A REPORT ON

# POVERTY ESTIMATION USING SATELLITE IMAGERY THROUGH DEEP LEARNING

by

Manav Kaushik (2016B3A30472P)

Prepared in Partial Fulfillment of

**Study Project**
**Course No: EEE F266**

Submitted to
**Prof. Surekha Bhanot**
Department of Electrical & Electronics Engineering

**BITS Pilani-** Pilani Campus

# Acknowledgement

I would like to express my deepest gratitude to my guide, mentor and instructor-in-charge, Prof. Surekha Bhanot for her consistent motivation and guidance throughout the duration of our Project which helped us to achieve the set target and complete the project within the stipulated timeline.

This would not have been possible without her guidance and support.

## Table of Content

## List of Figures:

## List of Tables:

## PROBLEM STATEMENT

Reliable data on economic livelihoods and local poverty remain scarce especially in the developing world, hampering efforts to study these indicators and to design policies that improve them for the people of these countries.

Thus, the problem that I try to solve through this project is:
**The Estimation of Poverty of specific regions using Day Time Satellite Imagery and Night-Time Light Intensity data of those regions through Convolutional Neural Networks and Transfer Learning.**

# INTRODUCTION

Accurate measurements of the economic characteristics of any country's population critically influence both research and policy development. These measurements not only help to shape decisions by individual governments about how to allocate scarce resources but at the same time provide the foundation for global efforts to understand and track the progress towards improving livelihoods of the citizens of that country. Although the quantity and quality of economic, financial and household data available in developing countries have improved in the past few years, data on some major measures of economic development such as poverty, living standards, etc are still lacking for several developing countries of the world. This data gap is badly affecting the efforts to identify and understand variation in these essential indicators and to target policy changes and interventions effectively to the regions where it is needed the most.

I draw inspiration from the work of Neal Jean et. al (2016) and aim to help predict poverty of a given region by combining day-time satellite Imagery and Convolutional Neural Network with night-time light intensity data. While talking about using day-time satellite images, I am basically using features hidden in these high-resolution day time images which will eventually help us to map the poverty in our region of Interest.

A simple question which might arise here is why to deal with day-time images while predicting poverty if we are trying to majorly use nighttime light intensity data? A more direct approach that I can think of is directly predicting Poverty using Night-time imagery from the satellite. Nighttime images can directly tell us about some of the key aspects that we may be looking for like how well lit our area of interest is and thus, accordingly we can divide the area into low, medium or high poverty ranges to effectively map poverty. But the major issue here, as also discussed by the author Neal Jean well in his paper, is that the nighttime satellite imagery can be very deceptive as a highly lit area may be covered by dense clouds or an area with low light intensity may appear bright because of its surrounding areas. Also, a sparsely populated area cannot be directly considered as a low poverty area. Thus, it is not difficult to agree on the fact that nighttime images are not the best parameter to map poverty in a given region.

Talking about day-time images, earlier scientists had to manually select the relevant features from their dataset in order to train their models and this process was known as feature engineering or simply feature selection. However, this can be a very complex task as it is very difficult to extract features that will be important and benefit the model especially when we talk about image data. This is where the role of Convolutional Neural Networks comes into play. One of the greatest advantages of convolutional neural networks is that they can learn appropriate features by themselves without any explicit training. We can simply provide them with raw images as inputs, and CNN can learn how to fetch the appropriate features for our model training.

At last, let's focus on Neal Jean's paper where he explains the developed model. In this paper, they begin with a Convolutional Neural Network which is pre-trained on the ImageNet dataset and then fine-tune it using satellite daytime images and nighttime light intensity data using transfer learning. A final softmax classifier layer is used to estimate the nighttime light intensity of the particular region of interest. The paper then tries to extend its goal so that the CNN features learned from the nightlights classification task can be used to predict indicators of interest to the international scientific community, such as poverty, or wealth.

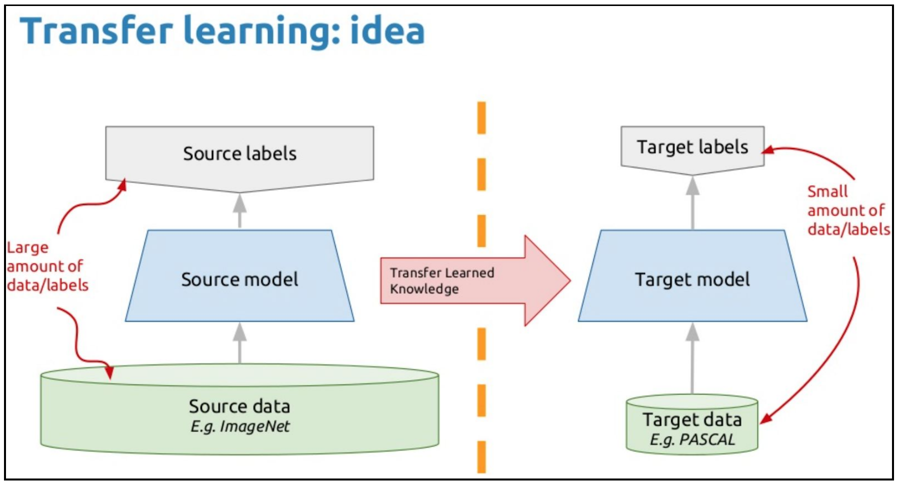
## PROBLEM SETUP

Let's begin by reviewing transfer learning and convolutional neural networks which are the building blocks of our approach in this project.

### Transfer Learning

I have formalized transfer learning as in (Pan and Yang 2010): A The domain $D = \{X, P(\chi)\}$ consists of feature space $\chi$ and a marginal probability

*Fig. 1: Transfer Learning*



distribution $P(\chi)$. Given a domain, a task $T = \{Y, f(\cdot)\}$ comprises of a label space Y and a prediction function f(·) which models as $P(y|x)$ for $y \in Y$ and $x \in X$. Given a source domain $D_S$ and learning task $T_S$, and a target domain $D_T$ and learning task $T_T$, transfer learning tries to improve the learning of the target predictive function $f_T(.)$ in $T_T$ using the knowledge from $D_S$ and $T_S$, where $D_S \neq D_T$, $T_S \neq T_T$ or both. Transfer learning is particularly effective when given labelled source domain data $D_S$ and target domain data $D_T$, we find that $|D_T| \ll |D_S|$. In this current setting, we wish to focus on more than two related learning tasks. First, let's define a transfer learning problem $P = (D, T)$ as a domain-task pair. The transfer learning graph is then defined as follows: A transfer learning graph $G = (V, E)$ is a directed acyclic graph where vertices $V = \{P_1, \cdots, P_V\}$ are transfer learning problems and $E = \{(P_{i1}, P_{j1}), \cdots, (P_{ie}, P_{je})\}$ is an edge set. For every transfer learning problem $P_i = (D_i, T_i) \in V$, the final aim is to improve the learning of the target prediction function $f_i(\cdot)$ in $T_i$ using the knowledge gained in $\cup$ $(j, i) \in E P_j$.

## Convolutional Neural Networks

Deep learning approaches are based on automatically learning nested, hierarchical representations of data. Deep feedforward neural networks are a typical example of deep learning models. Convolutional Neural Networks (CNN) include convolutional operations over the input and are designed specifically for vision tasks. Convolutional filters are useful for encoding translation invariance, a key concept for discovering useful features in images (Bouvrie 2006). A CNN is a general function approximator defined by a set of convolutional and fully connected layers ordered such that the output of one layer is the input of the next. For image data, the first layers of the network typically learn low-level features such as edges and corners, and further layers learn high-level features such as textures and objects (Zeiler and Fergus 2013). Taken as a whole, a CNN is a mapping from tensors to feature vectors, which become the input for a final classifier. A typical convolutional layer maps a tensor $x \in \Re^{h\times w\times d}$ to $g_i \in \Re^{h\times w\times d}$ such that

$$g_i = p_i(f_i(W_i * x + b_i)),$$

where for the $i$-th convolutional layer, $W_i \in \Re^{l\times l\times \widehat{d}}$ is a tensor of $\widehat{d}$ convolutional filter weights of size $l \times l$, $(*)$ is the 2-dimensional convolution operator over the last two dimensions of the inputs, bi is a bias term, fi is an elementwise nonlinearity function (e.g., a rectified linear unit or ReLU), and pi is a pooling function. The output dimensions $\widehat{h}$ and $\widehat{w}$ depend on the stride and zero-padding parameters of the layer, which controls how the convolutional filters slide across the input. For the first convolutional layer, the input dimensions *h, w,* and *d* can be interpreted as height, width, and the number of colour channels of an input image, respectively.

In addition to convolutional layers, most CNN models have fully connected layers in the final layers of the network. Fully connected layers map an unrolled version of the input $\widehat{x} \in \Re^{hwd}$, which is a one-dimensional vector of the elements of a tensor $x \in \Re^{h\times w\times d}$, to an output $g_i \in \Re^{k}$ such that

$$g_i = f_i(W_i\widehat{x} + b_i),$$

where $W_i \in \Re^{h\times wd}$ is a weight matrix, $b_i$ is a bias term, and $f_i$ is typically a ReLU nonlinearity function. The fully connected layers encode the input examples as feature vectors, which are used as inputs to a final classifier. Since the fully connected layer looks at the entire input at once, these feature vectors "summarize" the input into a feature vector for classification. The model is trained end-to-end using minibatch gradient descent and backpropagation. After training, the output of the final fully connected layer can be interpreted as an encoding of the input as a feature vector that facilitates classification. These features often represent complex compositions of the lower-level features extracted by the previous layers (e.g., edges and corners) and can range from grid patterns to animal faces (Zeiler and Fergus 2013; Le et al. 2012).

### Combining Transfer Learning and Deep Learning

The low-level and high-level features learned by a CNN on a source domain are transferred to augment learning in a different but related target domain. For target problems with abundant data, we can transfer low-level features, such as edges and corners, and learn new high-level features specific to the target problem. For target problems with limited amounts of data, learning new high-level features is difficult. However, if the source and target domain are sufficiently similar, the features learned by CNN on the source task can be used for the target problem. Deep features extracted from CNNs trained on large annotated datasets of images have been used as generic features quite effectively for a wider range of vision tasks (Donahue et al. 2013; Oquab et al. 2014).

## OBJECTIVES

- **Data Collection:**
    - Collect nighttime light intensity data from the National Oceanic and Atmospheric Administration (NOAA)
    - Download economic data from the Demographic and Health Surveys (DHS Data)
    - Download day-time satellite of the region of interest from Google Static Maps

- **Setup a Transfer Learning Model:**
    - Prepare a VGG16 model for transfer learning
    - Use ImageNet dataset for training weights with VGG16 model to transfer knowledge to predict nightlight intensity.

- **Poverty Estimation using Deep Learning and Transfer Learning:**
    - Fine-tune the CNN model using satellite daytime images and nighttime light intensity data through transfer learning
    - Use the features that the model has learned from the nighttime light intensity classification task to map poverty in the region of interest.

## DATA SOURCES:

- **DHS Data**: Economic & Geographical Data
- **DMPS-OLS**: Nighttime light intensity data
- **Google Static Map API**: Day time satellite imagery data

**COUNTRY ANALYSED:** Rwanda

## METHODOLOGY

## 1. Data Collection & Preprocessing:

### a. Night Light Data:

The data for Nighttime Light Intensity was downloaded from DMSP-OLS Website. The data represents the nighttime luminosity of different regions of the world.

A raw image (filtered) of the nighttime light intensity is shown below:

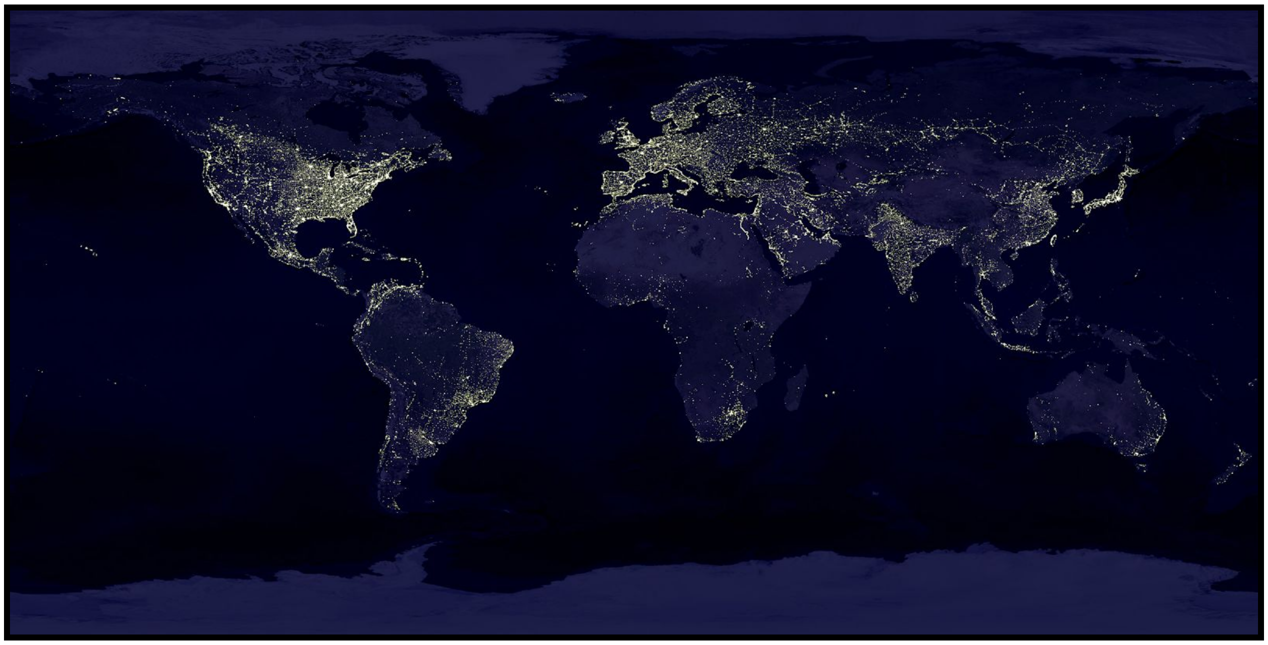


*Fig. 2: Nightlight Sample Image*

### b. DHS Data & CLustering:

Demographic and Health Surveys (DHS) are nationally-representative household surveys that provide data for a wide range of monitoring and impact evaluation indicators in the areas of population, health, and nutrition. For this assignment, I downloaded the 2012 Rwandan DHS data.

The immediate goal is to take the raw survey data, covering 12,540 households, and compute the average household wealth for each survey cluster (cluster can be thought of as a village).

Geographic Datasets: This dataset contained the location of each cluster in Rwanda. It is in the format of shapefile, which needs QGIS or other GIS software to open.

The cluster locations, overlaid on the nightlights data, are shown in the figure below.

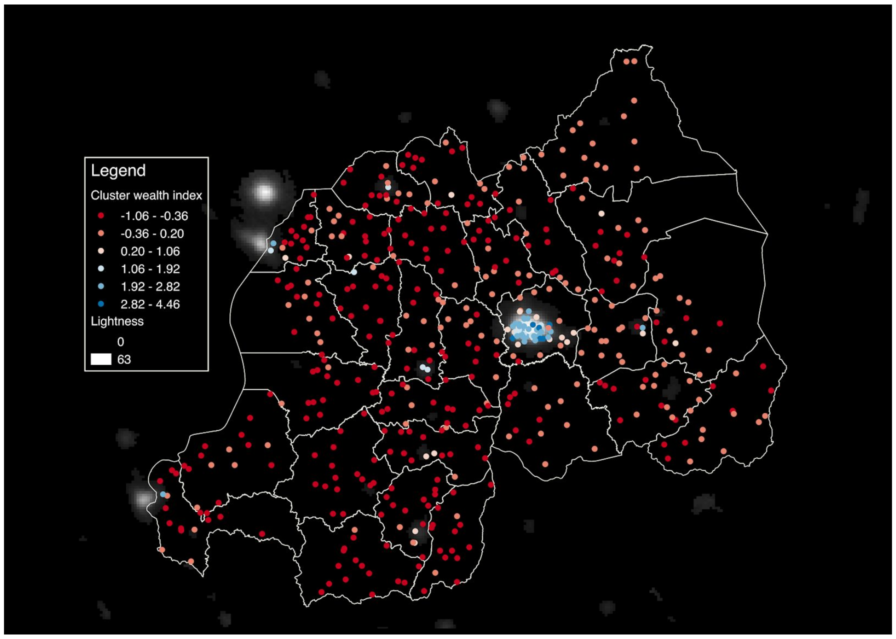


*Fig. 3: Rwanda Wealth Clusters*

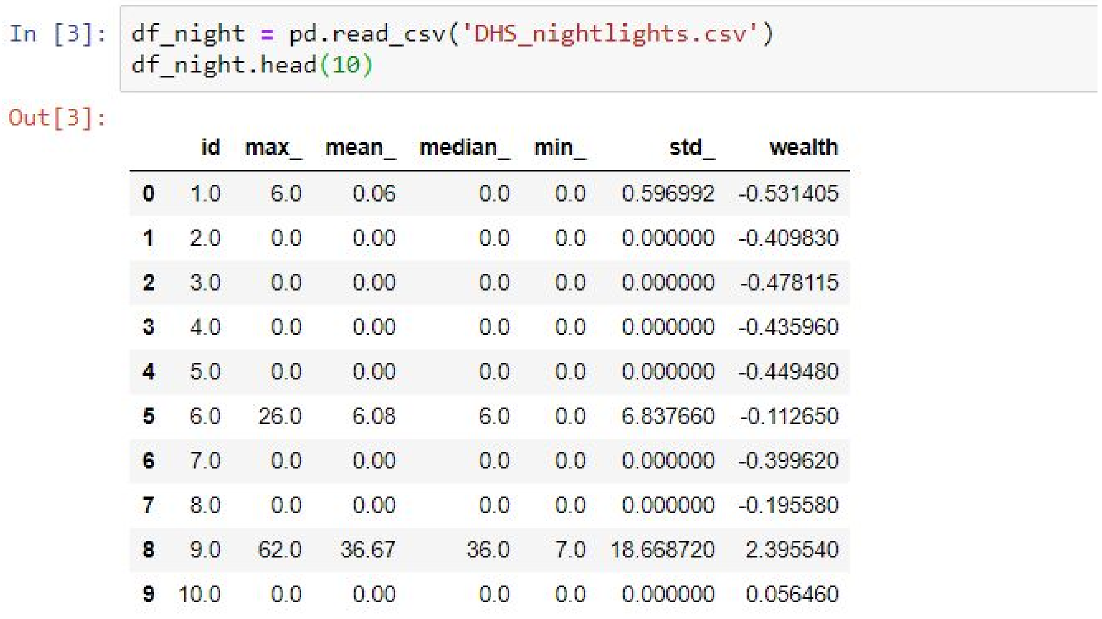
## c. Merging Nightlights & DHS Data at Cluster Level:

Here I merge the two data that I have collected in the previous two steps i.e.

- Nighttime Light Intensity Data (492 rows, 4 columns)
- DHS Cluster Data for Average Wealth (492 rows, 1 column)

The resolution of each pixel in the nightlight image is about 1km. I used 10 pixels x 10 pixels to average the luminosity of each cluster.

I start by just taking the Mean of the luminosity in the 100 pixels and comparing this to cluster average wealth. I also compute other luminosity characteristics of each cluster, such as the Max, Min, Standard Deviation of the 100-pixel values.



```
In [3]: df_night = pd.read_csv('DHS_nightlights.csv')
        df_night.head(10)

Out[3]:
```

|   | id | max_ | mean_ | median_ | min_ | std_ | wealth |
|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 6.0 | 0.06 | 0.0 | 0.0 | 0.596992 | -0.531405 |
| 1 | 2.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.000000 | -0.409830 |
| 2 | 3.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.000000 | -0.478115 |
| 3 | 4.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.000000 | -0.435960 |
| 4 | 5.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.000000 | -0.449480 |
| 5 | 6.0 | 26.0 | 6.08 | 6.0 | 0.0 | 6.837660 | -0.112650 |
| 6 | 7.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.000000 | -0.399620 |
| 7 | 8.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.000000 | -0.195580 |
| 8 | 9.0 | 62.0 | 36.67 | 36.0 | 7.0 | 18.668720 | 2.395540 |
| 9 | 10.0 | 0.0 | 0.00 | 0.0 | 0.0 | 0.000000 | 0.056460 |

*Fig. 4: Code Snippet 1*

To read the raw raster (nightlights) files, I used GDAL library.

A plot of Average nighttime luminosity vs Average cluster wealth is shown below:

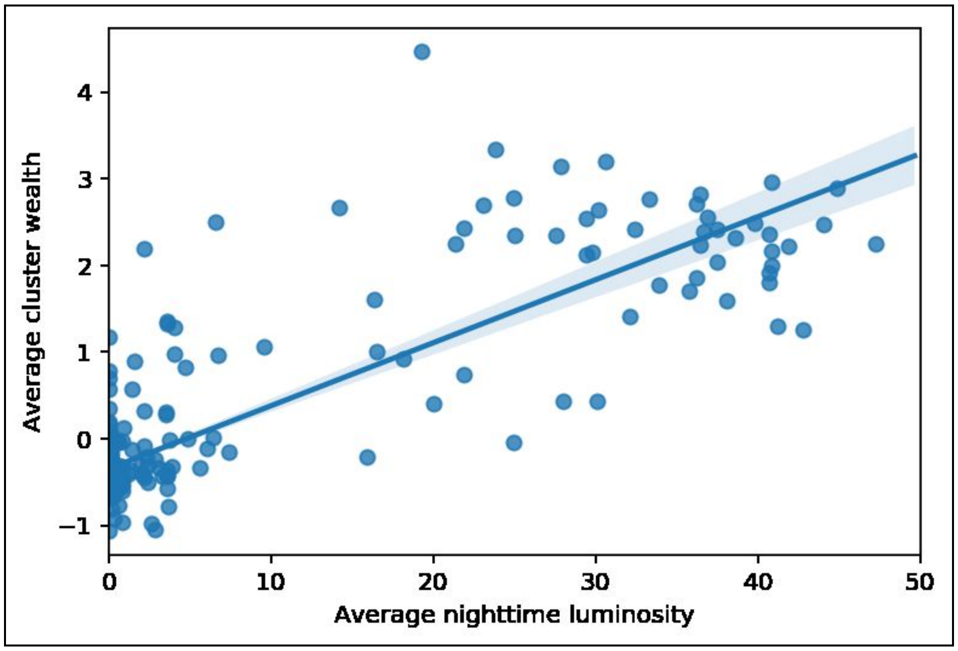


*Fig. 5: Analysis of Wealth Index & Luminosity*

## d. Daytime Satellite Imagery:

e.

The Google Static Maps API has been used to download the satellite images for day-time using shapefiles that specifies the borders of Rwanda. Some sample images are shown below:

*Fig. 6: Sample Satellite Images (Day-time)*

All the files were organized in 64 folders (as the light intensity was measured from 0 to 64), with the folder name indicating the nightlight intensity of the pixel corresponding to the daytime image i.e. if you download a daytime image for which the nighttime pixel has value 35, store that daytime image in a folder labelled '35'. This way, all the day-time images within each folder will have the same nightlight luminosity. The file name is columnIndex_rowIndex.jpg, in which the row index and column index are the indexes in the nightlight image as you can see from the figure below.

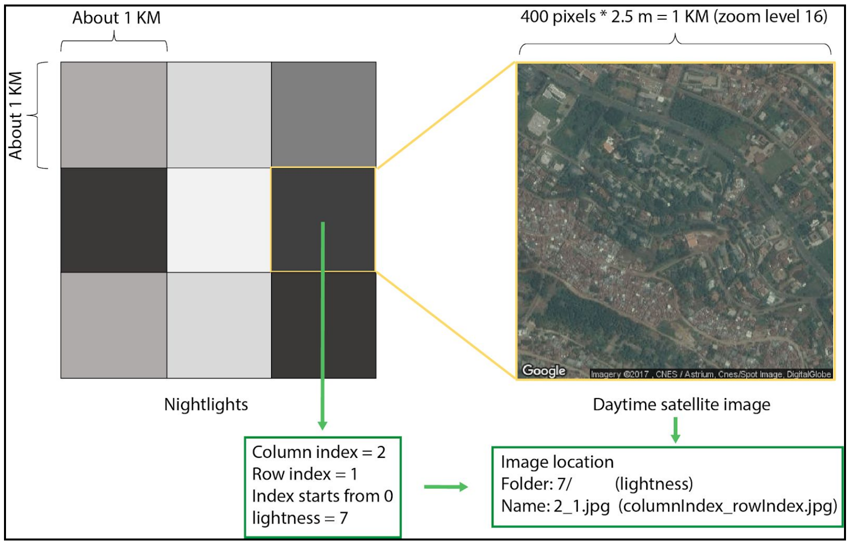


*Fig. 7: Image Representation*

## 2. Setup a Transfer Learning Model

### a. Model Selection for Pre-Training:

A VGG16 model was used for our pre-training purpose. VGG 16 has kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3x3 kernel-sized filters one after another. With a given receptive field multiple stacked smaller size kernel is better than the one with a larger size kernel because multiple non-linear layers increase the depth of the network which enables it to learn more complex features, and that too at a lower cost.

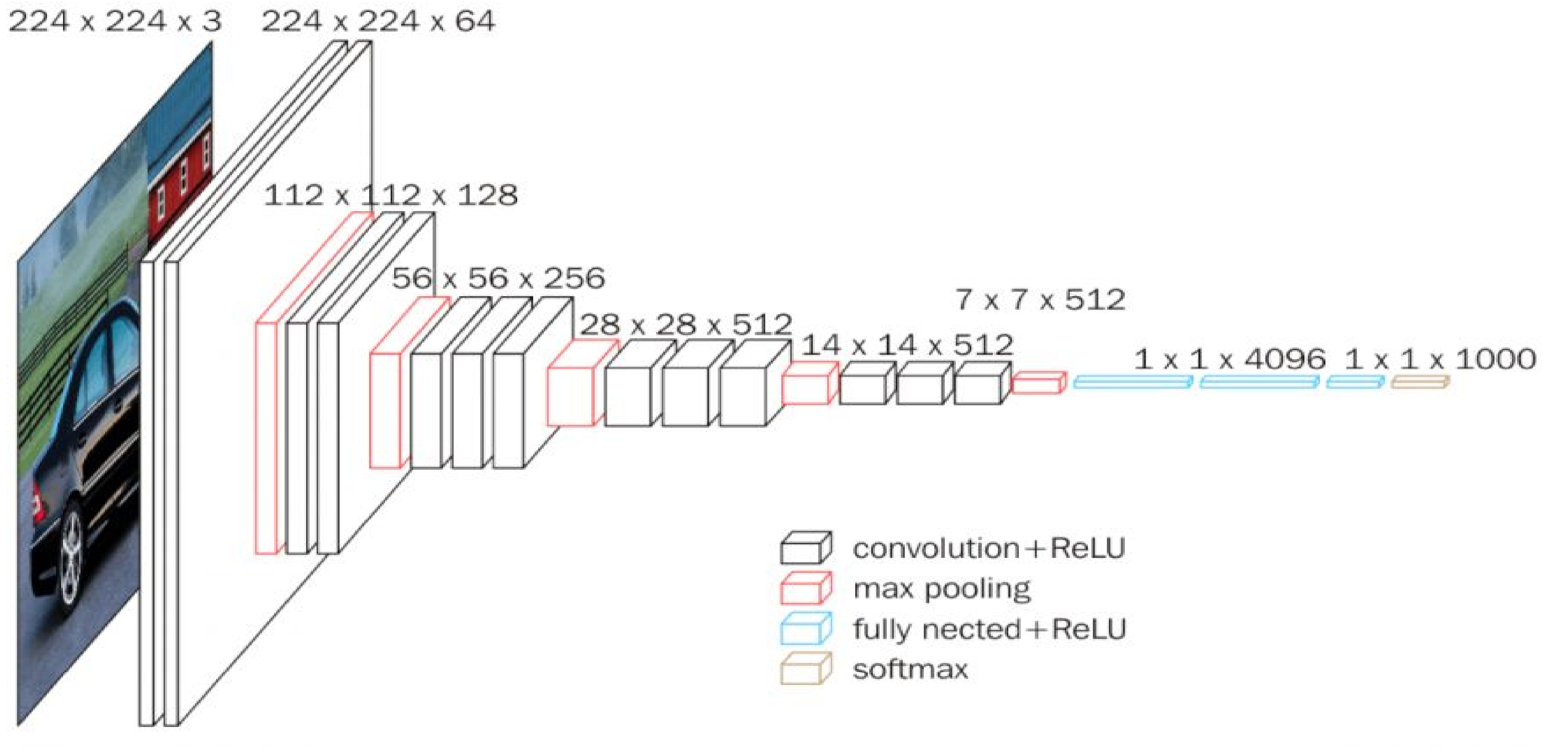


*Fig. 8: VGG16 Architecture*

## <u>**About VGG16:**</u>

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes. It was one of the famous model submitted to ILSVRC-2014. It makes the improvement over AlexNet by replacing large kernel-sized filters (11 and 5 in the first and second convolutional layer, respectively) with multiple 3×3 kernel-sized filters one after another. VGG16 was trained for weeks and was using NVIDIA Titan Black GPU’s.

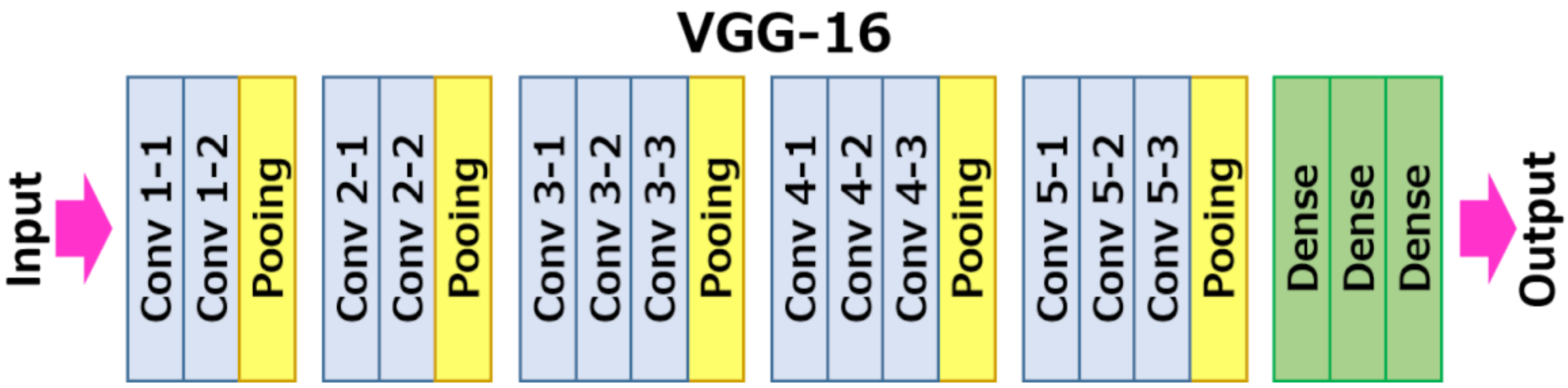


*Fig. 9: VGG Layers*

The input to cov1 layer is of fixed size 224 x 224 RGB image. The input image is passed through a stack of convolutional (conv.) layers, where the filters were used with a very small receptive field: 3×3 (which is the smallest size to capture the notion of left/right, up/down, centre). In one of the configurations, it also utilizes 1×1 convolution filters, which can be seen as a linear transformation of the input channels (and then followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of Conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 Conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the Conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2×2 pixel window, with stride 2.

Three Fully-Connected (FC) layers follow a stack of convolutional layers (which has a different depth in different architectures): the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU) non-linearity. It is also noted that none of the networks (except for one) contain Local Response Normalisation (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

### b. Dataset for Pre-Training:

As suggested in Jean et. al., we have used the pre-training dataset as ImageNet which has an object classification image dataset of over 14 million images with a total of 1000 class labels. CNN models trained with ImageNet are considered good feature extractors for low-level and mid-level features like edges and corners.

Some of the samples from ImageNet are shown below:

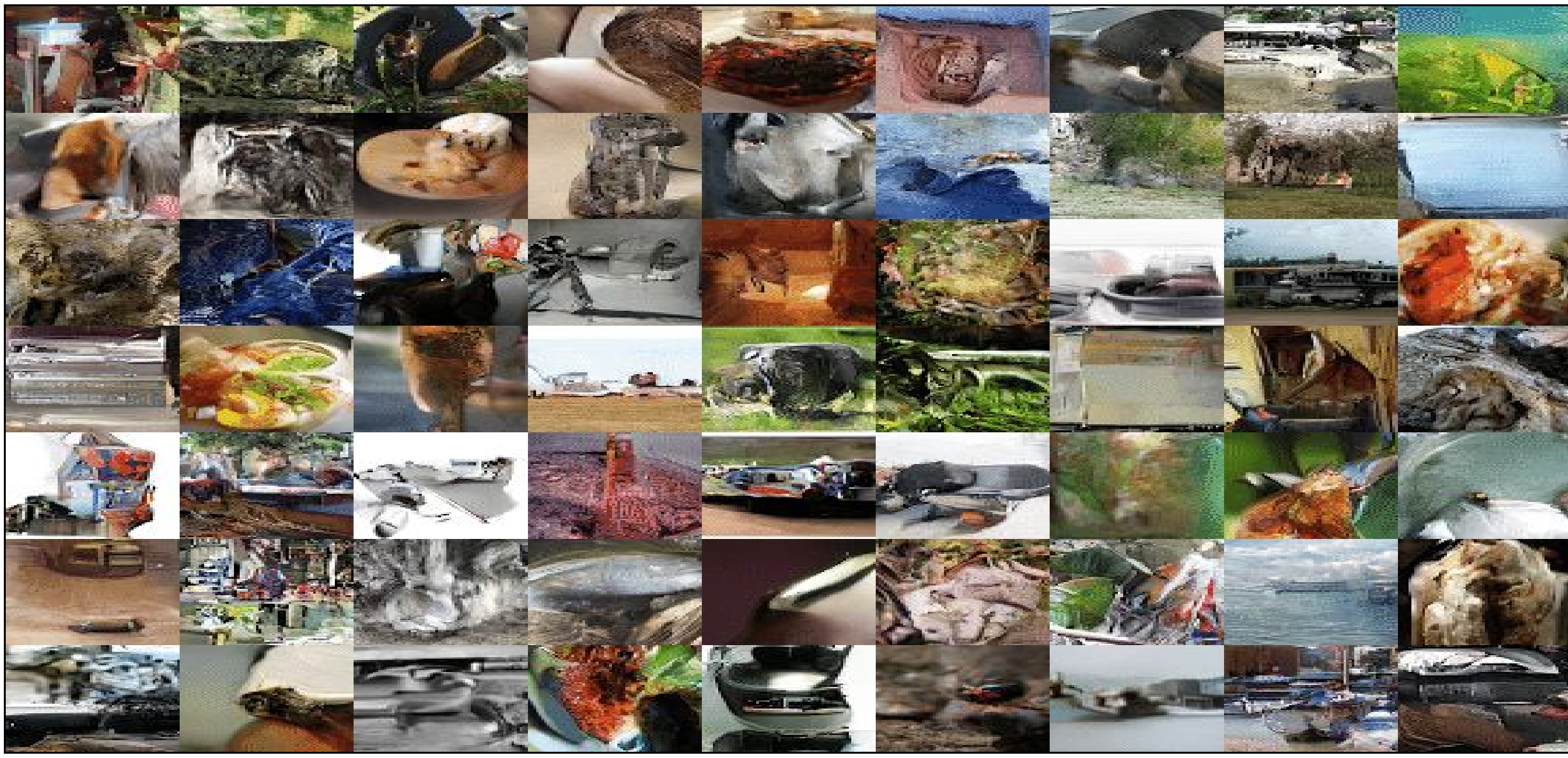


*Fig. 10: Sample Images for ImgeNet*

## 3. Poverty Estimation using Deep Learning & Transfer Learning

### a. Transfer Learning for Poverty Mapping:

In the approach followed here, a linear chain transfer learning graph is constructed with $V = \{P_1, P_2, P_3\}$ and $E = \{(P_1, P_2), (P_2, P_3)\}$. The first transfer learning problem is $P_1$ which is the object recognition on ImageNet, $P_2$ is predicting nighttime light intensity from daytime satellite images; while the third problem $P_3$ is predicting poverty using daytime satellite imagery.

ImageNet to Nighttime Lights ImageNet is an object classification image dataset of over 14 million images with 1000 class labels that, along with CNN models have fueled major breakthroughs in many vision tasks (Russakovsky et al. 2014). CNN models trained on the ImageNet dataset are recognized as a good generic feature extractor, with low-level and mid-level features such as edges and corners that are able to generalize to many new tasks (Donahue et al. 2013; Oquab et al. 2014). Our goal is to transfer knowledge from the ImageNet object recognition challenge ($P_1$) to the target problem of predicting nighttime light intensity from daytime satellite imagery ($P_2$). In $P_1$, we have an object classification problem with source domain data D1 = $\{(x_{1i}, y_{1i})\}$ from ImageNet that consists of natural images $x_{1i} \in X_1$ 1 and object class labels.

In $P_2$, we have a nighttime light intensity prediction problem with target domain data $D_2 = \{(x_{2i}, y_{2i})\})$ that consists of daytime satellite images $x_{2i} \in X_2$ and nighttime light intensity labels. Although satellite data is still in the space of image data, satellite imagery presents information from a bird's-eye view and at a much different scale than the object-centric ImageNet dataset *($P(X_1) \neq P(X_2)$)*. Previous work in domains with images fundamentally different from normal "human-eye view" images typically resort to curating a new, specific dataset such as Places205 (Zhou et al. 2014). In contrast, our transfer learning approach does not require human annotation and is much more scalable.

## b. Nighttime Lights to Poverty Estimation

The final and most important learning task $P_3$ is that of predicting poverty from satellite imagery, for which we have very limited training data. Our goal is to transfer knowledge from $P_2$, a data-rich problem, to $P_3$. The target domain data $D_3 = \{(x_{3i}, y_{3i})\}$ consists of satellite images $x_{3i} \in X_3$ from the feature space of satellite images of Rwanda and a limited number of poverty labels $y_{3i} \in Y_3$ . The source data is $D_2$, the nighttime lights data. Here, the input feature space of images is similar in both the source and target domains, drawn from a similar distribution of images (satellite images) from related areas (Africa and Rwanda), implying that $X_2 = X_3$, $P(X_2) \approx P(X_3)$. The source (lights) and target (poverty) tasks both have economic elements but are quite different.

### c. Final Model for Poverty Prediction:

Fully convolutional models have been used successfully for spatial analysis of arbitrary size inputs (Wolf and Platt 1994; Long, Shelhamer, and Darrell 2014). We construct the fully convolutional model by converting the fully connected layers of the VGG 16 network to convolutional layers. This allows the network to efficiently "slide" across a larger input image and make multiple evaluations of different parts of the image, incorporating all available contextual information. Given an unrolled $h \times w \times d$-dimensional input $x \,\varepsilon\, \Re^{hwd}$, fully connected layers perform a matrix-vector product

$$\widehat{x} = f(Wx + b)$$

where $W \,\varepsilon\, \Re^{k \times hwd}$ is a weight matrix, $b$ is a bias term, $f$ is a nonlinearity function, and $\widehat{x} \,\varepsilon\, \Re^{k}$ is the output. In the fully connected layer, we take $k$ inner products with the unrolled $x$ vector. Thus, given a differently sized input, it is unclear how to evaluate the dot products. We replace a fully connected layer by a convolutional layer with k convolutional filters of size $h \times w$, the same size as the input. The filter weights are shared across all channels, which means that the convolutional layer actually uses fewer parameters than the fully connected layer. Since the filter size is matched with the input size, we can take an element-wise product and add, which is equivalent to an inner product. This results in a scalar output for each filter, creating an output $\widehat{x} \,\varepsilon\, \Re^{1 \times 1 \times k}$ . Further fully connected layers are converted to convolutional layers with filter size $1 \times 1$, matching the new input $\widehat{x} \,\varepsilon\, \Re^{1 \times 1 \times k}$ . Fully connected layers are usually the last layers of the network, while all previous layers are typically convolutional. After converting fully connected layers to convolutional layers, the entire network becomes convolutional, allowing the outputs of each layer to be reused as the convolution slides the network over a larger input. Instead of a scalar output, the new output is a 2-dimensional map of filter activations.

In our fully convolutional model, the *400×400* input produces an output of size *2 × 2 × 4096*, which represents the scores of four (overlapping) quadrants of the image for *4096* features. The regional scores are then averaged to obtain a *4096*-dimensional feature vector that becomes the final input to the classifier predicting nighttime light intensity.

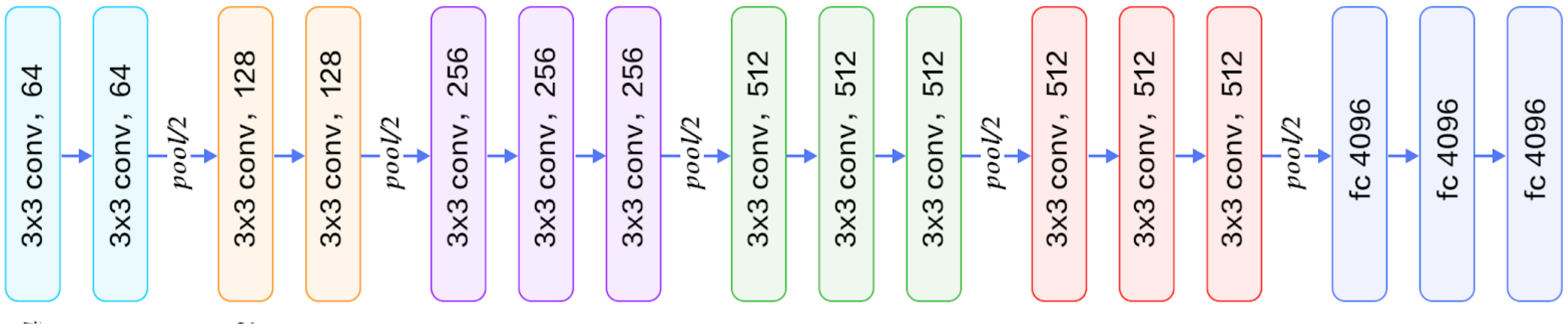


*Fig. 11: Usual version of VGG16*

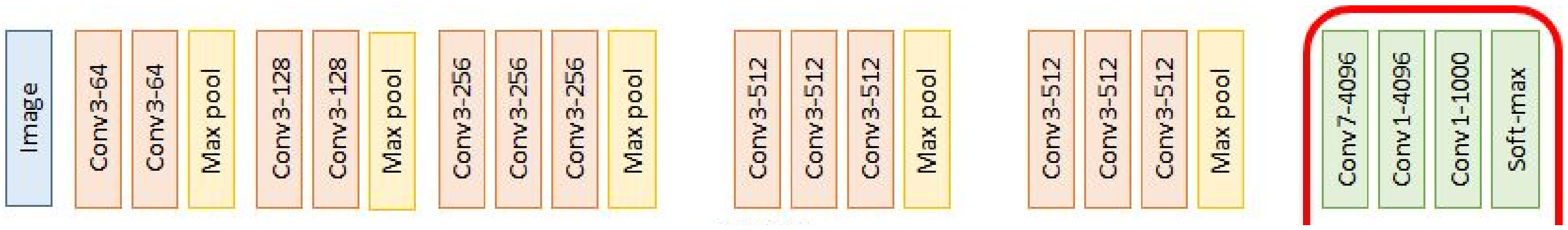
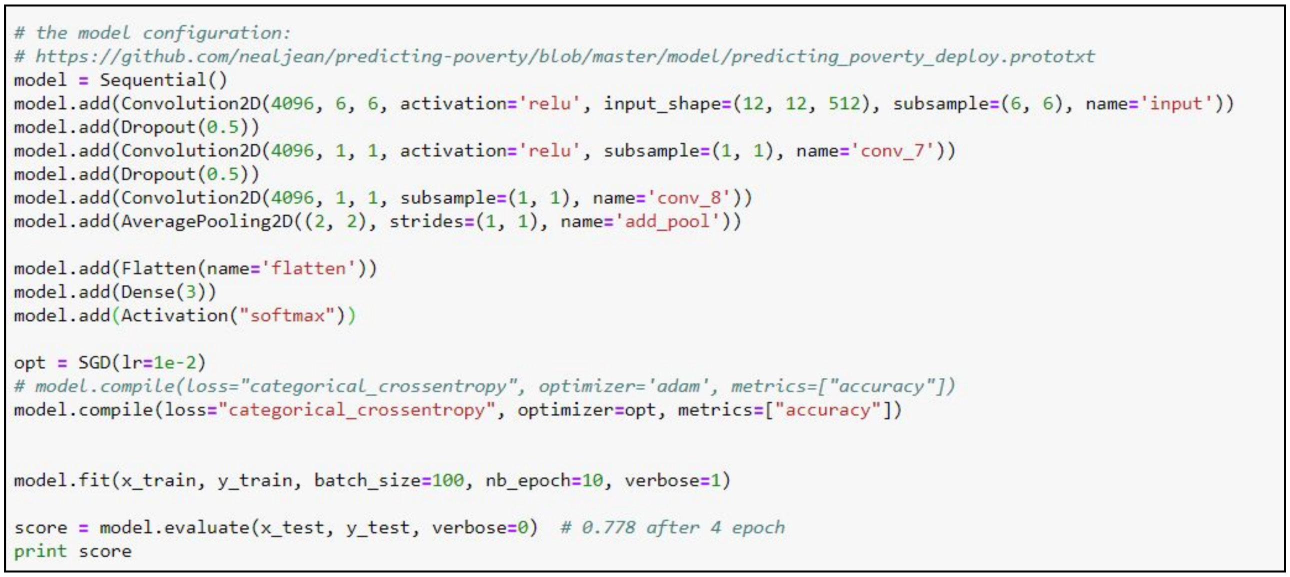


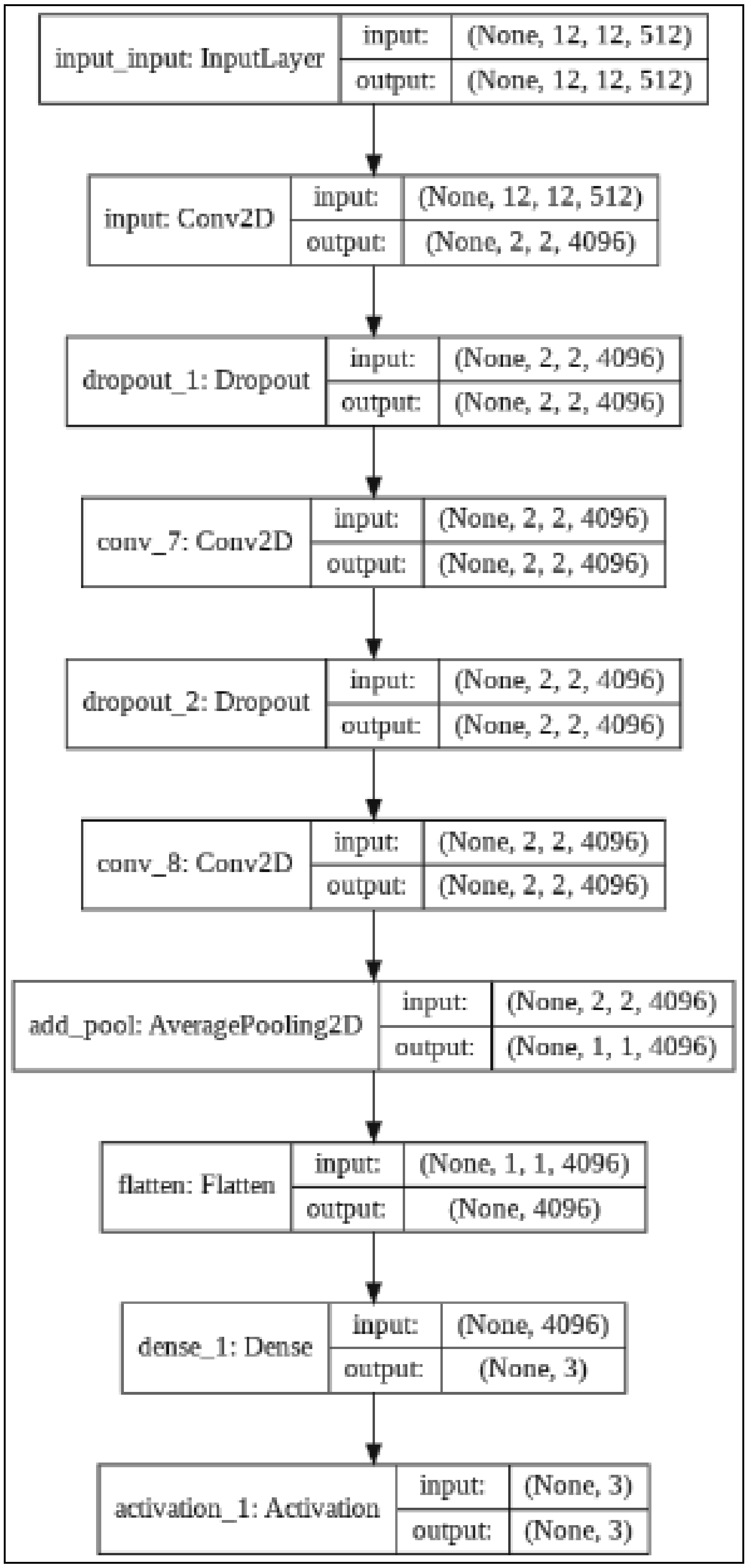
*Fig. 12: Fully Convolutional version of VGG16 (what we used)*

This feature vector is then given as input to the classifier predicting nighttime light intensity.
The structure of the final model is shown below:



```
# the model configuration:
# https://github.com/nealjean/predicting-poverty/blob/master/model/predicting_poverty_deploy.prototxt
model = Sequential()
model.add(Convolution2D(4096, 6, 6, activation='relu', input_shape=(12, 12, 512), subsample=(6, 6), name='input'))
model.add(Dropout(0.5))
model.add(Convolution2D(4096, 1, 1, activation='relu', subsample=(1, 1), name='conv_7'))
model.add(Dropout(0.5))
model.add(Convolution2D(4096, 1, 1, subsample=(1, 1), name='conv_8'))
model.add(AveragePooling2D((2, 2), strides=(1, 1), name='add_pool'))

model.add(Flatten(name='flatten'))
model.add(Dense(3))
model.add(Activation("softmax"))

opt = SGD(lr=1e-2)
# model.compile(loss="categorical_crossentropy", optimizer='adam', metrics=["accuracy"])
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=["accuracy"])


model.fit(x_train, y_train, batch_size=100, nb_epoch=10, verbose=1)

score = model.evaluate(x_test, y_test, verbose=0)  # 0.778 after 4 epoch
print score
```

*Fig. 13: Code Snippet 2*



input_input: InputLayer
input: (None, 12, 12, 512)
output: (None, 12, 12, 512)
input: Conv2D
input: (None, 12, 12, 512)
output: (None, 2, 2, 4096)
dropout_1: Dropout
input: (None, 2, 2, 4096)
output: (None, 2, 2, 4096)
conv_7: Conv2D
input: (None, 2, 2, 4096)
output: (None, 2, 2, 4096)
dropout_2: Dropout
input: (None, 2, 2, 4096)
output: (None, 2, 2, 4096)
conv_8: Conv2D
input: (None, 2, 2, 4096)
output: (None, 2, 2, 4096)
add_pool: AveragePooling2D
input: (None, 2, 2, 4096)
output: (None, 1, 1, 4096)
flatten: Flatten
input: (None, 1, 1, 4096)
output: (None, 4096)
dense_1: Dense
input: (None, 4096)
output: (None, 3)
activation_1: Activation
input: (None, 3)
output: (None, 3)

*Fig. 14: Model Structure*

## RESULTS & PERFORMANCE

### a. Comparison of Performance:

To evaluate the results of the proposed model, we compare it with three other models, two of which are proposed by Perez et. al. in his published work.

#### 1. VGG16 without Transfer Learning:

We simply trained the model with the same structure as discussed above but without pretraining it on the ImageNet data and then measured the performance.

#### 2. VGG-F with Transfer Learning:

In their work, Perez et. al. choose the VGG F model trained on ImageNet as the starting CNN model. The VGG F model has 8 convolutional and fully connected layers. Like many other ImageNet models, the VGG F model accepts a fixed input image size of 224 × 224 pixels. Input images in D2, however, are 400 × 40 pixels, corresponding to the resolution of the nighttime lights data.

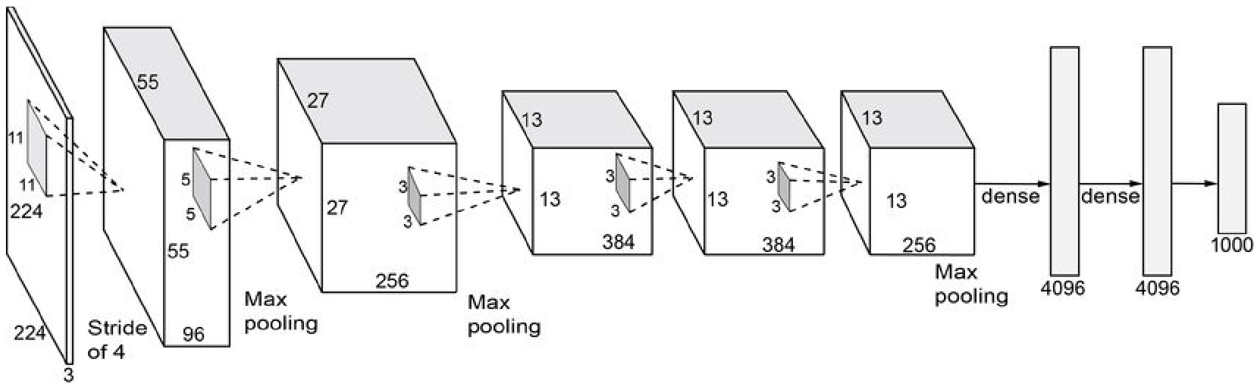


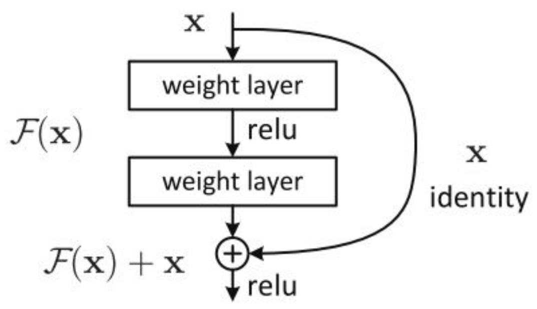
*Fig. 15: VGG-F Architecture*

3. **ResNet with Transfer Learning:**

Perez et. al. also used ResNet as one of their pretraining models to learn the initial weights from the ImageNet dataset.

<u>**About ResNet:**</u>

The core idea of ResNet is introducing a so-called "identity shortcut connection" that skips one or more layers, as shown in the following figure.


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

The authors (Zhang, Ren, Sun (2015)) argue that stacking layers shouldn' degrade the network performance because we could simply stack identity mappings (the layer that doesn't do anything) upon the current network, and the resulting architecture would perform the same. This indicates that the deeper model should not produce a training error higher than its shallower counterparts. They hypothesize that letting the stacked layers fit a residual mapping is easier than letting them directly fit the desired underlying mapping. And the residual block above explicitly allows it to do precisely that.

The complete architecture of ResNet looks like the following:

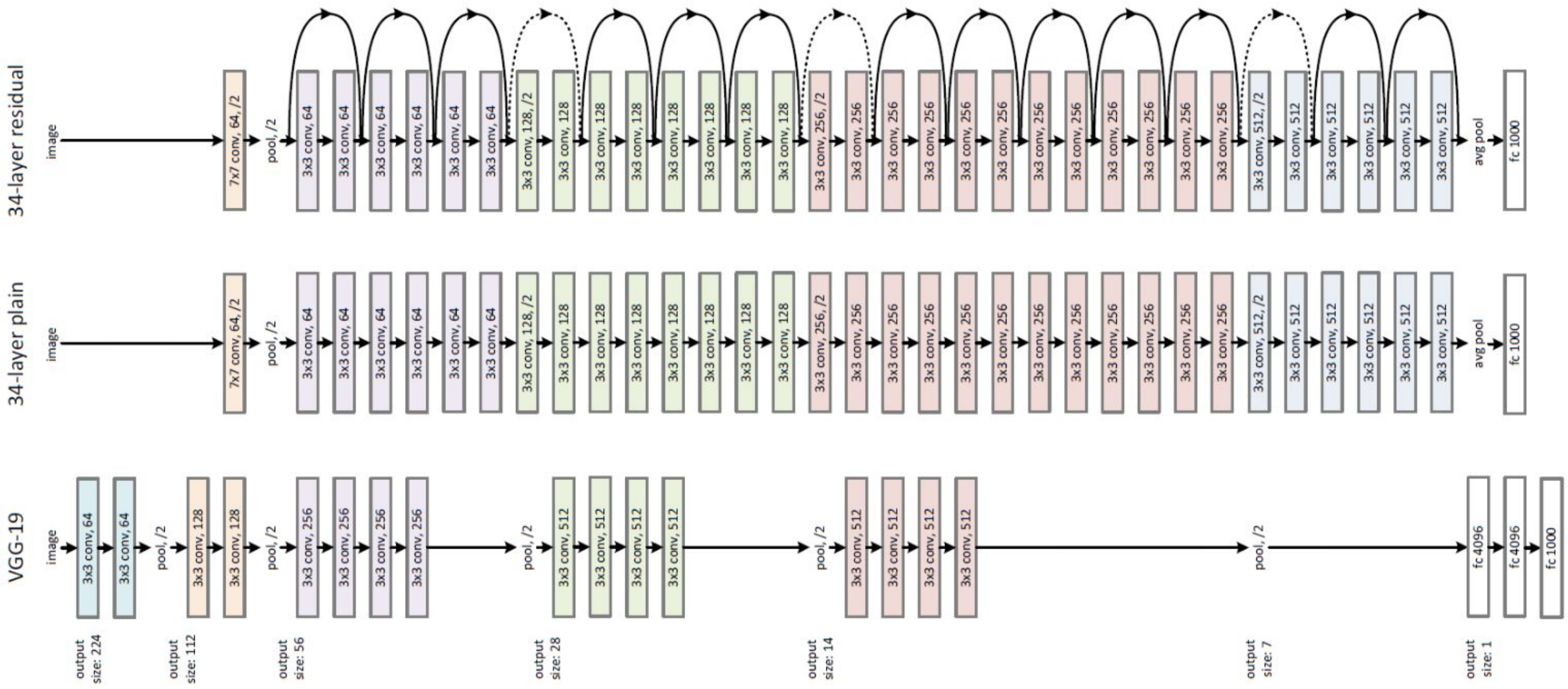


*Fig. 16: Resnet Architecture*

## b. Evaluation Metrics:

To measure the performance of each of the model, we have used 5 metrics of performance as follows:

- Accuracy (%)
- $r^2$ Score
- F1 Score
- Precision
- Recall

## c. Final Results:

We summarize the results of the whole analysis in the following table

| Model Name | Accuracy (%) | $r^2$ Score | F1 Score | Precision | Recall |
|---|---|---|---|---|---|
| **VGG16 with Transfer Learning** | 69.3% | 0.725 | 0.481 | 0.382 | 0.648 |
| **VGG16 without Transfer Learning** | 64.2% | 0.692 | 0.431 | 0.332 | 0.614 |
| **VGG-F with Transfer Learning** | 66.8% | 0.70 | 0.457 | 0.358 | 0.633 |
| **ResNet with Transfer Learning** | 71.2% | 0.74 | 0.484 | 0.386 | 0.650 |

*Table 1: Performance Evaluation*

At last, I prepared a heatmap based on the wealth predictions that we got from our model using the shapefile of Rwanda.

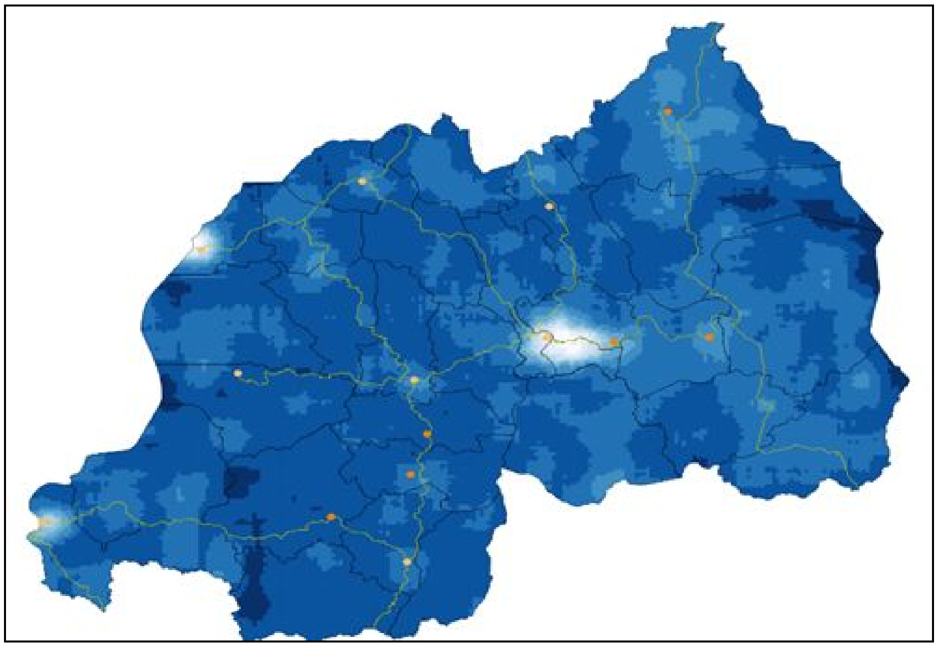


*Fig. 17: Heat Map of Wealth for Rwanda*

## CONCLUSIONS

From the analysis shown above, it can be clearly stated that:

- Convolutional Neural Networks (CNNs) works significantly, better than any other deep learning network as far as visual data is concerned, given the lesser processing power and processing time they consume to reach convergence and learn low-level as well high-level features.

- The only drawback of CNNs is that they need a very large amount of data to train and perform well. At the same time, data is a highly scarce resource in tasks like these. Thus, transfer learning is of high importance to learn features even with scarce data using similar data from some other source.

- Transfer Learning showed significant improvement in the results while predicting poverty of a region, using day-time satellite images, as compared to the CNN model which was not pre-trained for transfer learning.

- It is very much evident that modern-day deep learning techniques like Convolutional Neural Networks and Transfer Learning can be of tremendous help for policymakers especially in the cases of underdeveloped countries where the availability of primary data is scarce.

- The approach displayed here demonstrates that existing high-resolution daytime satellite imagery can be very useful for making fairly accurate predictions about the spatial distribution of economic well-being across developing countries where getting good quality primary data is a problem.

- The model performs very well despite inexact data on both the timing of the daytime imagery and the location of clusters in the training data, and more precise data in either of these dimensions are likely to further improve model performance. Thus, such a tool can be very handy for policymakers to drive and gauge the effects of their public policy measures.

## REFERENCES

[1] 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, 2016.

[2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[3] Perez, A., C. Yeh, G. Azzari, M. Burke, D. Lobell, and S. Ermon. 2017. "Poverty Prediction with Public Landsat 7 Satellite Imagery and Machine Learning." preprint arXiv:1711.03654.

[4] Xie M, Jean N, Burke M, Lobell D, Ermon S. Transfer learning from deep features for remote sensing and poverty mapping. In: Proceedings 30th AAAI conference on artificial intelligence. 2015. p. 1–10.

[5]. ] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575, 2014.

[6]. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. CoRR, abs/1310.1531.

[7]. K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385,2015.

[8]. Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks. In Proc. CVPR, 2014.

[9]. J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[10]. M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In ECCV, 2014.