regression.

Convolutional Layer

Convolutional layer is the core building block of CNN architectures. The goal of this layer is to learn higher-order features in the data via a mathematical operation called *convolution.* Simply put, it transforms the input data by using a square patch (kernel or filter) that is locally connected to the neurons in previous layer. This layer will compute a dot product between the region of the neurons in the input layer and the values in the *kernel* (weights).

In general form, *convolution* is defined as a mathematical operation on two functions of a real-valued argument in order to merge them. In CNN terminology, the first argument refers to the *input*, the second argument is the *kernel*, and the output of the convolutional is called *feature map* or *activation map*. Figure 2.10 shows the example of the operation in which the values in the input are the pixel values of the image [35].

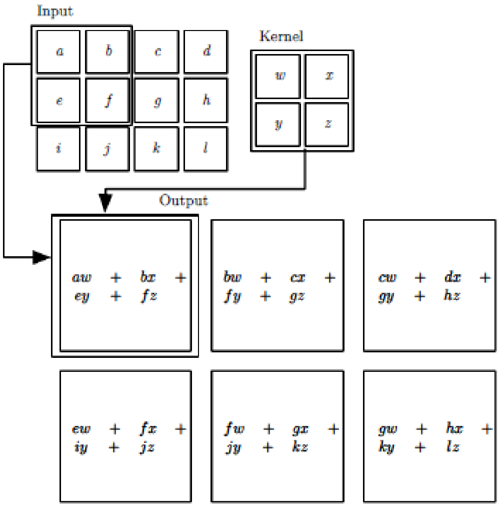


Figure 2.10: Example of convolutional operation

The figure above illustrates how the kernel is slid across the input data to produce the convoluted features. At each step, the dot product between the kernel and the input data values within its bound is calculated, resulting in a single entry in the feature map.

The kernel is a smaller square matrix than the image itself whose function is to find patterns in the image pixels. It is applied for every depth of the input volume, so the kernel has the same depth as the input. For illustration, the input volume is of size $5 \times 5 \times 3$ corresponding to an image with width $= 5$, height $= 5$, and depth $= 3$. Then, we define the kernel that is smaller spatially but the depth must be the same as the input, for example $3 \times 3 \times 3$. The values in the kernel represent the *weights* in conventional neural network. As a result of the convolutional operation, the two-dimensional layer feature map (also called *activation map*) is created in which each unit is a set of units located in a small neighborhood in the input space. This area in the input image is called *local receptive fields*, and it refers to the part of the image that is visible to one kernel (filter) at a time. Hence, the feature map will contain elementary visual features such as oriented edges, end-points, corner, etc. With the use of multiple kernels, we can create multiple activation maps. Besides, we can also perform convolution with bias. Bias is the value that we can add to each element in the activation map to add additional influence from neighboring pixels.

Pooling Layer

Typically, hidden layers of CNN consist of three stages. First, the layer performs several convolutions to produce a set of linear activations. Second, each linear activation is run through a nonlinear activation function, such as the ReLU function. In the last stage, the *pooling* function is used to modify the output of the layer further. This function is applied in the pooling layer to progressively reduce the spatial size of the data representation over the network and help control overfitting. It usually uses the *max()* operation to resize the input data. This operation is called *max pooling.* For example, if we have a 2 × 2 filter size, the *max()* operation will take the largest of four numbers in the filter area.

Fully Connected Layer

The output of a convolution layer is a set of activation maps, each of which can be seen as a sheet of neurons. An element or neuron in the activation map is corresponding to a small region in the image that represents high-level features in the data. The fully connected layer will learn the non-linear combinations of these features and result in a feature vector. This vector holds information that is vital and relevant to the input. When the network gets trained, this feature vector is then further used for classification, regression, or input for other networks.

Regularization

One aspect that must be considered in designing a machine learning model is avoiding *overfitting.* The overfitting problem happens when a model fits the training data too well. The model learns the noise in the training data resulting in a complex model that has low performance on unseen data. To avoid the problem, regularization is applied to learning algorithms. In linear regression, it makes slight modifications to the learning algorithm such that the model generalizes better. Similarly, in CNN, by regularizing, we want the weight of spurious features to be reduced, and only the good features that are generated. The following are the regularization methods that can be applied in deep learning to prevent overfitting [36].

Dropout Dropout is the most frequently used regularization technique in neural network. At every iteration during training phase, it randomly selects some neurons and removes them along with their connections. It helps to reduce interdependent learning amongst the neurons and prevents the neurons from learning too much. Each iteration has a different set of neurons, and this results in a different set of outputs so that we can view dropout as a form of *ensemble learning.* Ensemble method combines a number of weaker models into one predictive model. It usually performs better than a single model as they capture more randomness.

Batch Normalization Batch normalization means normalizing the input values of each layer during the training phase in such a way that they have a mean output activation of zero and standard deviation of one. It reduces overfitting because it has a slight regularization effect. Similar to dropout, it adds some noise to activations in each hidden layer. It is better to use batch normalization together with dropout [37].

Stochastic Depth Using stochastic depth, during the training phase, we randomly drop layers instead of neurons like in dropout technique. It aims to shrink the depth of a network during training and bypass their transformations through skip connections. It has a regularizing effect as dropout because the layers cannot easily co-adapt. Moreover, using this technique is similar to training an ensemble of networks with different depths. It reduces training time substantially and improves the test error significantly [38].

Transfer Learning

Several things determine the quality of the CNN model. One of these is the amount of data used to fit the model. More training dataset will produce a model with better performance. However, it is relatively rare to obtain large-scale datasets. Sufficient resources are also needed to develop the model from scratch. A strategy to cope with the issues is to use a previously trained CNN model that had good results and then train it further to perform another task. The idea is to take the knowledge learned in the pre-trained model and apply it to our task. This is referred to as *transfer learning.* It is implemented as a shortcut to speed up the training process and improve the performance of our deep learning model [28].

A pre-trained model is a model created by someone else to solve a similar problem. Our training dataset should share visual features with the base dataset used in the pre-trained model. The are lots of existing models that are already trained for various tasks such as object recognition, face recognition, semantic segmentation, ImageNet Large Scale Visual Recognition Challenge (ILSVRC) classification, etc. [39]. Two use cases of the pre-trained model are as follows:

Using an existing convolutional model as a feature extractor We take a pre-trained convolutional model, remove the last fully-connected layer, then treat the rest of the convolutional network as a fixed feature extractor for the smaller new dataset.

Fine-tuning existing model The second use case is to not only replace and retrain the model on top of the convolutional network on the new dataset but to also fine-tune the weights of the pre-trained network by continuing the backpropagation.

The diagram below helps to decide on how to proceed on using the pre-trained model in our case.

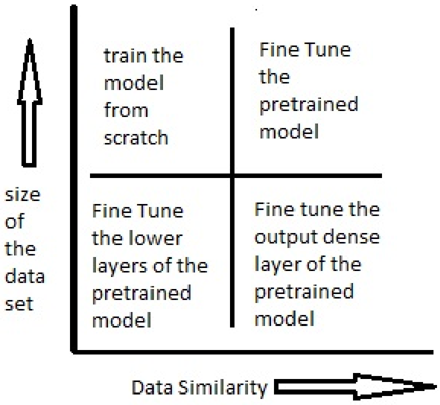

train the model from scratch
Fine Tune the pretrained model
size of the data set
Fine Tune the lower layers of the pretrained model
Fine tune the output dense layer of the pretrained model
Data Similarity

Figure 2.11: The use of pre-trained CNN model

2.2.3 Evaluation Metrics

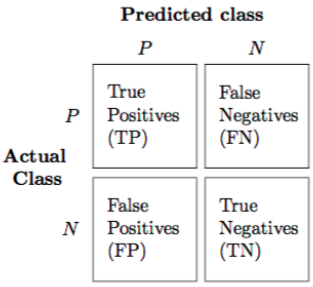
To assess the model performance, various metrics are used to compare the predicted values against the actual values. Regression and classification problems will have different performance measures. R^2 is mostly used to measure the result of regression task.

R **Squared** (R^2) is used for explanatory purposes and explains how well our independent variables explain the variability in our dependent variable.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} = 1 - \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{2.14}$$

The numerator is MSE (Mean Square Error) and the denominator is the variance in Y values. The higher the MSE, the smaller the R^2 and the poorer is the model.

For a classification task, we can create the confusion matrix as illustrated in the figure below and calculate the True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) cases in our matrix.



		Predicted class	
		P	*N*
Actual Class	*P*	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Figure 2.12: The confusion matrix

Accuracy is calculated as the portion of true labeled cases to total number of cases.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.15}$$

Precision is the number of correct prediction divided by the number of total predictions. In a multi-class classification, it can be calculated separately for each class. For each row (class x), we take the number on the diagonal, and divide it by the sum of all the values in the column.

$$Precision_x = \frac{TP_x}{TP_x + FP_x} \tag{2.16}$$

Recall is the number of correct predictions divided by the total number of cases present in that class. For each column (class x), it is the value on the diagonal, divided by the sum of the values in the row.

$$Recall_x = \frac{TP_x}{TP_x + FN_x} \tag{2.17}$$

F1 measurement is the harmonic mean of precision and recall, and acts as a combined measure of the precision and recall.

$$F1_x = 2 \times \frac{Precision_x \times Recall_x}{Precision_x + Recall_x} \tag{2.18}$$

2.3 Related Works

2.3.1 Applications of Satellite Data for Measuring Economic Indicators

In order to circumvent the lack of reliable data to measure economic development, works have been done to study the possible use of new data source such as satellite images to supplement the existing data. Most of the previous works using satellite images have used night lights as a proxy for economic development. For estimating GDP at the national level, the sum of light (SOL) intensity was extracted from the radiance-calibrated nighttime images at a spatial resolution of 1 km^2 and regressed against the official GDP value. Using this regression model, R^2 greater than 0.9 was obtained for 36 administrative units in China, India, Mexico, and the US used in the experiment [40]. In poverty measurement, the stable nighttime lights image was used along with the LandScan population grid to develop the grid of Poverty Index (PI) [41]. In another study, Average Light Index (ALI) was computed for a region on the basis of the total number of lit pixels. Regression analysis between PI and ALI for the 31 provinces in China gave an R^2 of 0.85 [42]. This research proved yet again the significance of the nighttime images as a proxy variable for analyzing poverty. In general, those research using nighttime lights image developed regression models to learn the relationship between economic activity and lights data. In the studies mentioned above, since these estimates can be made as spatially 1 km^2 grids, the greatest advantage of all these estimates is that they can be aggregated to any desirable mapping unit.

Daytime satellite images have also been an alternative source for measuring economic activity. With the development of machine learning techniques, machine learning has proven useful for a very large number of applications in geosciences and remote sensing. Nonparametric regression and classification can be used to tackle problems in that field of study using satellite imagery [43]. Same as nighttime lights image, in most works, the econometric analysis was not directly conducted from the satellite imagery. They used linear regression to estimate the economic parameters from variables extracted from the images. A recent work introduced a new approach to poverty prediction using satellite images. A multistep learning approach was proposed to overcome the scarcity of poverty data in African countries. It uses convolutional neural network (CNN) to learn the relationship between million of daytime satellite images and nighttime images. With the assumption that the features useful for predicting nighttime lights intensity are also useful for predicting poverty levels, in the next step, it builds linear ridge regression to predict poverty measures from the night lights feature vector output by the CNN. As the results, this model can explain 55 to 75% of the variation in average household asset wealth and 37 to 55% in average household consumption in the countries they examined [16]. The use of only publicly available data for this prediction makes this technique become a baseline for similar researches. This method has been applied in other countries [18], examined for other human development indicators [17], and has been explored using different sources of satellite images [19]. However, when it is implemented in other countries, the effectiveness of this approach depends on the tuning of algorithms used to derive features from satellite imagery. The ability of the model to generalize the different geographical contexts or other human development measures also becomes an issue and it needs additional testing before claims about global generalizability can be made. These studies show that there is compelling potential for adapting machine learning to describe poverty as a step forward in tackling the problem. Moreover, we can use the much more detailed economic data available to explore that potential and refine the ways that satellite data are used.

2.3.2 Implementation of Convolutional Neural Network on Satellite Imagery

The application of machine learning based methods continues to grow and expand in the geosciences and remote sensing area. They are increasingly used for interpreting the remote sensing images. Prior studies explored those images for solving problems in the field of environment, urban development, and demography. The machine learning techniques such as artificial neural network, support vector machines, decision trees, etc. become powerful approaches to handle this type of big data. They have notable effects both on the predictive analytics and the classification purpose [43].

Due to the high variability inherent in satellite images, most of the traditional supervised learning models are not suitable for handling satellite datasets. DeepSat was introduced as a classification framework for satellite imagery. It combines the deep belief network and neural network to extract features from images and use the normalized feature vectors to classify the images into barren land, none, grasslands, and trees. It produces an accuracy of 97.95% and 93.9% on the SAT-4 and SAT-6 datasets [44]. SatCNN was then presented as an agile CNN architecture to perform the same task. It uses deeper convolutional layers and smaller kernels to build an effective architecture. It can be trained relatively fast, achieving overall accuracies of 99.65% and 99.54%, which is the state-of-the-art for SAT datasets [45]. The CNN using a two-stage framework (ImageNet pre-trained model and trainable CNN) was also studied. The performance of the method using pre-trained network to classify aerial images into a set of diverse land-use classes can surpass the result of the UCML benchmark, improving the accuracy from 83.1% up to 92.4% [46].

In addition to the task above, deriving value and insights from this kind of data is also a big challenge. More knowledge can be extracted by analyzing each pixel in the images. Instead of labeling the images, an approach was proposed for per-pixel classification in satellite images using SegNet (encoder-decoder architectures). This approach introduced a multi-kernel convolutional layer that performs convolutions with several filter sizes to aggregate multi-scale predictions. Basically, it is equivalent to averaging an ensemble of multiple models sharing weights. It improves the accuracy of the models in the ISPRS 2D Vaihingen semantic labeling challenge by 1%. This technique was then combined with data fusion with a dual-stream architecture to refine the prediction. The combination improves the previous result with the multi-kernel technique from 89.4% to 89.8% [47]. Another approach used a pre-trained CNN model with labeled pixels created using simple linear iterative clustering (SLIC). The model was then fine-tuned for the task of per-pixel classification. The model implemented single CNN and multiple CNNs on the multispectral orthography images. The combination of four CNNs achieves the best classification accuracy of 94.49% [48]. Recent work adapted deep convolutional neural network for semantic segmentation (pixel-wise classification) on multispectral imagery RIT-18. Almost the same as the previous work, it creates synthetic images as the ground truth data by manual labeling the pixels using software DIRSIG. Moreover, the experiments also use several pre-trained models. The results showed that the synthetic imagery could be used to assist the training of end-to-end semantic segmentation model when there is not enough annotated image data [49].

Chapter 3

Methods

In this research, we use several types of data. This chapter attempts to provide the general information about the data, how we obtain and pre-process the data. Moreover, we present the overview of approaches that will be used to answer the research questions defined in Chapter 1.

3.1 Data Description

3.1.1 Poverty Data

We conduct the experiments using the data from Indonesia. The poverty dataset is publicly available, and it can be downloaded from Statistics Indonesia website [50]. It contains the poverty line measurement of municipalities in Indonesia, and it will be used as the ground truth for the models (Y_{true}) to estimate the poverty of a region. The unit value of a poverty line is the country's currency per person per month.

We use the latest poverty data from 2017. Besides, to test the model with a higher level of administrative unit, we use the poverty data on the provincial level. We also evaluate the out-of-country generalization of our model using the poverty data from Sri Lanka and Thailand. Since both countries use different currencies than the training dataset, we first converted the poverty value using the Purchasing Power Parity (PPP) metric. It can be defined as the number of units of the local currency required to buy the same amounts of goods and services in the domestic market as the US Dollar would buy in the United States. As mentioned in the previous chapter, the calculation of poverty line depends on the price of goods and services in the country. We use the PPP rate constructed by the World Bank to compare the economic well being between countries instead of exchange rate [51]. The tables below show the summary of poverty measurement in the training and testing datasets.

	Training	**Testing**
Number of municipalities	50	25
Poverty line (Rupiah/Person/Month)		
Minimum	211,485	200,663
Mean	377,023	353,428
Maximum	849,496	620,712
Log poverty line		
Minimum	12.26	12.21
Mean	12.81	12.74
Maximum	13.65	13.34

Table 3.1: The summary statistics of poverty data (municipality level)

	Testing
Number of provinces	25
Poverty line (Rupiah/Person/Month)	
Minimum	333,200
Mean	417,400
Maximum	587,500
Log poverty line	
Minimum	12.72
Mean	12.93
Maximum	13.28

Table 3.2: The summary statistics of poverty data (provincial level)

	Sri Lanka	**Thailand**
Number of regions	25	25
Poverty line	**(Rs/Person/Month)**	**(Baht/Person/Month)**
Minimum	4,181	2,053
Mean	4,437	2,361
Maximum	4,820	2,910
Log poverty line*		
Minimum	12.80	13.44
Mean	12.86	13.58
Maximum	12.94	13.79

*Converted to Indonesia currency value using the PPP conversion factor 2017 (Indonesia = 4190.49, Srilanka = 48.37, Thailand = 12.52) [52]

Source: National Statistical Office of Thailand, Department of Census and Statistics Sri Lanka

Table 3.3: The summary statistics of poverty data (out-of-country)

3.1.2 Daytime Satellite Imagery

The primary input of our models is satellite images provided by Google Satellite Maps. Although these images are regularly updated, we cannot predict when the maps will be updated. The update happens about once a month, but it can take months to process, verify, and set up the data before it is publicly accessible. We confirmed that the last update of Indonesia map is in 2018 based on the date stamp marking on the map. Since we use the poverty data 2017, we assume there is no significant temporal difference in the landscape features within a year.

Daytime satellite imagery is extracted from Google Static Maps API. By providing an API key, we are able to generate the high-resolution images given the geolocation information and the zoom level. The geolocation consists of latitude and longitude value indicating a location of a place in the real world. Meanwhile, the zoom level of Google Maps ranges from 0 to 19 describing the map scale. Google Maps is built on a 256 × 256 pixel tile system where zoom level 0 is a 256 × 256 pixel image of the whole earth. A 256 × 256 tile for zoom level 1 enlarges a 128 × 128 pixel region from zoom level 0 [53].

To provide inputs for the API, we generated random samples of coordinate points in each municipality in Indonesia. Figure 3.1 illustrates the geographic coordinate samples in a municipality, and the line in the figure represents the border of the municipality.

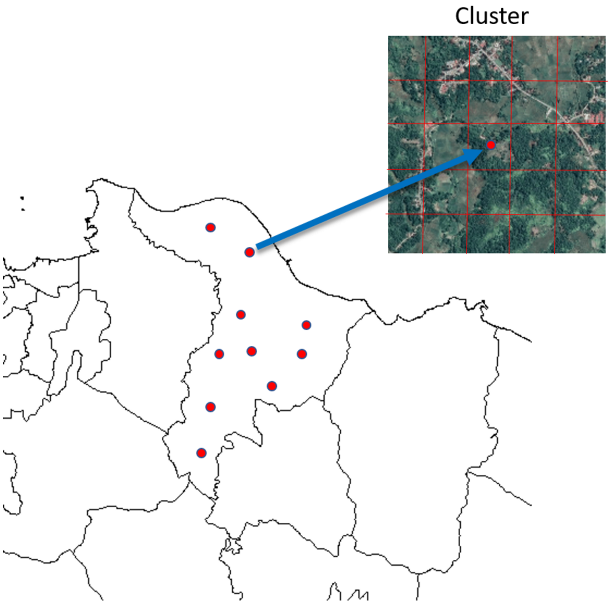


Figure 3.1: The sample coordinates in a municipality area

As illustrated in the figure above, we generated 10 random points for each region. Each point acts as the center of a cluster, and 25 satellite images were extracted from each cluster. Hence, each poverty data point is represented by 250 images. We set the resolution of the images 400 × 400 same as the size used by the reference study. We use the zoom level 16 (1 pixel = 2.387 meter). It means that each image covers ∼1 km in width and height. In the end, we have 12,500 images for training dataset and 6,250 images each testing dataset.

3.1.3 Nighttime Lights Image

The nighttime lights images were recorded by the Defense Meteorological Satellite Program (DMSP) in US Air Force Weather Agency. The image and data processing were performed by the NOAA National Centers for Environmental Information (NCEI). The data are cloud-free composites made using all the available archived DMSP-OLS (Operational Linescan System) smooth resolution data for calendar years. The products are 30 arc second grids, spanning -180 to 180 degrees longitude and -65 to 75 degrees latitude. The Version 4 DMSP-OLS Nighttime Lights Time Series can be downloaded for free in GeoTIFF format on NOAA website [54]. With this format, we can extract images based on the latitude and longitude of a location in this world. Even though the new data are added annually, the latest update is in 2013. We use this version of nighttime lights image. Because the year of data differs from daytime images, we assume that the luminosity in the nighttime images does not significantly change during those different time periods.

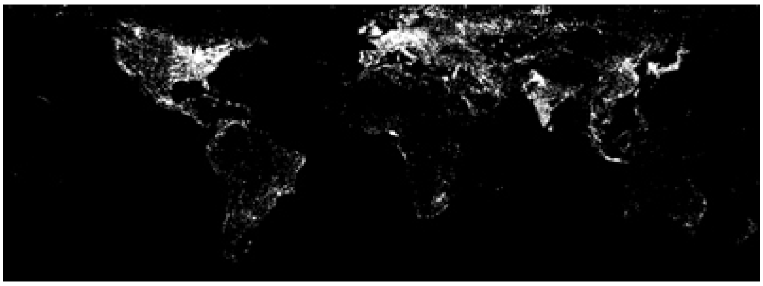


Figure 3.2: The DMSP-OLS Nighttime Lights

The downloaded nighttime lights image has resolution 16,801 × 43,201 for the whole world. One pixel in the image covers 1 km^2 in the real world. Therefore, each daytime image will correspond to a single pixel from the nighttime image. The pixel values are integers ranging from 0 to 63 representing the level of nighttime lights intensity in a 1 km^2 area. Figure 3.3 illustrates the nighttime image corresponding to one cluster in the daytime image dataset. Table 3.4 summarizes the mean value of night lights intensity per municipality (10 clusters) in the training and testing dataset.

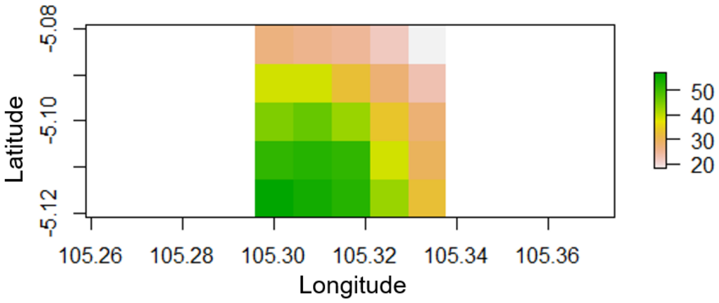


Figure 3.3: The nighttime image of one cluster. A pixel (square) corresponds to one daytime image.

	Training	Testing
Mean of Luminosity/Municipality		
Minimum	0.6	0
Mean	16.624	10.556
Maximum	61.524	63

Table 3.4: The summary statistics of the night lights intensity per municipality

3.2 Methods

We experiment with and evaluate different combinations of dataset and approach to estimate poverty. In this section, we introduce and discuss the methods to be applied. We use the study by Jean et al. as our baseline method [16]. We then present alternative techniques which also use a CNN model to address our research questions. The first approach is the baseline method to be used as a comparison with other approaches. The last two describe our proposed methods for predicting poverty.

3.2.1 Multistep Learning Approach (Baseline)

The multistep learning method introduced by Jean et al. has been used as the baseline model for similar works [16]. For our experiment, we also replicate this method using our dataset as a comparison for the proposed approaches. The method involves three main steps. First, we employ a VGG-F CNN model that has been pre-trained on ImageNet, a large image classification dataset with 1000 different categories. Second, we fine-tune that model on a new task to estimate the nighttime lights intensity corresponding to input daytime satellite imagery. We treat this step as a classification problem to predict the lights intensity classes.

Firstly we created classes that are determined by observing the histogram of nighttime lights intensities in our training set.

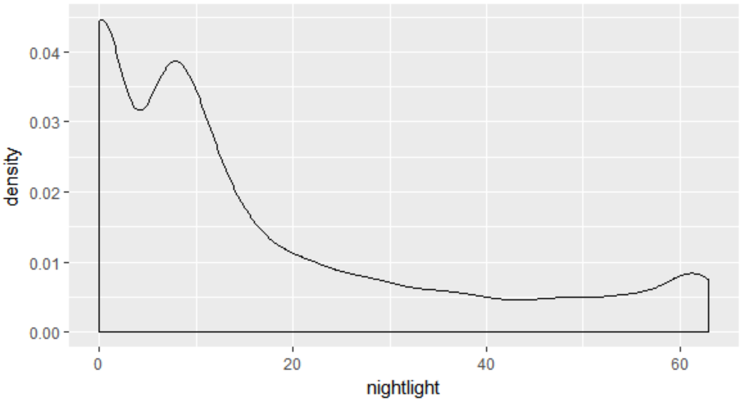


Figure 3.4: The distribution of lights intensity (nighttime lights image) in the training set

As can be seen from the density plot above, there is a mixture of three Gaussian distributions to the frequencies of the nighttime lights intensity values. Hence, by considering the balance of classes, we divided the interval 0-63 into three classes: a low class corresponding to near 0 nighttime lights intensity, a medium class corresponding to 6-16 range, and a high class corresponding to 17-63.

Pixel Value	**Class**	**Number of Daytime Images**	
		Training	**Testing**
0 - 5	0 (low)	3,890	3,181
6 - 16	1 (medium)	4,420	1,829
17 - 63	2 (high)	4,190	1,240
Total		12,500	6,250

Table 3.5: The number of cases in each lights intensity class

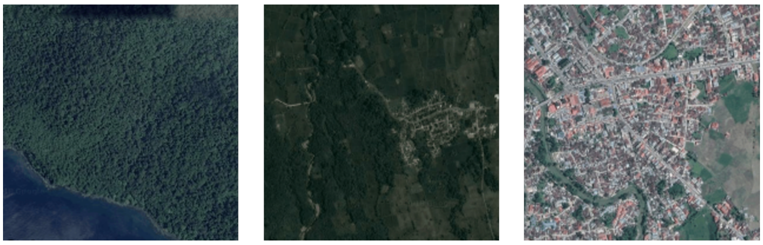


Figure 3.5: The example of daytime satellite images that correspond to a low class (*left*), a medium class (*middle*), and a high class (*right*)

Finally, a regression model with ridge regularization will be trained to learn the poverty rate of a region from the image features extracted by the CNN model stored in its last convolutional layer. The expectation is that the high-level features would probably capture more information relevant to poverty than low-level features.

3.2.2 Naive Approach

For the second method, we implement a naive approach to estimate the poverty rate. It is a straightforward approach by using the satellite images as the input of the CNN model to learn a regression task. To predict a continuous poverty value, we include a regression layer at the end of the network. As explained previously in the Literature Review chapter, the middle layers of the CNN network are the core CNN architecture, where most of the learning and computation take place same as the baseline approach. The final layers define the size and type of output data. In this regression task, we only have one output that represents the predicted poverty rate.

Although extensive research has been carried out on the use of CNN for this purpose, no single study exists which implement this direct approach. This approach aims to assess to which extent the simple regression convolutional network is able to predict poverty. Since we do not use a large training dataset, in this naive approach, we utilize the existing model which have been previously trained on datasets from other domains (*transfer learning*). Same as the baseline approach, we employ the model pre-trained on the ImageNet dataset, then further fine-tune it on our image dataset. Prior studies used the ImageNet pre-trained model to help increase the accuracy of the CNN model to classify the satellite images [46], [55]. The pre-trained model acts as a starting point to perform feature extraction in our complete network. We maintain the weights of the pre-trained model since they capture general features like curves and edges that might be relevant to our problem. Then, we refine the weights by learning dataset-specific features from our training images. Furthermore, we customize the last layer to output the estimation of poverty.

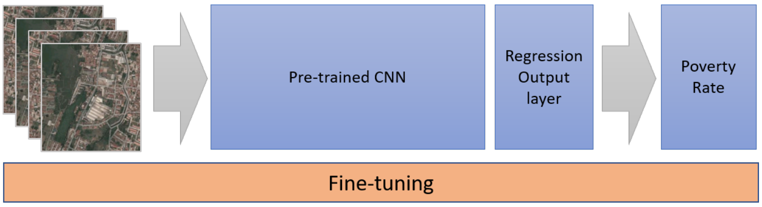


Figure 3.6: The naive approach

3.2.3 Semantic Segmentation Approach

The second proposed approach identifies the features in the daytime satellite images by performing semantic segmentation. This approach consists of two steps: performing semantic segmentation to derive the landscape features from the training images and building a regression model to estimate the poverty from the extracted features.

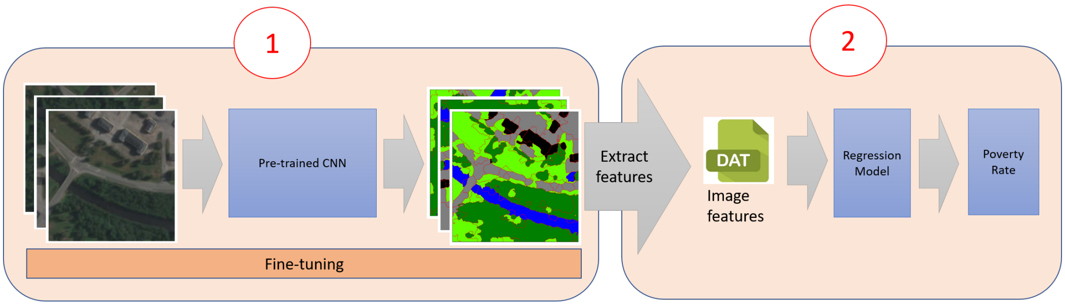


Figure 3.7: The semantic segmentation approach. (1) CNN model to perform semantic segmentation, (2) regression model to estimate the poverty rate from the extracted image features.

The semantic segmentation is referred to as per-pixel classification. It predicts the label for every pixel so that each pixel is labeled with the class of its enclosing object or region [56]. We categorize the features in an image into six categories: vegetation, ground, road, buildings, water, and miscellaneous structure. The classes are similar to those used in a study on classification and segmentation of satellite orthoimagery. That study classifies the geographical features into five categories: vegetation, ground, road, building, and water [48]. We include a new class, miscellaneous structure, to capture landscape structures that cannot be categorized into five other classes such as clouds or noises that are captured in the satellite imagery.

To begin the first step, same as the first approach we use a CNN model pre-trained on ImageNet dataset. The weights of the pre-trained model are then adapted and refined to the segmentation task using our training images. To create the training dataset, we took 20 daytime satellite images and created the ground truth by manually labeling the pixels using the semantic segmentation editor [57]. We converted the ground truth images to be single channel images in which each pixel is labeled with its class. We then performed image augmentation to increase our dataset size. Table 3.6 presents the augmentation techniques used to generate the new images. For the testing dataset, we randomly took 10 daytime satellite images and created the ground truth of those images using the same tool.

Dataset	**Number of Images**
Training	
Original	20
Image augmentation: (1) Horizontal flip (2) Rotate 90°, 180°, and 270° (3) Add random noise (4) Rescale the intensity (5) Gamma correction (6) Combination of (2) and (1) (7) Combination of (2) and (4) (8) Combination of (2) and (5)	320
Total	**340**
Testing	**10**

Table 3.6: The number of training and testing images for semantic segmentation task

The figure below illustrates how the area of an image is segmented manually into six labels. The first row of Figure 3.8 is the training images, and the second row is the finished labeled pixels. The labeled pixels in the images serve as the explanatory variables for estimating the poverty of a region.

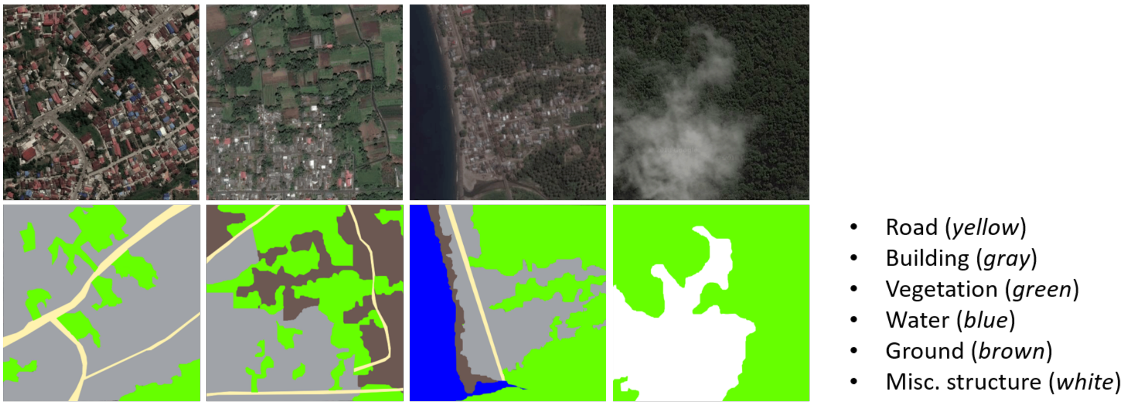


Figure 3.8: The illustration of satellite imagery segmentation

Following the segmentation step, a regression model is applied to learn the features (classes) outputted by the CNN model to estimate the poverty rate. We take the percentage of each feature in the images representing a region as the predictors. To get the best performing regression model, we apply different regularization techniques and evaluate the results.

Chapter 4

Experiments and Evaluation

The main objective of our experiments is to investigate whether we can use the satellite imagery to measure the poverty of a region. We will evaluate how well the approaches described in the previous chapter perform to meet this objective. For each approach, we built the model and measured its performance in determining which approach provides the best prediction.

4.1 Experimental Setup

We built CNN models with different configurations on a machine with 24 × Intel Xeon(R) CPU X5650 @ 2.67 GHz, Memory 94.4 GiB, and GPU NVidia GeForce GTX 1080/PCIe/SSE2. The system runs on 64-bit version of Ubuntu 16.04.4 LTS.

We employed Caffe library to build the CNN models. The pre-trained models such as VGG F, AlexNet, and ResNet used in the experiments were obtained from Caffe Model Zoo [58]. The tables below provide the information on the training and testing setups of our experiment. We used the training and testing datasets described in the previous chapter. We built the models using different types of training dataset depending on what is needed by each approach. We also created another regression model using the nighttime lights data. Furthermore, we performed generalization tests on the province and out-of-country data.

Model	Approach	Training Dataset
1	Multistep Learning	Daytime satellite images, nighttime lights image, and poverty data
2	Naive	Daytime satellite images and poverty data
3	Semantic Segmentation	Daytime satellite images and poverty data
4	Night lights model	Nighttime lights image and poverty data

Table 4.1: The training setup

Model	Testing Dataset	Section
1	Daytime satellite images, nighttime lights image, and poverty data	4.2.1
2, 3	Daytime satellite images and poverty data	4.2.2 and 4.2.3
4	Nighttime lights image and poverty data	4.2.4
1, 2, 3	Daytime satellite images and poverty data (provincial level test)	4.2.5
1, 2, 3	Daytime satellite images and poverty data (out-of-country test)	4.2.6

Table 4.2: The testing setup

4.2 Experiments and Results

4.2.1 Multistep Learning

The first component of our baseline approach is the pre-trained CNN model. We utilized VGG F convolutional network as illustrated in Figure 4.1. This model is a deep architecture introduced in 2013 and has been trained on ImageNet dataset to predict 1000 different classes with top-1 error 41.1% and top-5 error 18.8% on the ILSVRC2012 validation data [59]. We fine-tuned this model on our training dataset to predict the night lights intensity class of the daytime satellite image. The number of output in the last layer of the pre-trained model was replaced from 1000 to 3 because we only use three classes in our case study which are low, medium, and high.

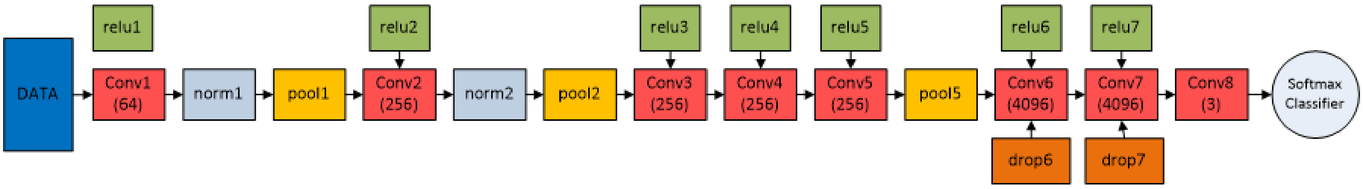


Figure 4.1: 8-layer CNN model (VGG F)

As presented in Table 3.5, we have 12,500 training images and 6,250 testing images of size 400 × 400 (RGB bands). To prevent overfitting, we divided the training set into training and validation sets by a ratio of 80% and 20% while maintaining the class balance to avoid bias in our results. Before feeding them to the convolutional neural network, we transformed the input images by subtracting the mean across every individual pixel in the data. Table 4.3 shows the configuration to build the CNN model.

CNN Configuration	
Maximum iteration	1,000,000
Learning rate: base / final	$1 \times 10^{-6}/1.25 \times 10^{-7}$
Batch size: training / validation	32 / 8
Step size	50,000
Gamma	0.5
Momentum	0.9
Weight decay	0.0005

Table 4.3: The CNN model configuration

From the learning curve in Figure 4.2, it is apparent that the loss in both training and validation set starts decreasing in the first few iterations, and it is more stable after 200,000 iterations. After performing optimization until 1 million iterations, we decided to take the snapshot model at the 200,000th iteration as our final model since there is no significant change in the loss after that iteration.

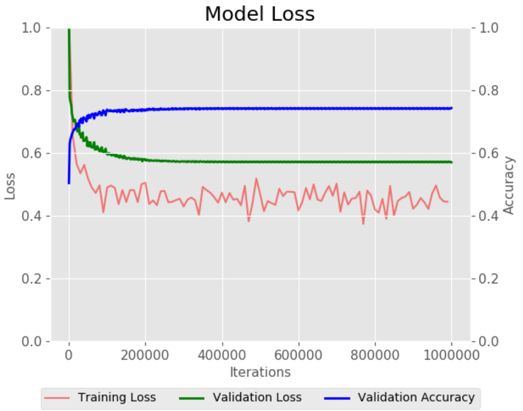


Figure 4.2: The learning plot

To evaluate the performance of the model, we used the constructed CNN model to predict the night lights intensity of the testing images. The confusion matrix is given in Table 4.4, and the evaluation metrics of our classifier are presented in Table 4.5. As shown in the matrix, there is still a confusion to distinguish the low and medium class. There is a number of low class images that are misclassified as the medium class, and vice versa. The same case also happens between medium and high classes, but the misclassification rate is smaller than the previous one. The model achieves an accuracy of 0.75 on the training set and 0.74 on the testing set. Furthermore, the precision and recall of the model do not differ much from the accuracy.

		Predicted		
		Low	Medium	High
True	Low	2,751	406	24
	Medium	611	1,010	208
	High	66	325	849

Table 4.4: The confusion matrix of the model over the test dataset

	Accuracy	**Precision**			**Recall**			**F1 score**		
		Micro	**Macro**	**Weighted**	**Micro**	**Macro**	**Weighted**	**Micro**	**Macro**	**Weighted**
Training	0.75	0.75	0.76	0.76	0.75	0.75	0.75	0.75	0.75	0.75
Testing	0.74	0.74	0.73	0.73	0.74	0.70	0.74	0.74	0.71	0.74

Table 4.5: The performance evaluation on the dataset

The CNN model was then used to extract image features from the satellite images that are useful for the poverty estimation task. It extracts 4096-dimensional feature vector in layer `conv7` from each input satellite image. Since we have 250 images representing a municipality, we then averaged those 250 extracted feature vectors to obtain one feature vector for each region, which was then used as input in a ridge regression model for predicting poverty. Furthermore, we applied *Principal Component Analysis* (PCA) to reduce the feature dimension. We can reduce the computational task and avoid the overfitting problem by using only the relevant features. We experimented with both full ($d = 4096$) and reduced features ($d = 100$). The result showed that the less complex model can retain the data variation since both inputs gave almost the same correlation value between input and output. The model used nested cross-validation to choose the best ridge regularization parameter (λ). The choice of regularization parameter for each fold is determined in an inner cross-validation loop to preserve the integrity of the hold-out test data. The reported R^2 is the average of R^2 across the cross-validation folds.

Algorithm 2: The regression model and configuration

```
$d = 100$ (dimension)
$\lambda_{low} = 1$ (log of smallest $\lambda$)
$\lambda_{high} = 5$ (log of largest $\lambda$)
$n = 10$ (number of regularization parameters to try between $\lambda_{low}$ and $\lambda_{high}$
foreach fold ($k$ = 10) do
    split the data into training and testing sets
    foreach inner fold ($inner\_k = 10$) do
        split the training set into training and validation sets
        get the best regularization parameter ($\lambda_{best}$) using training and validation sets
    end
    build a regression model using $\lambda_{best}$ and training set
    evaluate the model on testing set (compute $R_k^2$)
end
$R^2 \leftarrow$ average of $R_k^2$
return $R^2$
```

The results may differ slightly with each run due to randomly splitting data as described in Algorithm 2. Therefore, we ran the model 100 times, and the prediction plot presented in Figure 4.4 is the better average result obtained over 100 runs. Using the multistep learning, the image features can explain up to 45 percent of the variation of poverty. This number is similar to the result of applying the same technique in four African countries ranging from 37 to 55 percent [16]. Accordingly, we can say that we have successfully replicated the multistep learning approach for our dataset, and we can use it as the baseline for comparison.

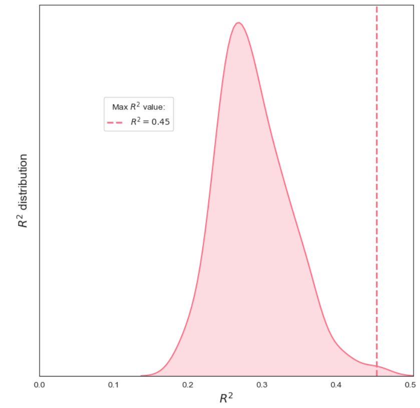


Figure 4.3: The distribution of R^2 over 100 runs

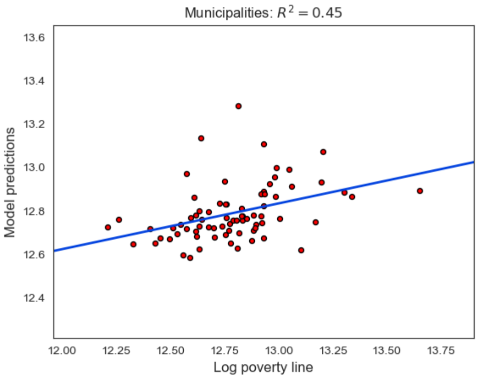


Figure 4.4: The prediction plot

In addition, we investigated whether a better classifier in the first step will produce a better predictor in the second step of this approach. During the construction of the CNN model, we built several models with various levels of performance provided in Table 4.6. For each model, we observed its R^2 distribution over 100 trials, and *Welch two sample t-test* was used to compare the difference between two models [60].

Models	**Accuracy**	**Precision**	**Recall**	**F1 score**
Comp1	0.72	0.73	0.72	0.72
Comp2	0.70	0.70	0.71	0.70
Comp3	0.55	0.60	0.55	0.56

Table 4.6: The evaluation metrics of compared models

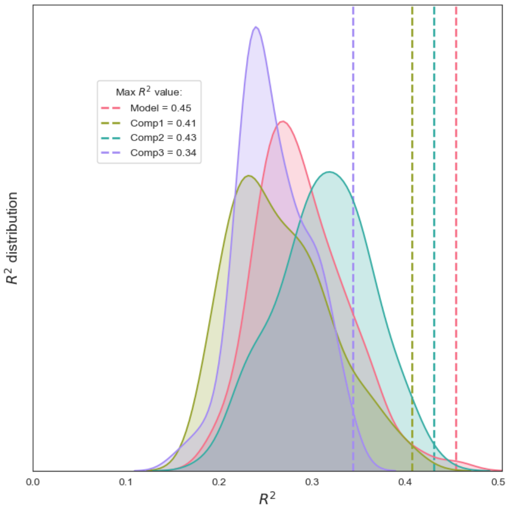


Figure 4.5: The comparison of R^2 distributions of CNN model with different accuracy values

The statistical tests revealed that the result of the regression model appears to be unaffected by the performance of the CNN model. There are several findings that can be pointed out from the table below. First, with confidence level 95%, our CNN model that has a slightly better performance than `Comp2` does not produce a bigger R^2 value than `Comp2`. Second, `Comp1` that has accuracy 0.72 cannot outperform the predictive power of features extracted by `Comp3` with accuracy 0.55. As is the case with `Comp2` and `Comp1`. Although the difference in accuracy is very small between two classifiers, `Comp2` predicts the poverty better than `Comp1`. Therefore, it seems that the performance on the classification task can be a rough indicator of the quality of the learned features, but it does not mean that the better performed CNN model will always produce a more accurate prediction of poverty.

Alternative hypothesis: true difference in means is greater than 0				
>	**Final Model**	**Comp1**	**Comp2**	**Comp3**
Final Model	-	0.0006461	0.9996	6.744e-07
Comp1	0.9994	-	1	0.1299
Comp2	0.000395	4.236e-10	-	1.641e-15
Comp3	1	0.8701	1	-

Table 4.7: The *p-value* of t-test between two models ($\alpha = 0.05$)

4.2.2 Naive

In this first proposed approach, we used the daytime satellite image features directly for estimating poverty. We examined if the low-level image features are more relevant and meaningful to the poverty than the high-level features derived in the multistep learning approach. We used a CNN model as the only step to predict the output. The CNN will recognize the low image features such as pixel intensity, pixel gradient orientation, and color from the satellite images and then directly use those raw features to predict the poverty. It is different from the baseline approach which utilizes the CNN to learn and combine that kind of features to produce high-level representations of the image and then uses them as the input of the regression model.

Initially, we experimented with AlexNet, VGG16, and ResNet25 architectures. However, for the last two architectures, the model cannot be built because the process was too big to fit in the GPU memory. Because of that, we used shallower models of VGG and ResNet which are VGG F and ResNet10. In this approach, we also utilized pre-trained CNN models to improve the performance of the resulted models.

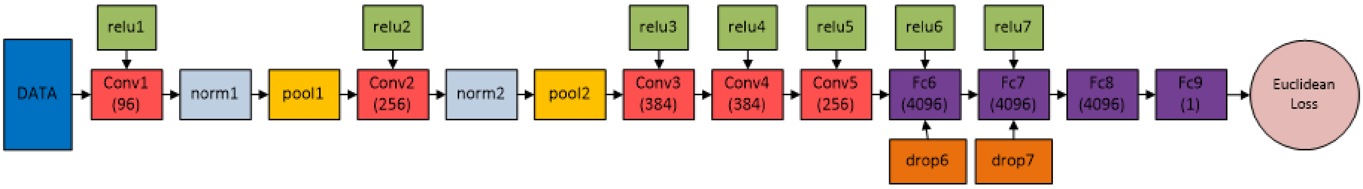


Figure 4.6: AlexNet architecture

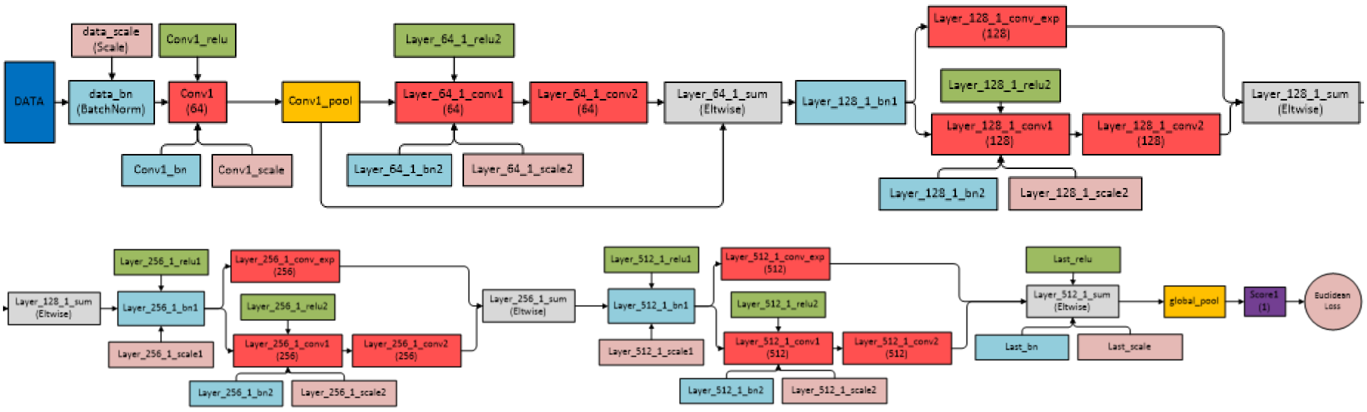


Figure 4.7: ResNet10 architecture

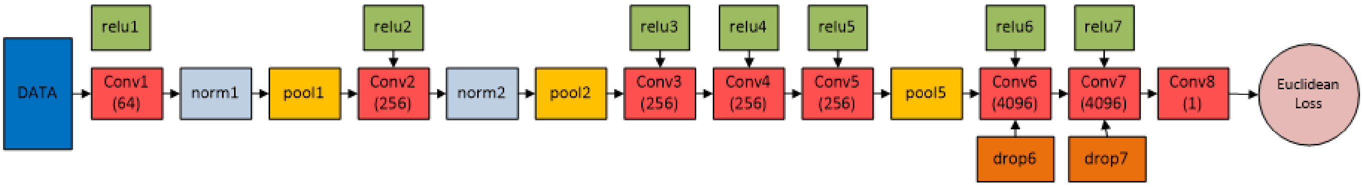


Figure 4.8: VGG F architecture

The figures above illustrate the architecture of AlexNet, ResNet10, and VGG F. The depth of AlexNet and VGG F is quite similar. Unlike AlexNet, VGG F does not include any fully connected layer in the model. On the other hand, ResNet10 includes several layers that sum two convolution layers element-wise. It also uses batch normalization instead of Local Response Normalization (LRN) as applied in

AlexNet and VGG F. We customized the final layer of all CNN models by changing the number of output to be 1. There is only one type of loss layer for regression task in Caffe library which is Euclidean Loss (E). Therefore, we used it to compute the loss of the regression task in the learning process. The loss can be written as

$$E = \frac{1}{2N} \sum_{n=1}^{N} \|\hat{y}_n - y_n\|^2 = \frac{1}{2} MSE \tag{4.1}$$

where MSE is the mean of the squares of the loss $(\hat{y}_n - y_n)^2$. The model hyperparameters are provided in Table 4.8. Prior to training the models, we performed log transformation on the poverty value to make the loss pattern more visible. The optimization was performed until 200,000 iterations. By considering the available resource, we can only use the maximum batch size of 32 for the training set and 10 for the validation set. As shown in Figure 4.9, the three models have the same loss curve even though the exact loss value for each model is different. The ResNet10 has the smallest loss among the three models. We can see from the learning plots that there is a clear trend of decreasing in the training and validation loss, and they remain constant after approximately 10,000 iterations. In all CNN models, we took the snapshot model at the 50,000th iteration to predict poverty.

CNN Configuration	
Maximum iteration	200,000
Learning rate	1×10^{-6}
Batch size: training / validation	32 / 10
Step size	50,000
Gamma	0.5
Momentum	0.9
Weight decay	0.0005

Table 4.8: The CNN regression model configuration

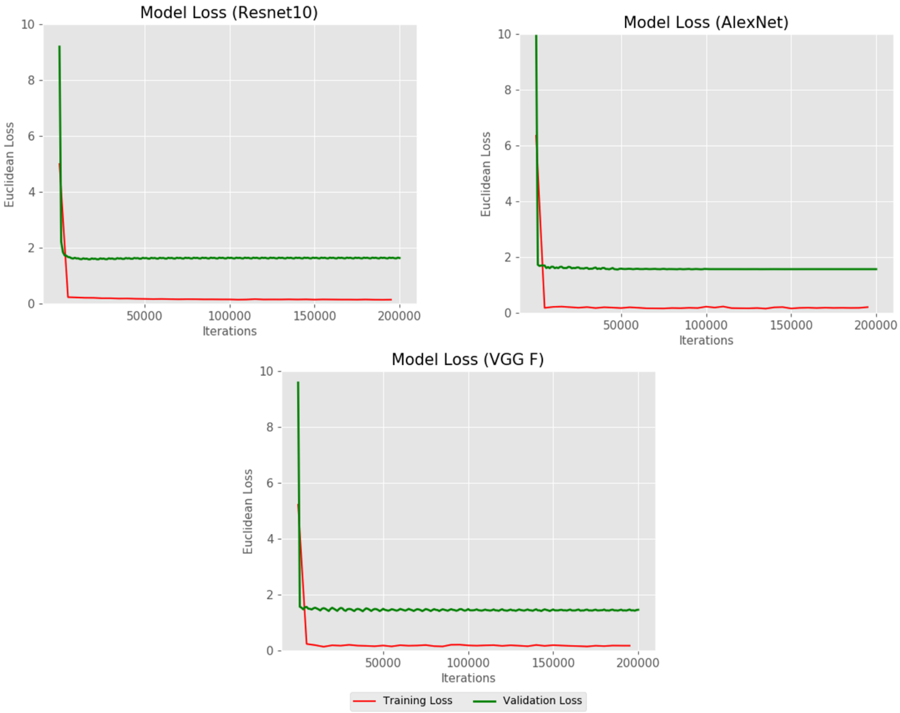
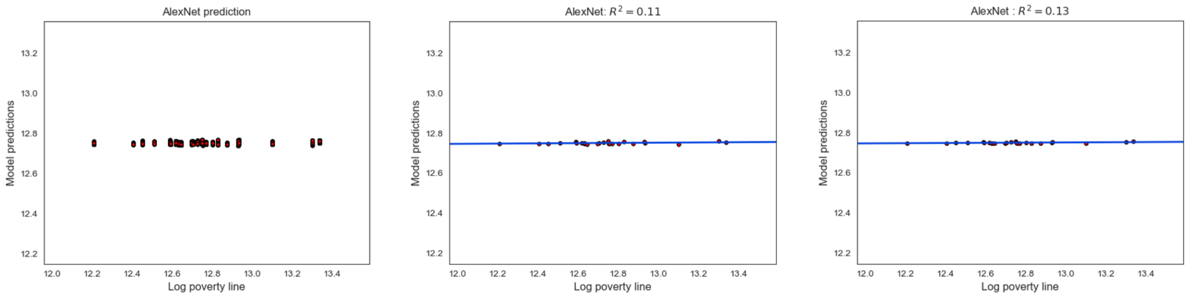


Figure 4.9: The learning plot of naive approach

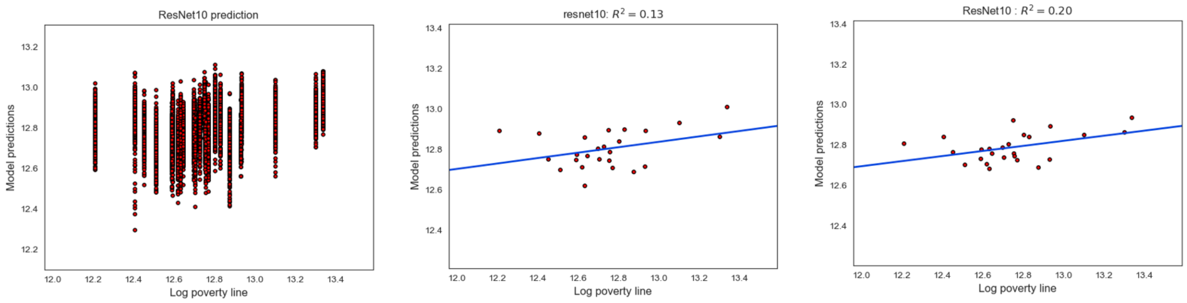
We evaluated the CNN models on the testing dataset. We have 250 images for each region, and each of them will have the same true poverty value. The satellite images from 25 municipalities were then fed into the model directly to obtain the estimation of poverty. The results obtained from AlexNet, ResNet10, and VGG F are presented on the left side of the figures below. Comparing the three results, it can be seen that AlexNet predicts almost the same value for all the input images. VGG F has slightly better predictions since it produces more diverse poverty values for different regions. In contrast, the variance of ResNet10 outputs is the highest among the three models. To compute the performance of the model, we applied majority voting and averaging strategy. Using majority voting strategy, we took the majority vote from 250 predictions for each region. Using averaging strategy, we calculated the mean value of the multi-predictions as the estimated poverty of each region. We visualized the results in the scatter plot on the right side of Figure 4.10, 4.11, and 4.12. The R^2 for AlexNet, ResNet10, and VGG F using majority voting are 0.11, 0.13, and 0.11, respectively. By applying averaging strategy, the results are slightly better than majority voting. The resulted R^2 are 0.13, 0.20, and 0.14 for AlexNet, ResNet10, and VGG F, respectively. By far, AlexNet has the poorest performance. Moreover, the difference between AlexNet and VGG F is very small, and both predict the poverty in almost uniform values for all testing regions. ResNet10 performs better than the others, although it cannot outperform the result of the previous approach. From these results, it can be seen that the depth of the network is crucial in the CNN architectures. With the same setting of hyperparameters, ResNet10 having deeper layers than the plain networks such as AlexNet and VGG F results in better performance.



(Majority Voting)

(Averaging)

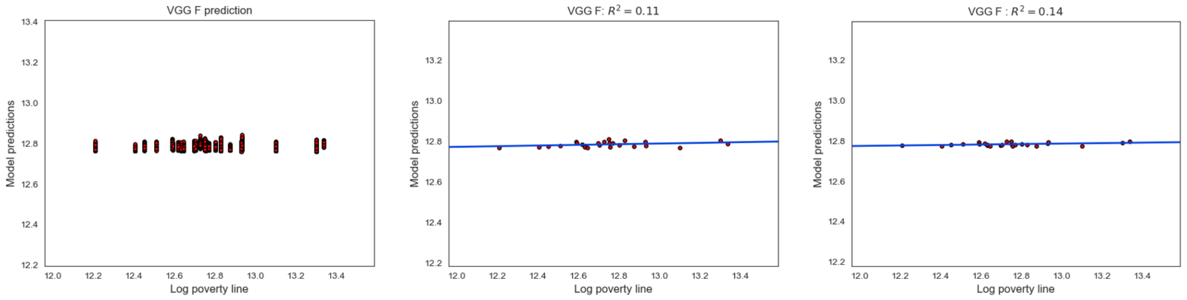
Figure 4.10: AlexNet prediction result



(Majority Voting)

(Averaging)

Figure 4.11: ResNet10 prediction result



(Majority Voting)

(Averaging)

Figure 4.12: VGG F prediction result

Overall, the regression models in the naive approach do not produce better performance than the baseline approach. The high-level features extracted from the multistep learning are more relevant than the low-level features of satellite images to predict poverty of a region.

4.2.3 Semantic Segmentation

In the second proposed approach, firstly we implemented SegNet model for semantic pixel-wise segmentation [61]. The model consists of an encoder network, a corresponding decoder network followed by a pixel-wise classification layer. The encoder comprises convolutional layers with batch normalization and a ReLU non-linearity followed by non-overlapping max-pooling. The decoder maps the low-resolution feature maps resulted by the encoder to full input resolution feature maps. It uses pooling indices computed in the max-pooling step of the corresponding encoder to perform non-linear upsampling. The sparse upsampled maps are then convolved with trainable filters to produce dense feature maps.

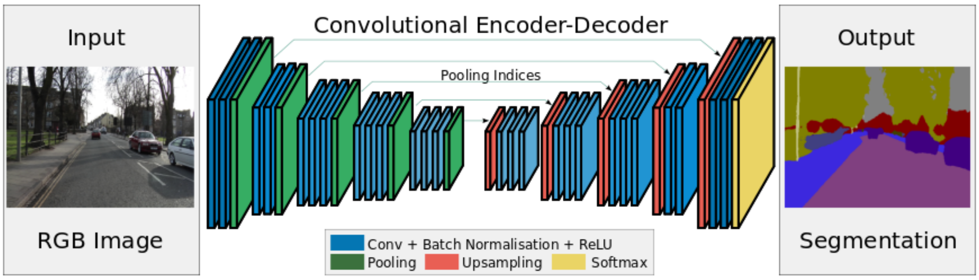


Figure 4.13: The encoder-decoder architecture of SegNet [61]

We experimented using SegNet and SegNet-Basic architectures. The encoder in SegNet is topologically identical to the convolutional layers in VGG16. SegNet uses the first 13 convolution layers in the VGG16 and removes the fully connected layers of VGG16. Each encoder layer in the architecture has a corresponding decoder layer, and hence the decoder network also has 13 layers. The final decoder output is fed to a multi-class Softmax classifier to produce class probabilities for each pixel independently. Meanwhile, SegNet-Basic is smaller version of SegNet that has four convolution layers in both encoder and decoder layers. In [61], it is shown that a reasonably good performance up to 82 percent of global accuracy is achieved by SegNet-Basic. We investigated whether using the basic version is enough to do the task rather than using network with lots of layers that require more computational resources. In both SegNet and SegNet-Basic, batch normalization is used after each convolutional layer in the encoder and decoder networks. ReLu is only used in the encoder, and it is not present in the decoder network.

We trained the models using 340 daytime satellite images (320 augmented images and 20 original images) presented in Table 3.6 to segment the 400 × 400 pixels in the images into six classes: road, building, vegetation, water, ground, and miscellaneous structure. For the SegNet network, we initialized the weights of the encoder using the VGG16 model pre-trained on ImageNet dataset. Table 4.10 below provides the configuration of the CNN models. Furthermore, since there is a large variation of the number of pixels for each class in the training images, we used *median frequency balancing* to calculate the weight assigned to each class in the Softmax layer. By doing so, we weighted the loss differently based on the true label of the pixel.

Class	Road	Building	Vegetation	Water	Ground	Misc.
Frequency	1,665,082	11,929,291	25,721,102	2,791,434	7,606,140	4,686,951
Weight	3.691437	0.515248	0.238969	2.201931	0.808103	1.311417

Table 4.9: The weight assigned to each class that is calculated using median frequency balancing

CNN Configuration	
Maximum iteration	40,000
Learning rate: : SegNet / SegNet-Basic	0.001 / 0.1
Batch size: SegNet / SegNet-Basic	2 / 4
Step size	100,000
Gamma	1
Momentum	0.9
Weight decay	0.0005

Table 4.10: The SegNet and SegNet-Basic model configuration

Due to the limited capability of the GPU, we only used the batch size of 2 for SegNet and 4 for SegNet-Basic. It is a relatively small number for training a CNN model. However, the previous study also experimented with a mini batch size of 5 to classify the pixels in the road scenes (11 classes) and a batch size of 4 for indoor scenes (37 classes), and even the batch size of as low as 2 or 3 still trains well [61]. Figure 4.14 plots the cross-entropy loss in each training iteration for SegNet-Basic and SegNet. As shown in the plot, SegNet learns better than the SegNet-Basic. The loss produced by SegNet both in training and validation decreases, and it converges after 30,000 iterations. Meanwhile, SegNet-Basic has higher loss than SegNet. It is expected since SegNet has more layers to help extract and recognize more features in our images which improves its performance. We selected the SegNet and SegNet-Basic snapshots after 40,000 iterations of optimization as our models.

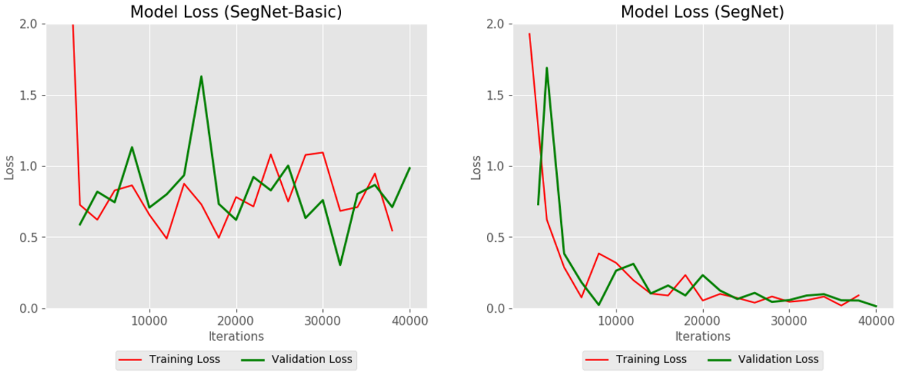


Figure 4.14: The learning plot of semantic segmentation approach

We evaluated the performance of the pixel-wise classifiers on the testing dataset that consists of 10 images with resolution 400 × 400 (1,600,000 pixels). The confusion matrices for both models are presented in Tables 4.11 and 4.12. It is not surprising that SegNet performs better that SegNet-Basic in this experiment since we already know from the learning plots that SegNet-Basic has a higher cross-entropy loss than SegNet. By observing the confusion matrix in Table 4.11, it can be seen that SegNet-Basic cannot distinguish the image feature well enough. All the predicted features are classified mostly as vegetation. The applied class balancing cannot remedy the lack of modeling power of the SegNet-Basic. It still biases towards the majority class which is vegetation. Furthermore, the model is confused to classify road, building, and ground. For SegNet, we can see that this model performs well to identify the image features. Most of the pixels in the testing images are classified correctly. However, if we inspect the matrix in Table 4.12 closer, the model fails to identify the miscellaneous image feature such as the clouds or shadows, and it is mostly recognized as vegetation. It is quite interesting since most of the miscellaneous features in the training images have the contrasting RGB color (white) than other features. It could be because clouds or shadows appear above the other feature pixels so that the color representing the miscellaneous feature is not very clear and contains the characteristics from other features.

		Predicted					
		Road	Building	Vegetation	Water	Ground	Misc.
True	Road	0	0	40,463	0	87	0
	Building	0	0	429,232	0	3,565	0
	Vegetation	0	0	849,236	0	0	0
	Water	0	0	91,653	0	0	0
	Ground	0	0	158,511	0	7,544	0
	Misc.	0	0	19,709	0	0	0

Table 4.11: The confusion matrix of SegNet-Basic (testing set)

		Predicted					
		Road	Building	Vegetation	Water	Ground	Misc.
True	Road	25,436	5,898	4,895	504	3,817	0
	Building	34,488	317,935	44,900	54	35,420	0
	Vegetation	12,621	40,900	738,316	22,409	34,990	0
	Water	2,130	16	11,011	67,286	11,210	0
	Ground	10,427	20,249	26,808	1,395	107,176	0
	Misc.	3,854	449	15,406	0	0	0

Table 4.12: The confusion matrix of SegNet (testing set)

For each model, we computed the global accuracy, precision, recall, and F1 score based on the confusion matrices. In Table 4.13, we reported the numerical results of our analysis on both the training and testing datasets. By examining the performance metrics on both datasets, we can see if the models are overfitted or not. As shown in the table, the models perform similarly in both datasets, and it indicates that our models generalize well to our testing dataset. Moreover, we can also see that SegNet outperforms SegNet-Basic in all the performance metrics. Since the number of cases in each class varies, precision and recall computed from the micro-average method is more relevant in these results. Overall, the results suggest that the use of more convolution layers and pre-trained model leads to better performance. Besides, we also presented the comparison of the input and the predicted image using SegNet in Figure 4.15.

Model	Accuracy	Precision			Recall			F1 score		
		Micro	Macro	Weighted	Micro	Macro	Weighted	Micro	Macro	Weighted
Training										
SegNet-Basic	0.48	0.48	0.16	0.29	0.48	0.17	0.48	0.48	0.12	0.31
SegNet	0.81	0.81	0.58	0.78	0.81	0.75	0.81	0.81	0.64	0.78
Testing										
SegNet-Basic	0.54	0.54	0.20	0.35	0.54	0.17	0.54	0.54	0.13	0.38
SegNet	0.79	0.79	0.55	0.80	0.79	0.60	0.79	0.79	0.56	0.79

Table 4.13: The performance evaluation of the models on the dataset

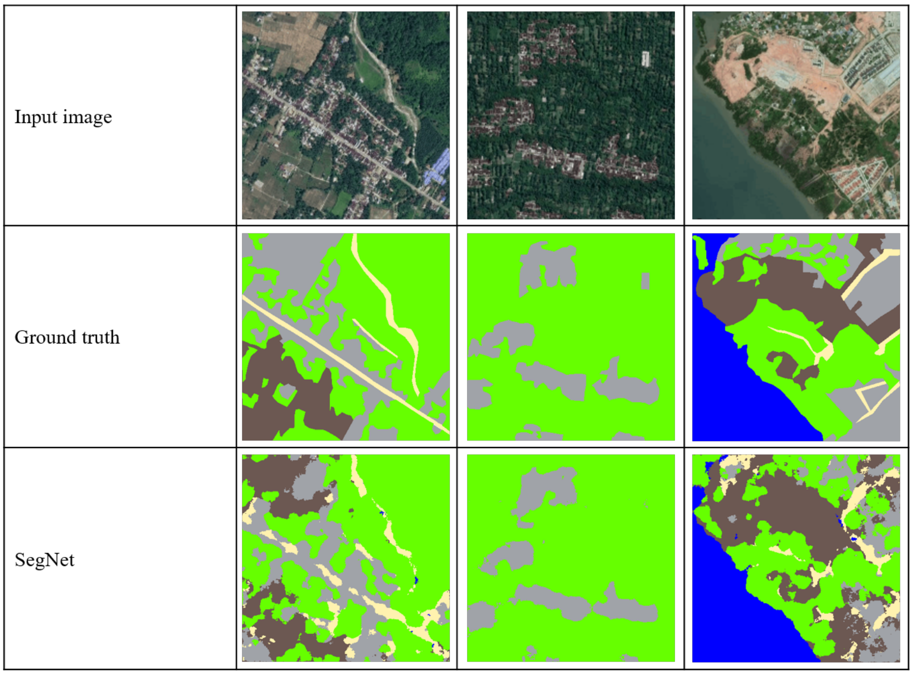


Figure 4.15: The comparison of the input, ground truth, and predicted image using SegNet model (from testing dataset)

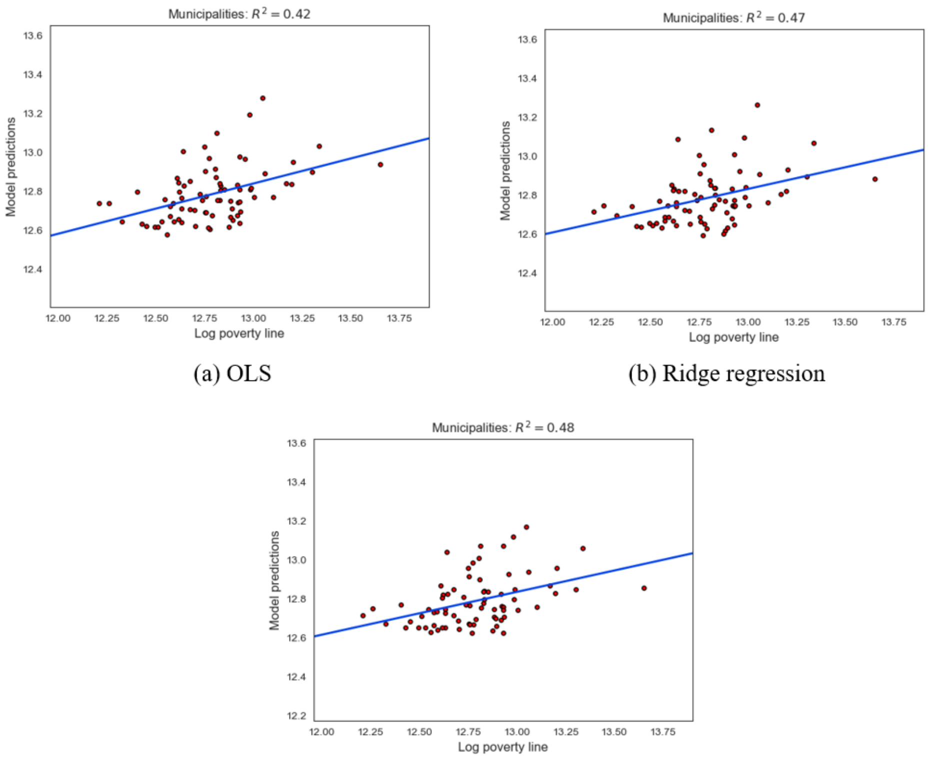
In the second step of this approach, we built a regression model to predict the poverty value of a region based on the image features identified by the SegNet model in the first step. For each region, we aggregated the number of pixels of each landscape feature from 250 images representing that region and calculated the percentage of each feature. We used the values as our predictor variables: % of road area, % of building area, % of vegetation area, % of water-filled area, and % of ground area. We did not include the last feature because there is no pixel predicted as miscellaneous landscape structure from our dataset.

We experimented with a simple linear regression (OLS) as well as regularized regression models: Lasso and Ridge regression. To make it comparable to the baseline approach, we used the same method to calculate the R^2 (Algorithm 2). Table 4.14 shows the model setting to obtain the results presented in Figure 4.16.

Parameters	OLS	Ridge	Lasso
Log of smallest λ	-	0	0.00001
Log of largest λ	-	3	1
Number of regularization parameters λ to try (n)	-	5	100
Number of fold (k)	10	10	10
Number of inner fold $(inner\_k)$	10	10	10

Table 4.14: The setting of the regression models for Algorithm 2

The three regression models generate the predictions of poverty. The R^2 values for OLS, Ridge, and Lasso are 0.42, 0.47, and 0.48, respectively. In other words, a regression model with five image features as the predictors can explain 42 to 48 percent of the variation of poverty in a region. By performing regularization, the model can achieve better performance. It is apparent that our independent variables are highly correlated. To put it another way, the data suffer from multicollinearity. An increase or decrease in the percentage of an image feature will affect the value of other independent variables. Hence, OLS will give a less accurate prediction of poverty. By adding a degree of bias to the estimation, Ridge regression overcomes this problem and improves the prediction of the OLS by five percent. Furthermore, the results of Ridge and Lasso are not much different in this experiment.



(a) OLS

(b) Ridge regression

(c) Lasso regression

Figure 4.16: The prediction results of OLS, Ridge, and Lasso regression

A comparison of the three approaches reveals that the semantic segmentation approach gives the better performance than the other two approaches. This approach improves the performance of the multistep learning approach by 3 percent using Lasso regularization.

4.2.4 Additional Model using Nighttime Lights Data

In the baseline approach, we used the night lights data as a proxy to predict poverty. Our proposed methods use only the daytime image features as the input. Therefore, we also built additional models to test whether the model using the nighttime images alone will perform well to predict poverty compared to the model using the daytime images. We experimented with two regression models, namely a model with the night lights intensity as the only predictor and a model that also uses the semantic image features other than the night lights data.

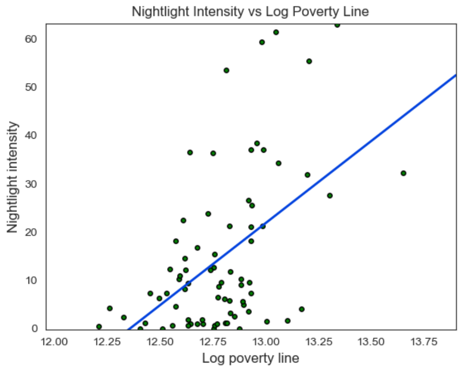


Figure 4.17: The scatter plot of the night lights intensity VS log poverty line

First, we built a simple regression model (OLS) using the mean of luminosity in Table 3.4 as the predictor. Using the same algorithm as previous models (Algorithm 2), we created the OLS model with 10 fold (k) and 10 inner folds ($inner\_k$). The best reported cross-validated R^2 from the model is 0.44 as can be seen in Figure 4.18. This result is almost similar to the performance of the model from the baseline approach which is 0.45. It even outperforms the result of the naive approach which is 0.20.

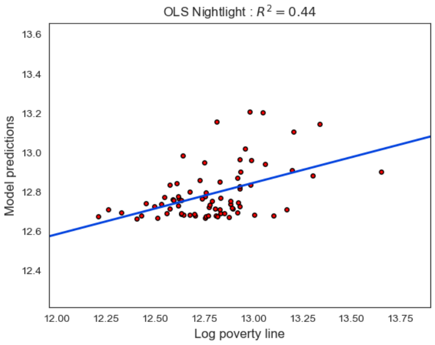
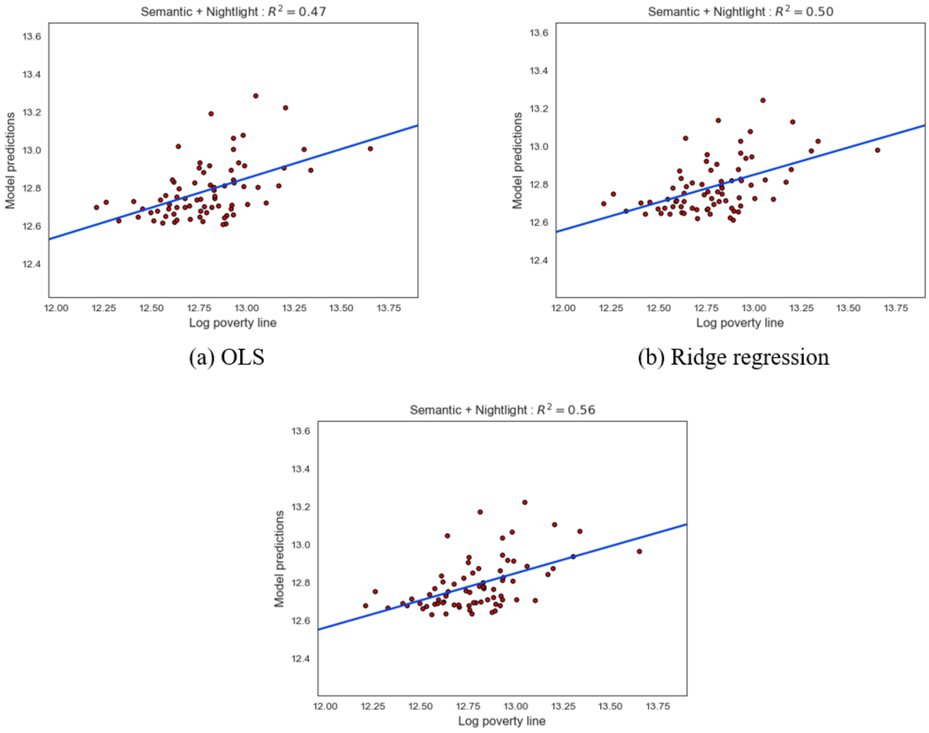


Figure 4.18: The prediction results of OLS with the nighttime lights intensity as the only predictor

Second, we combined the mean of luminosity and the extracted image features in the semantic segmentation approach as predictors for the regression model. In the end, we have six independent variables to estimate the poverty of a region. We implemented the same configuration in the model as described in Table 4.14. The results show that the nighttime lights and daytime image features can explain up to 56 percent of the variance in log poverty lines. Furthermore, by adding the new feature, the performance of the model increases in OLS, Ridge, and Lasso regression by 0.05, 0.03, and 0.08, respectively. Of all the models that have been built, this model performs the best in predicting the poverty line measurement of a region.



(a) OLS

(b) Ridge regression

(c) Lasso regression

Figure 4.19: The prediction results of regression models that include the nighttime lights intensity

The significance test of the OLS coefficient was performed to see the relevance of the features and the poverty. In the table below, t is the t-statistics, and $P > |t|$ is the 2-tailed p-value used in testing the null hypothesis that the coefficient is zero. A p-value of less than 0.05 is considered to be statistically significant. As reported in the table, the coefficient for all predictors is significantly different from zero because its p-value is smaller than 0.05. It means that there is an association between the six features and the poverty line measurement of a region. Furthermore, we presented the R^2 decomposition in Table 4.16. The relative importance is an averaging of the sequential sum-of-squares obtained from all possible orderings of the independent variables. The metric in Table 4.16 shows that the mean of luminosity contributes the most to the R^2 by 38.71 percent. Meanwhile, the other five predictors share a total of 61.29 percent to the predictive power of the model to predict poverty.

Features	**Coefficient Est.**	t	$P < \lvert t \rvert$
(Intercept)	12.8446	51.625	0.000
% road	0.00567	9.873	0.000
% building	-0.00572	31.684	0.000
% vegetation	-0.00239	188.221	0.000
% water	0.00577	38.632	0.000
% ground	-0.00332	50.423	0.000
Mean of luminosity	0.01072	2.672	0.009

Table 4.15: The hypothesis testing of least squares coefficient estimates

Features	**Relative Importance**
% road	0.0995
% building	0.1682
% vegetation	0.1614
% water	0.1212
% ground	0.0626
Mean of luminosity	0.3871

Table 4.16: Share of variance explained (R^2) by each feature

4.2.5 Testing on Higher Level of Administrative Unit

In order to assess the generalization of the models, we tested the models on the provincial level dataset described in Table 3.2. The testing dataset covers the administrative area one level higher than the training dataset. As described earlier, we used the municipality level data to train and build the models. We evaluated whether the models can make poverty predictions for different administrative level. We ran the tests using the models from the multistep learning, naive, and semantic segmentation approaches. The results obtained from each approach are summarized in the table below. The image features from municipality level imagery can explain 60, 31, and 75 percent of the poverty line variation in the provincial level using multistep learning, naive, and semantic segmentation approaches, respectively. Overall, the R^2s increase compared to the test results using municipality level data. These results indicate that we can use the models to predict poverty in higher level administrative data.

Approach	R^2
Multistep Learning	0.60
Naive	0.31
Semantic Segmentation	0.75

Table 4.17: The test on the provincial level data

4.2.6 Testing on Out-of-country Data

We also evaluated the out-of-country generalization of the models. The landscape features of each country in this world could be very different. We examined whether our models trained using the landscape features from Indonesia can be used to estimate poverty in another country. We used the data from Sri Lanka and Thailand provided in Table 3.3 as our testing datasets. Table 4.18 presents the test results for each approach. As we can see from the table below that the overall out-of-country evaluation gives us poor prediction results. Most of the resulted R^2s in Table 4.18 are really small and decline significantly from our initial testing. Using the baseline approach, multistep learning, the value of R^2 has decreased significantly to 0.14 and 0.17 for the two testing countries. Using the naive approach, there is almost no correlation between the Indonesia landscape features and poverty line of Sri Lanka illustrated by the small value of R^2 which is 0.08. However, the naive model gives us better prediction when tested using Thailand data. On the contrary, the semantic segmentation model predicts poverty in Sri Lanka better than in Thailand. We have tried to tune the hyperparameters of the regression model for the multistep learning and semantic segmentation approaches, but the results are not much different from the values presented in the table. Overall, these results suggest that the models built using the three approaches are not robust to be used to predict poverty in other countries.

Approach	R^2	
	Sri Lanka	**Thailand**
Multistep Learning	0.14	0.17
Naive	0.08	0.46
Semantic Segmentation	0.41	0.04

Table 4.18: The test on the out-of-country data

4.3 Evaluation of Results

We conducted experiments to evaluate the capability of our baseline (multistep learning) and proposed approaches (naive and semantic segmentation) to estimate the poverty of a region. The multistep learning model uses the extracted high-level image features from the CNN used to predict the night lights intensity. Instead of using a two-step predictive model, the naive approach uses only one step by directly predicting poverty from satellite imagery. Different from the two approaches, the semantic segmentation approach performs pixel-wise classification to extract five features related to the area type (road, building, vegetation, water, ground) that are used as the predictors.

We used the experimental results to answer the research questions presented in the first chapter.

1. *How good is the performance of the model if we estimate the poverty of a region only from the satellite imagery?*

   The results presented in the previous section indicate that our semantic segmentation approach gives the best performance compared to the multistep learning and naive approaches. The best reported R^2 achieved by the multistep learning, naive, and semantic segmentation are 0.45, 0.20, and 0.48, respectively. Put differently, the models can explain the variation of log poverty line by 45 percent using the multistep learning approach, 20 percent using the naive approach, and 48 percent using the semantic segmentation approach.

   The naive approach fails to make improvement to the baseline approach. Using the CNN model to directly make predictions from daytime satellite images results in lower performance than the multistep learning. The difference is quite large, which is 0.25. It indicates that high-level satellite features contain more information relevant to poverty than low-level features. On the other hand, the second proposed approach which is semantic segmentation outperforms the baseline approach by 3 percent. Even though it is a relatively small improvement, there are several advantages to this approach compared to the multistep learning. First, this approach needs a smaller number of training images to build the CNN model. Both approaches employ two steps for estimating poverty in which the CNN model built in the first step is used to derive the image features. In our experiment, we used 12,500 training images for multistep learning. For the semantic segmentation, we only used 340 training images that were created from the augmentation of 20 original ground truth images. Second, the regression model of our proposed approach is much simpler than the baseline approach. The multistep learning uses the extracted 100 features (reduced from 4096 features using Principal Component Analysis) to build the regression model. Our proposed approach uses only five semantic features to predict poverty. Therefore, in terms of interpretability, it is better than the multistep learning. The interpretation of its linear model is possible since we know the meaning of each feature. Besides, having much less features could mitigate the overfitting problem. Taken together, our approach appears more capable to predict the variation of poverty than the baseline approach.

A comparison of the three approaches reveals that the semantic features derived from satellite imagery can explain the variation of poverty in a region better than the low-level and high-level image features. This result confirms that it is possible to estimate poverty using only the daytime satellite imagery with a performance that is not inferior to the multistep learning model which also uses nighttime lights in addition to the satellite imagery.

2. *How well does the model perform in comparison to both the model using nighttime lights image alone and the model using a combination of daytime and nighttime lights satellite images?*

   The nighttime lights image is used only by the baseline approach as a proxy for predicting the economic well-being. To evaluate the direct use of nighttime lights to estimate poverty, we extracted the mean of luminosity of each region from the nighttime lights image, and then we built a simple regression model using luminosity data as the only independent variable. The reported R^2 achieved by the model is 0.44. What is interesting about the result is that the model performs almost the same as the baseline approach (0.45). This finding is unexpected and suggests that the nighttime lights intensity contains the same predictive power as the daytime image features used in the multistep learning approach. If we compare the two techniques in terms of their goal to predict poverty, this could be a disadvantage to the multistep learning approach. As explained previously, it requires more efforts to accomplish the task using the multistep learning approach, but its performance is almost similar to a model that merely utilizes the luminosity data. Furthermore, since we know that the nighttime lights variable is relevant to poverty estimation, we experimented with another model that includes the mean of luminosity as one of the predictors other than the landscape features from the semantic segmentation approach. We found that using nighttime lights as the additional information beyond the five semantic features indeed improves the performance of the model by 8 percent. Thus far, this is our best-performed model, and it can explain 56 percent the variance of poverty.

3. *Which landscape features are most correlated with the measure of poverty?*

From the regression model in semantic segmentation approach, we found that the extracted semantic features and luminosity are correlated to the poverty. We calculated the contribution percentage of the correlated predictors to the value of R^2. The result shows that the luminosity is the biggest contributor. As the most important variable in the model, it contributes 38.71 percent to the predictive power. The remaining 61.29 percent is shared by the other five landscape features. The building, vegetation, and water variables each explain 12 to 16 percent variation of the poverty. While road and ground variables contribute the smallest by 6 and 10 percent, respectively. These results confirm that although landscape features explain a share of the variation, they are less correlated to the measure of poverty than the luminosity variable.

4. *How does the level of region for training data affect the result of the model implemented in a higher level of the administrative unit? How is the generalizability of the model to predict poverty in other countries?*

   We examined the generalization capability of the models to fit the higher-level administrative and out-of-country data. The tests of models using provincial level data do not show any significant decrease in the value of R^2. It occurs in all approaches used to build those models. However, poor generalization was identified when the models were used to predict poverty in other countries. Since we only used two countries for testing, it is difficult to make a conclusion from the results. Using the naive approach, the value of R^2 is stable for Thailand but not for Sri Lanka data, while using semantic segmentation approach, the opposite happens. From the experiments, we can say for now that the models can be used in higher-level administrative data, but they are not robust for out-of-country data.

Chapter 5

Conclusion

In the previous chapters, we have discussed and compared multistep learning, naive, and semantic segmentation approaches to predict poverty of a region. We investigated how well the direct use of daytime satellite imagery in the naive and semantic segmentation approaches compared to the use of night lights as a proxy for poverty estimation in the multistep learning. Moreover, we assessed whether the daytime satellite imagery is more relevant to poverty rather than the nighttime lights image.

We experimented with publicly available satellite imagery from Google Maps and NOAA. We analyzed the images using various convolution networks such as VGG F, AlexNet, Resnet10, and SegNet architectures. We also implemented transfer learning strategy to build the models. We utilized the CNN models that have been previously trained on ImageNet dataset to initialize the weights of our CNNs. The fine-tuning of the pre-trained model was successful as it is able to improve the performance of our models. This result supports the idea that the visual filters from general images such as ImageNet can be used as a starting point for processing satellite images.

This study has found that in general, the semantic segmentation approach gives a better result than the multistep learning approach. The landscape features such as road, building, vegetation, water, and ground appear to be more predictive than the high-level image features extracted by the CNN model in the multistep learning. Those landscape features explain collectively 48 percent of the variance in the log poverty line. It surpasses the result of the multistep learning which is 45 percent. Furthermore, the simplicity of the model using only five features also becomes an advantage of this approach. In contrast, the naive model that directly perform prediction from satellite images cannot outperform the result of multistep learning. The naive model can only explain the poverty line variation up to 20 percent. We also performed an evaluation of model performance against night lights. The night lights model surprisingly performs in the way that is almost similar to the multistep learning. The night lights variable alone can explain poverty by 44 percent. The best model in our experiments that achieves 56 percent of R^2 value was obtained by combining the semantic segmentation approach and the night lights data.

Moreover, we calculated the relative importance of each feature in that model. The metric reveals that the luminosity is more relevant to poverty estimation than the landscape features. It contributes 38.71 percent to the predictive power of the regression model, while the share of each landscape feature ranges from 6 to 16 percent. It is expected that the luminosity is more important than other features since adding it as a predictor increases the performance of model significantly. This finding is also clearly supported by the resulted performance of the night lights model presented earlier.

5.1 Discussion and Future Work

The findings of this study raise several questions for further work:

- *Will more data help to obtain a better prediction?*

  The size of the municipalities used as the training set varies with a range of 10.77 to 45,000 km^2. Taking randomly 10 clusters of 5 km by 5 km area (250 km^2) for each region can be considered as a small sample. Moreover, there is a temporal difference between the poverty data and satellite images. It could affect the accuracy of the estimation. The cross-validated prediction of our best model explains up to 56 percent the variation of log poverty lines. It performs fairly well considering those spatial and temporal factors. However, those could be the weakness of this study. Further studies are needed to assess the improvement of model performance with a larger sample. Even, if the poverty data are available in lower level administrative units such as village or household, further research might explore the potential of satellite imagery for small area poverty estimation.

  The generalizability of the models is subject to certain limitation. We have shown that the models of three approaches overall are not robust to predict poverty in other countries. This may be partly due to the fact that the testing countries have different geographic characteristics than the training country, and the landscape features in training country cannot represent other countries. Furthermore, using only two countries might be not enough for testing the out-of-country generalization of the models. Considerably more testing countries will be needed to understand the extent in which the models can generalize to different landscape features.

- *Will a deeper CNN or different feature extractions improve the result of the naive approach?*

  Due to the limited GPU capability, we only used shallow convolution networks in our experiments. More experiments using deeper network architectures with larger batch size might provide a more accurate prediction, especially for the naive approach. Besides, in naive approach, we extracted the low-level satellite features using CNN. Experiments using other feature extractions such as RGB, HOG, or color histogram could also be conducted to determine the effectiveness of this approach.

- *Will a better prediction of area type help to improve the poverty estimation?*

  The effectiveness of semantic segmentation approach depends on the visibility of landscape features in the satellite imagery. It cannot be applied in the region with only blurry satellite images available since the CNN model might not be able to identify the class of each pixel correctly. As in the multistep learning, the performance of the CNN to perform semantic segmentation in this approach is a rough indicator that determines the result of poverty estimation. The landscape features have been proven to be relevant to poverty. Therefore, it would be better to improve the performance of the CNN model that derive those features. Larger training dataset and batch size could result in more accurate SegNet model, and it is ultimately expected to also improve the poverty estimation result.

- *Will more category of landscape feature improve the performance of the model?*

  Since we utilized the satellite images from Google Maps with limited visible landscape features, we can only categorize the landscape features into six categories. Further experiments using higher resolution satellite imagery from different sources could be conducted to extract more features other than those six features. If they are relevant to poverty prediction, the model could have better performance.

Further experiments with a greater focus on the list discussed above could produce interesting findings that account more for the effectiveness of the proposed approaches. Finally, notwithstanding the relatively limited sample and resource, the findings of this research provide insights that there is a possibility to utilize novel data sources such as satellite image to predict poverty of a region. Moreover, it might be possible to use our proposed methods to estimate other socioeconomic indicators.

Chapter 6

References

[1] *The sustainable development goals.* United Nations Publications, 2017.

[2] *Introduction to poverty analysis (English).* World Bank Group, 2014.

[3] J. Feng, “How should we measure poverty?”, *World Economic Forum*, Nov. 2014. [Online]. Available: `https://www.weforum.org/agenda/2014/11/how-should-we-measure-poverty/`.

[4] J. E. Blumenstock, “Fighting poverty with data”, *Science*, vol. 353, no. 6301, pp. 753–754, 2016.

[5] A. Llorente, M. Garcia-Herranz, M. Cebrian, and E. Moro, “Social media fingerprints of unemployment”, *PloS one*, vol. 10, no. 5, e0128692, 2015.

[6] H. Choi and H. Varian, “Predicting the present with google trends”, *Economic Record*, vol. 88, no. s1, pp. 2–9, 2012.

[7] J. Blumenstock, G. Cadamuro, and R. On, “Predicting poverty and wealth from mobile phone metadata”, *Science*, vol. 350, no. 6264, pp. 1073–1076, 2015.

[8] Google. (-). Google earth engine - datasets. (accessed July 2, 2018), [Online]. Available: `https://earthengine.google.com/datasets/`.

[9] V. Henderson, A. Storeygard, and D. N. Weil, “A bright idea for measuring economic growth”, *American Economic Review*, vol. 101, no. 3, pp. 194–99, 2011.

[10] X. Chen and W. D. Nordhaus, “Using luminosity data as a proxy for economic statistics”, *Proceedings of the National Academy of Sciences*, vol. 108, no. 21, pp. 8589–8594, 2011.

[11] S. Michalopoulos and E. Papaioannou, “Pre-colonial ethnic institutions and contemporary african development”, *Econometrica*, vol. 81, no. 1, pp. 113–152, 2013.

[12] ——, “National institutions and subnational development in africa”, *The Quarterly Journal of Economics*, vol. 129, no. 1, pp. 151–213, 2013.

[13] M. L. Pinkovskiy, “Growth discontinuities at borders”, *Journal of Economic Growth*, vol. 22, no. 2, pp. 145–192, 2017.

[14] M. Harari, “Cities in bad shape: Urban geometry in india”, *February. http://real. wharton. upenn. edu/˜ harari/Harari_Papers/CityShapeHarariMarch2016_updated. pdf*, 2016.

[15] M. Pinkovskiy and X. Sala-i-Martin, “Lights, camera… income! illuminating the national accounts-household surveys debate”, *The Quarterly Journal of Economics*, vol. 131, no. 2, pp. 579–631, 2016.

[16] N. Jean, M. Burke, M. Xie, W. M. Davis, D. B. Lobell, and S. Ermon, “Combining satellite imagery and machine learning to predict poverty”, *Science*, vol. 353, no. 6301, pp. 790–794, 2016.

[17] A. Head, M. Manguin, N. Tran, and J. E. Blumenstock, “Can human development be measured with satellite imagery?”, in *Proceedings of the Ninth International Conference on Information and Communication Technologies and Development*, ACM, 2017, p. 8.

[18] P. K. Suraj, A. Gupta, M. Sharma, S. B. Paul, and S. Banerjee, “On monitoring development using high resolution satellite images”, *arXiv preprint arXiv:1712.02282*, 2017.

[19] A. Perez, C. Yeh, G. Azzari, M. Burke, D. Lobell, and S. Ermon, “Poverty prediction with public landsat 7 satellite imagery and machine learning”, *arXiv preprint arXiv:1711.03654*, 2017.

[20] R. Engstrom, J. S. Hersh, and D. Newhouse, “Poverty from space: Using high-resolution satellite imagery for estimating economic well-being”, 2017.

[21] B. Klemens, A. Coppola, and M. Shron, “Estimating local poverty measures using satellite images: A pilot application to central america”, 2015.

[22] BPS-Indonesia. (-). Kemiskinan dan ketimpangan (indonesia). (accessed June 28, 2018), [Online]. Available: `https://www.bps.go.id/subject/23/kemiskinan-dan-ketimpangan.html`.

[23] X. Amatriain. (2016). What’s the relationship between machine learning and data mining? (accessed January 16, 2018), [Online]. Available: `https://medium.com/@xamat/what-s-the-relationship-between-machine-learning-and-data-mining-8c8675966615`.

[24] E. Alpaydin, *Introduction to machine learning.* MIT press, 2014.

[25] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning.* Springer, 2013, vol. 112.

[26] F. Lindsten, N. Wahlström, A. Svensson, and T. B. Schön, “Statistical machine learning”, 2018.

[27] R. Tibshirani, “Regression shrinkage and selection via the lasso”, *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288, 1996.

[28] J. Patterson and A. Gibson, *Deep Learning: A Practitioner’s Approach.* " O’Reilly Media, Inc.", 2017.

[29] A. Castrounis. (2016). Artificial intelligence, deep learning, and neural networks, explained. (accessed June 25, 2018), [Online]. Available: `https://www.kdnuggets.com/2016/10/artificial-intelligence-deep-learning-neural-networks-explained.html`.

[30] T. Keenan. (2017). Neural networks demystified. (accessed June 25, 2018), [Online]. Available: `https://www.upwork.com/hiring/data/neural-networks-demystified/`.

[31] K. Chen. (2017). Gradient descent and backpropagation. (accessed June 28, 2018), [Online]. Available: `https://www.linkedin.com/pulse/gradient-descent-backpropagation-ken-chen`.

[32] D. Gupta. (2017). Fundamentals of deep learning – activation functions and when to use them? (accessed June 28, 2018), [Online]. Available: `https://www.analyticsvidhya.com/blog/2017/10/fundamentals-deep-learning-activation-functions-when-to-use-them/`.

[33] S. Sharma. (2017). Activation functions: Neural networks. (accessed June 28, 2018), [Online]. Available: `https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6`.

[34] Y. LeCun, Y. Bengio, *et al.*, "Convolutional networks for images, speech, and time series", *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[35] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning.* MIT press Cambridge, 2016, vol. 1.

[36] S. Jain. (2018). An overview of regularization techniques in deep learning (with python code). (accessed July 1, 2018), [Online]. Available: `https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/`.

[37] F. Doukkali. (2017). Batch normalization in neural networks. (accessed July 1, 2018), [Online]. Available: `https://towardsdatascience.com/batch-normalization-in-neural-networks-1ac91516821c`.

[38] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth", in *European Conference on Computer Vision*, Springer, 2016, pp. 646–661.

[39] MatConvNet. (2017). Pretrained models. (accessed July 25, 2018), [Online]. Available: `http://www.vlfeat.org/matconvnet/pretrained/`.

[40] T. Ghosh, S. J. Anderson, C. D. Elvidge, and P. C. Sutton, "Using nighttime satellite imagery as a proxy measure of human well-being", *Sustainability*, vol. 5, no. 12, pp. 4988–5019, 2013.

[41] C. D. Elvidge, P. C. Sutton, T. Ghosh, B. T. Tuttle, K. E. Baugh, B. Bhaduri, and E. Bright, "A global poverty map derived from satellite data", *Computers & Geosciences*, vol. 35, no. 8, pp. 1652–1660, 2009.

[42] WorldBank. (2013). Dc big data exploration final report. (accessed July 9, 2018), [Online]. Available: `https://www.scribd.com/doc/142012481/DC-Big-Data-Exploration-Final-Report?cid=CTR_TwitterWBopenfinances_D_EXT`.

[43] D. J. Lary, A. H. Alavi, A. H. Gandomi, and A. L. Walker, "Machine learning in geosciences and remote sensing", *Geoscience Frontiers*, vol. 7, no. 1, pp. 3–10, 2016.

[44] S. Basu, S. Ganguly, S. Mukhopadhyay, R. DiBiano, M. Karki, and R. Nemani, "Deepsat: A learning framework for satellite imagery", in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, 2015, p. 37.

[45] Y. Zhong, F. Fei, Y. Liu, B. Zhao, H. Jiao, and L. Zhang, "Satcnn: Satellite image dataset classification using agile convolutional neural networks", *Remote Sensing Letters*, vol. 8, no. 2, pp. 136–145, 2017.

[46] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep learning earth observation classification using imagenet pretrained networks", *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 105–109, 2016.

[47] N. Audebert, B. Le Saux, and S. Lefèvre, "Semantic segmentation of earth observation data using multimodal and multi-scale deep networks", in *Asian Conference on Computer Vision*, Springer, 2016, pp. 180–196.

[48] M. Längkvist, A. Kiselev, M. Alirezaie, and A. Loutfi, “Classification and segmentation of satellite orthoimagery using convolutional neural networks”, *Remote Sensing*, vol. 8, no. 4, p. 329, 2016.

[49] R. Kemker, C. Salvaggio, and C. Kanan, “Algorithms for semantic segmentation of multispectral remote sensing imagery using deep learning”, *ISPRS Journal of Photogrammetry and Remote Sensing*, 2018.

[50] -. (2017). Poverty data. (BPS - Statistics Indonesia, accessed July 12, 2018), [Online]. Available: `https://bps.go.id/subject/23/kemiskinan-dan-ketimpangan.html`.

[51] W. Bank. (2018). Povcalnet: An online analysis tool for global poverty monitoring. (accessed November 18, 2018), [Online]. Available: `http://iresearch.worldbank.org/PovcalNet/home.aspx`.

[52] I. C. P. d. World Bank. (2018). Ppp conversion factor. (accessed November 18, 2018), [Online]. Available: `https://data.worldbank.org/indicator/PA.NUS.PPP`.

[53] -. (2018). Google maps platform - maps javascript api. (accessed July 18, 2018), [Online]. Available: `https://developers.google.com/maps/documentation/javascript/maptypes#WorldCoordinates`.

[54] ——, (2013). Version 4 dmsp-ols nighttime lights time series. (NOAA’s National Geophysical Data Center, accessed May 23, 2018), [Online]. Available: `https://ngdc.noaa.gov/eog/dmsp/downloadV4composites.html`.

[55] A. Albert, J. Kaur, and M. C. Gonzalez, “Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale”, in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2017, pp. 1357–1366.

[56] J. Lee. (2018). How to do semantic segmentation using deep learning. (accessed July 20, 2018), [Online]. Available: `https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef`.

[57] H. Automotive and I. Laboratory. (2018). Semantic segmentation editor. (accessed September 1, 2018), [Online]. Available: `https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor`.

[58] Caffe. (2018). Model zoo. (accessed October 1, 2018), [Online]. Available: `https://github.com/BVLC/caffe/wiki/Model-Zoo`.

[59] O. M. Parkhi, A. Vedaldi, A. Zisserman, *et al.*, “Deep face recognition.”, in *BMVC*, vol. 1, 2015, p. 6.

[60] Stephanie. (2015). Welch’s test for unequal variances. (accessed January 16, 2018), [Online]. Available: `https://www.statisticshowto.datasciencecentral.com/welchs-test-for-unequal-variances/`.

[61] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”, *arXiv preprint arXiv:1511.00561*, 2015.