# MAPPING POVERTY WITH SATELLITE IMAGERY

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
BACHELORS OF SCIENCE WITH HONORS

Michael Xie
May 2017

# Abstract

Poverty eradication is the first of 17 Sustainable Development Goals set by the United Nations to reach by 2030 [Jerven, 2014]. Measuring poverty is an important step to alleviating poverty, as it helps to inform research studies, target aid efforts, guide policy decisions, and generally monitor the progress of such an initiative. In the developing world, where the need for poverty data is the most pressing, poverty data is particularly scarce due to the resource cost associated with conducting surveys. This *data gap* is one of the crucial challenges to overcome in order to alleviate poverty. This thesis aims to cover a variety of methods towards closing this poverty data gap using remote sensing and satellite imagery data. We introduce a high-resolution poverty mapping method using only publicly available satellite data. The approach utilizes transfer learning to leverage knowledge from data-rich sources and combat the data gap. We also attempt to reduce the data gap by incorporating additional data, detailed in preliminary work using multiple resolutions of satellite imagery to improve predictive performance. We develop a semi-supervised method which narrows the data gap by utilizing abundant unlabeled satellite imagery. We show that can use this method to also take advantage of spatial correlations of poverty measures to improve our model predictions. Experiments are conducted on a variety of real-world datasets, as well as poverty measure prediction problems for 5 African countries – Malawi, Tanzania, Uganda, Nigeria, and Rwanda – demonstrating that our methods based on only publicly available data can approach the predictive performance of surveys conducted in the field and potentially transform efforts to track and alleviate poverty. Finally, we detail the implementation of a deployment pipeline system designed to support automated production of global scale poverty maps that is flexible enough to incorporate any dataset and model. This represents the first step towards providing up-to-date poverty maps to guide the decision-making process of nonprofit organizations and policymakers.

# Acknowledgments

I would like to thank my adviser Stefano Ermon, as well as Marshall Burke and David Lobell, for their guidance throughout this project. I would like to thank Neal Jean for his support and for being a great collaborator on much of this work. I would like to thank Matthew Davis for his help on acquiring, processing, and understanding the data. I would like to thank Jake Kim for his work and collaboration on extending the transfer learning method for multiple resolutions. Lastly, I would like to thank Christopher Yeh and Jason Liu for their hard work on the development and debugging of the deployment pipeline.

# Contents

**Abstract** ii

**Acknowledgments** iii

**1 Introduction** 1

**2 Transfer Learning from Data-Rich Domains** 4

2.1 Motivation . . . 5

2.2 Background . . . 5

2.2.1 Transfer Learning . . . 5

2.2.2 Convolutional Neural Networks . . . 6

2.2.3 Transfer Learning in Deep Learning Models . . . 7

2.3 Transfer Learning for Poverty Mapping . . . 7

2.3.1 ImageNet to Nighttime Lights . . . 8

2.3.2 Nighttime Lights to Poverty . . . 11

2.4 Experiments . . . 11

2.4.1 Nighttime Light Prediction step . . . 11

2.4.2 Visualization of Extracted Features . . . 12

2.4.3 Poverty Line Classification . . . 13

2.4.4 Consumption Expenditure and Asset Index Regression . . . 15

2.5 Using Multiple Resolutions . . . 16

**3 Semi-supervised Learning and Spatial Models** 18

3.1 Motivation . . . 19

3.2 Background . . . 20

3.2.1 Gaussian Processes . . . 21

3.2.2 Deep Kernel Learning . . . 21

3.2.3 Posterior Regularization . . . 22

3.3 Semi-supervised Deep Kernel Learning . . . 22

3.3.1 Posterior Variance Minimization . . . 22
3.3.2 Grounding in Posterior Regularization . . . 23
3.4 Experiments . . . 24
3.4.1 Baselines . . . 25
3.4.2 UCI Regression Datasets . . . 26
3.4.3 Visualizing the Feature Space . . . 27
3.5 Spatial Models through a Location Kernel . . . 28
3.5.1 Asset Index Prediction . . . 28

**4 Achieving Global Scale Mapping** **30**
4.1 Research and Deployment . . . 31
4.2 Deployment Pipeline Architecture . . . 31
4.2.1 Runner and Configuration . . . 32
4.2.2 Batcher . . . 33
4.2.3 Model . . . 34
4.2.4 Saver . . . 34
4.2.5 Rasterizer . . . 34

**5 Conclusions and Future Work** **36**

**Bibliography** **38**

**A Appendix** **44**
A.1 Proof of Theorem 1 . . . 44
A.2 Filter Activation Maps for Transfer Learning Method . . . 47
A.3 Other Spatial Models . . . 50
A.3.1 Problem Setup . . . 50
A.3.2 Weak Predictor Model . . . 51
A.3.3 Joint model . . . 52
A.3.4 Toy Segmentation Problem . . . 55

# Chapter 1

# Introduction

Poverty eradication is the first of 17 Sustainable Development Goals set by the United Nations to reach by 2030 [Jerven, 2014]. Although the number of people in extreme poverty conditions has been halved from 1990, almost a billion people are still scrambling to meet their basic needs [United Nations, 2015]. Measuring poverty is an important step to alleviating poverty, as it helps to inform research studies, target aid efforts, guide policy decisions, and generally monitor the progress of such an initiative. However, measuring poverty requires data on poverty. **In the developing world, where the need for poverty data is the most pressing, poverty data is particularly scarce due to the resource cost associated with conducting surveys to gather such data. This *data gap* is one of the crucial challenges to overcome in order to alleviate poverty.**

The data gap in Africa is particularly wide. According to data from the World Bank, only 39 of 59 countries conducted one or fewer nationally representative surveys from 2000 to 2010 [World Bank, 2015]. Of these 39, 14 countries did not conduct a single nationally representative survey. Similarly, for the Demographic and Health Surveys (DHS), the primary source for population-level health statistics and household asset wealth data in developing countries, 39 of 59 African countries conducted 1 or fewer nationally representative DHS asset-based surveys from 2000 to 2010. Of these 39, 20 had conducted no surveys [ICF International, 2015].

Novel data sources stemming from the growth of information technology, such as social media, mobile phone networks, or remote sensing and satellite imagery data could help to close this data gap. Remote sensing and satellite imagery data in particular are perhaps the only cost-effective technology able to provide a high-resolution view of predictive indicators for poverty at a global scale, as remote sensing technology and infrastructure continue to advance and grow. It is expected that commercial satellite sources will be able to provide daily global coverage at a sub-meter resolution at a fraction of the cost within 10 years [Murthy et al., 2014]. This wealth of data could provide a bridge for developing countries to cross the data gap. However, this data is highly unstructured; satellite imagery does not explicitly contain markers delineating the wealth of the area depicted without

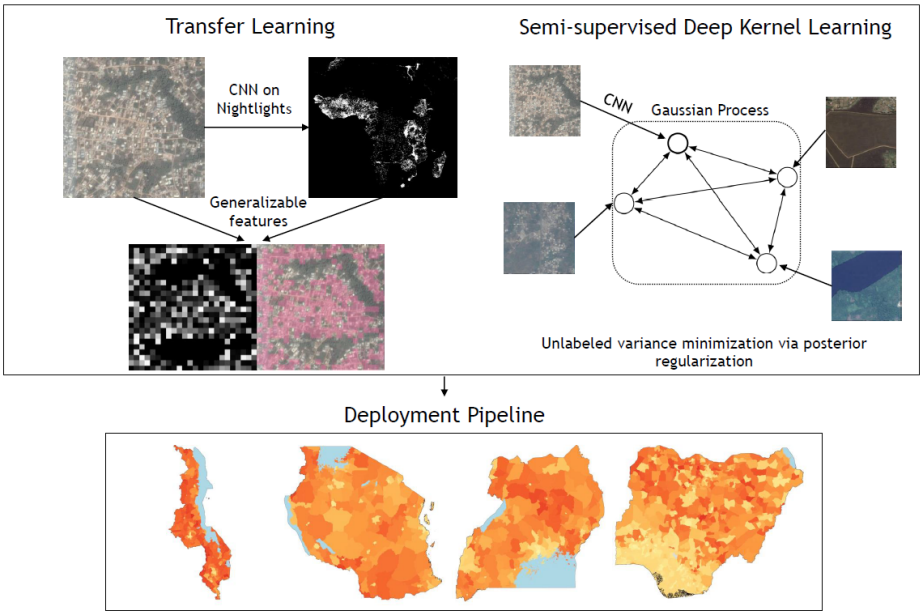


Figure 1.1: The transfer learning approach uses nighttime light data as a data-rich proxy for learning features generalizable to poverty estimation. Semi-supervised deep kernel learning can use satellite imagery without survey data directly through a novel semi-supervised objective minimizing posterior variance. Models and datasets are input into a flexible deployment pipeline that can provide up-to-date poverty maps to guide decision-makers.

intelligent processing. Therefore, machine learning approaches are needed to extract such insights from the unstructured data. In the past five years, deep learning approaches applied to large-scale datasets such as ImageNet have revolutionized the field of computer vision, leading to dramatic improvements in fundamental tasks such as object recognition [Russakovsky et al., 2014]. However, the major advances have mostly ridden on the coattails of an exponential growth in available data. In the poverty prediction problem, the available poverty data is inherently scarce, while the related unstructured data is abundant. **Fittingly, the data gap is therefore also the main challenge for the application of machine learning methods to the poverty mapping problem.**

This thesis aims to cover a variety of methods towards closing this poverty data gap using remote sensing and satellite imagery data. **In Chapter 2, we introduce a high-resolution poverty mapping method using only publicly available satellite data, as in our AAAI-16 conference and Science journal papers Xie et al. [2016], Jean et al. [2016a] (Nature Journal Highlight; Selected as World Changing Idea of 2016 by Scientific American)**. The approach utilizes transfer learning to leverage knowledge from data-rich sources and combat the data gap. This approach transfers knowledge from abundant high-resolution nighttime light intensity data as a proxy for economic development. We show that the transfer learning approach successfully learns features useful beyond nighttime light intensity prediction and applicable to

poverty measure prediction. For instance, the model learns filters identifying man-made structures such as roads, urban areas, and fields without any supervision beyond nighttime lights, i.e., without any labeled examples of roads or urban areas (Figure 2.3). **Remarkably, this approach finds satellite image indicators that can explain up to 75% of variation in local-level poverty measures in experiments using only publicly available survey and satellite data from Nigeria, Tanzania, Uganda, Malawi, and Rwanda, outperforming other remote sensing methods using proprietary data and even approaching the performance of survey data collected from the field.** This marks the first step in replacing the need for expensive surveys to make poverty estimations. We also attempt to reduce the data gap by incorporating additional data, detailed in preliminary work using multiple resolutions of satellite imagery to improve predictive performance.

In Chapter 3, we develop *semi-supervised deep kernel learning*, a semi-supervised regression method which narrows the data gap by utilizing the massive amounts of "unlabeled" satellite images for which we do not have corresponding survey data for that location, extending our 2016 NIPS workshop paper [Jean et al., 2016c, 2017]. The method is based on learning a structured feature embedding space that minimizes predictive variance of the unlabeled data. We learn a probabilistic model that quantifies the uncertainty of the model's predictions at unlabeled points by relating the unlabeled points to labeled points in the learned feature embedding space using a combination of a deep neural network and a Gaussian process. We ground the semi-supervised objective in the posterior regularization framework, a more flexible and direct way of controlling posterior distributions. **To our knowledge, this is the first application of posterior regularization to formulate semi-supervised learning objectives, which also generalizes earlier uncertainty minimization techniques in classification such as entropy minimization** [Grandvalet and Bengio, 2005, Zhao and Zhai, 2015]. We show that this model outperforms various other semi-supervised and supervised methods in a battery of real-world regression problems, as well as a poverty measure prediction problem. **We show that can naturally extend this method to also take advantage of spatial correlations of poverty measures to further improve our model predictions.**

**In Chapter 4, we detail the implementation of a deployment pipeline system designed to support automated production of global scale poverty maps that is flexible enough to incorporate any dataset and model.** As high-resolution satellite technology continues to develop and better predictive models are developed, this project closes the gap between model and data development to production of poverty maps. Being able to offer high-resolution poverty map data through publicly available data can potentially transform efforts to track and alleviate poverty. Together with the contributions presented in in the previous chapters, this represents the first step towards providing up-to-date high-resolution poverty maps to guide the decision-making process of nonprofit organizations and policymakers.

# Chapter 2

# Transfer Learning from Data-Rich Domains

This chapter introduces the transfer learning method of training a poverty measure predictor from satellite imagery [Xie et al., 2016, Jean et al., 2016a]. This approach seeks to close the data gap by using nighttime light intensity as a data-rich proxy for economic development. We describe a machine learning approach for extracting socioeconomic indicators from satellite imagery. In particular, we rely on the complex representation learning power of neural networks to find information in satellite images that can generalize over a sequence of transfer learning steps, from natural images to nighttime light intensity and finally poverty measures. We show that the transfer learning is successful in learning features relevant not only for nighttime light prediction but also for poverty mapping. For instance, the model learns filters identifying man-made structures such as roads, urban areas, and fields without any supervision beyond nighttime lights, i.e., without any labeled examples of roads or urban areas (see Figure 2.3). We show that this approach finds satellite image indicators that can explain up to 75% of variation in local-level economic outcomes and approaches the performance of survey data collected from the field with experiments using publicly available survey and satellite data from Nigeria, Tanzania, Uganda, Malawi, and Rwanda. This marks the first step in closing the poverty data gap in developing countries and for replacing the need for expensive surveys and censuses that have a limited spatial and temporal resolution.

Section 2.1 motivates the use of transfer learning to solve the data scarcity problem. Section 2.2 gives background on transfer learning and convolutional neural networks. Section 2.3 gives the methods for the transfer learning approach. Section 2.4 shows the experimental results on the five African countries. Section 2.5 shows initial results for incorporating imagery of multiple resolutions to enhance performance as in [Kim et al., 2016].

## 2.1 Motivation

There is a wealth of information in satellite imagery that can be exploited to produce insights to guide decision-making for organizations working to alleviate poverty. However, satellite data is an unstructured data source, and the poverty measure indicators present in the data are not obvious at once. Even with knowledge of all the correct indicators in the images to use, methods to extract the indicators would need to be individually developed. For example, the targeted cash transfer organization GiveDirectly has previously developed a method for extracting information about the number and types of roofs in an aerial image, as the general distinction between metal and thatched roofs is correlated with poverty measures [Abelson et al., 2014]. Recently, deep learning approaches such as convolutional neural networks (CNN) have revolutionized image processing and computer vision by automatically learning to extract indicators from the data. Fueled by large-scale datasets such as ImageNet, CNN approaches have dramatically improved core vision tasks such as object recognition [Russakovsky et al., 2014]. Although CNNs can be applied to satellite imagery, the poverty data gap in developing countries presents the primary challenge, as most current successes in CNN approaches have ridden on the coattails of large-scale datasets.

While there is a data gap in the poverty domain, we may be able to close this gap substantially if we can transfer the knowledge from a related domain that has a large amount of data. Transfer learning is a method of applying knowledge from one domain to help solve a problem in a different domain. A commonly used proxy for economic development available in developing countries at a high spatial resolution and at a global scale is the intensity of nighttime light for an area, which (at least) indicates development of lighting infrastructure. Therefore we hope to guide a CNN model towards learning indicators of economic development in general that can be transferred to build a predictor for poverty measures.

## 2.2 Background

### 2.2.1 Transfer Learning

Following Pan and Yang [2010], we give a formalism of transfer learning in which we ground our approach. A *domain* $\mathcal{D} = \{\mathcal{X}, P(\mathcal{X})\}$ consists of a feature space $\mathcal{X}$ and a marginal probability distribution $P(\mathcal{X})$. Given a domain, a *task* $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ consists of a label space $\mathcal{Y}$ and a predictive function $f(\cdot)$ which models $P(y|x)$ for $y \in \mathcal{Y}$ and $x \in \mathcal{X}$. Given a source domain $\mathcal{D}_S$ and learning task $\mathcal{T}_S$, and a target domain $\mathcal{D}_T$ and learning task $\mathcal{T}_T$, *transfer learning* aims to improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{T}_T$ using the knowledge from $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, $\mathcal{T}_S \neq \mathcal{T}_T$, or both. Transfer learning is particularly relevant when, given labeled *source domain data* $D_S$ and *target domain data* $D_T$, we find that $|D_T| \ll |D_S|$, so that the source domain has much more abundant data than the target domain.

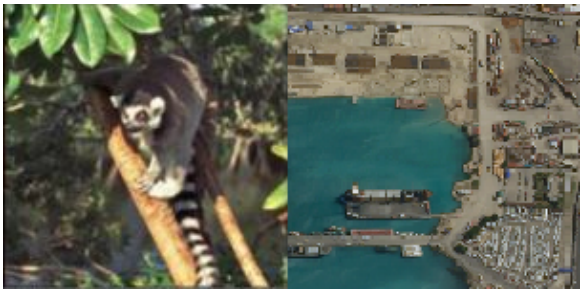

Figure 2.1: ImageNet data (left) versus satellite imagery data (right, Google Static Maps). High-level features such as faces and objects learned from ImageNet are not directly transferable to satellite imagery, but low-level features such as edges are present in both. We use the intermediate nighttime light intensity prediction problem to learn about the bird's-eye viewpoint of satellite imagery and extract features relevant to socioeconomic development.

In our setting, we have a sequence of transfer learning steps. We represent multiple source-target relationships as a *transfer learning graph*. Let a *transfer learning problem* $\mathcal{P} = (\mathcal{D}, \mathcal{T})$ be a domain-task pair. A transfer learning graph $G = (\mathcal{V}, \mathcal{E})$ is a directed acyclic graph where vertices $\mathcal{V} = \{\mathcal{P}_1, \cdots, \mathcal{P}_v\}$ are transfer learning problems and $\mathcal{E} = \{(\mathcal{P}_{i_1}, \mathcal{P}_{j_1}), \cdots, (\mathcal{P}_{i_e}, \mathcal{P}_{j_e})\}$ is an edge set. For each transfer learning problem $\mathcal{P}_i = (\mathcal{D}_i, \mathcal{T}_i) \in \mathcal{V}$, the aim is to improve the learning of the target predictive function $f_i(\cdot)$ in $\mathcal{T}_i$ using the knowledge in $\cup_{(j,i)\in\mathcal{E}}\mathcal{P}_j$.

We overcome the lack of poverty data by using two transfer learning steps. In the first step, we leverage ImageNet to extract features and representations that are useful for general image processing, which has been a successful strategy. Features from the `Overfeat` network trained on ImageNet for object classification achieved state-of-the-art results on tasks such as fine-grained recognition, image retrieval, and attribute detection [Razavian et al., 2014]. In our case, pre-training on ImageNet is useful for learning low-level features such as edges, but high-level features such as human faces would be less applicable to satellite imagery due to the difference in perspective of natural images from ImageNet and the aerial view of satellite imagery. In the second transfer learning step, we adapt the model to satellite image input and learn features simultaneously useful for predicting nighttime light intensities and poverty measures.

### 2.2.2 Convolutional Neural Networks

Deep learning approaches learn hierarchical representations of data through complex nonlinear transformations. Convolutional neural networks (CNN) are designed specifically for image input and involve convolution operations. Convolution operations encode translational invariance, an important aspect of image features [Bouvrie, 2006]. Translational invariance of image features is the property that moving the feature across the image does not affect the recognition of the feature. Intuitively, convolution operations slide filters across the image, and each filter is a template that matches

a particular image feature. Sliding the filter templates gives rise to the translational invariance property.

Deep learning models are deep in the sense that layers of operations are created via function composition. CNN models in particular have layers of convolution operations, where the initial layers of the CNN typically learn low-level image features such as edges and corners, while further layers learn higher-level features such as textures and objects [Zeiler and Fergus, 2013]. As a whole, the CNN model defines a mapping from input tensors (images) to feature vectors $f_\theta : \mathbb{R}^{w \times h \times d} \to \mathbb{R}^n$ where $\theta$ are the parameters to the CNN, $w \times h \times d$ are the dimensions of the input, and $n$ is the feature vector dimensionality. The feature vectors are then used as input to a final classifier or regression model to produce the predictions. Therefore, CNN models can be thought of as summarizing the complex image input into a feature vector for prediction. These features often represent complex compositions of the lower-level features extracted by the initial layers and can include textures and objects [Zeiler and Fergus, 2013, Le et al., 2012].

### 2.2.3 Transfer Learning in Deep Learning Models

The most direct method for transfer learning in deep learning models is to initialize the parameters in the training process of the target domain with the parameters learned from the source domain. The convolutional filters learned from the source domain may also be useful for the target domain, and the additional training process in the target domain helps to re-tune these filters for the new problem. For target problems with abundant data, the low-level filters in the initial layers can be transferred and new high-level features specific to the target problem can be learned. For target problems with a limited data, having sufficiently similar source and target domains can make it possible for the feature representation learned by the CNN to be transferred to the target problem without much additional tuning. Feature representations from CNN models trained on ImageNet have been directly used as generic image features effectively for a wide range of vision tasks [Donahue et al., 2013, Oquab et al., 2014].

## 2.3 Transfer Learning for Poverty Mapping

We construct a linear chain transfer learning graph with $\mathcal{V} = \{\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3\}$ and $\mathcal{E} = \{(\mathcal{P}_1, \mathcal{P}_2), (\mathcal{P}_2, \mathcal{P}_3)\}$. The first transfer learning problem $\mathcal{P}_1$ is object recognition on ImageNet [Russakovsky et al., 2014]. The second problem $\mathcal{P}_2$ is predicting nighttime light intensity from daytime satellite imagery. The third problem $\mathcal{P}_3$ is predicting poverty from daytime satellite imagery. Intuitively, we use the intermediate problem $\mathcal{P}_2$ to learn high-level features for aerial satellite imagery that are relevant to socioeconomic development.

### 2.3.1 ImageNet to Nighttime Lights

#### Problem Setup

In $\mathcal{P}_1$, we have an object classification problem with source domain data $D_1 = \{(x_{1i}, y_{1i})\}$ from ImageNet that consists of natural images $x_{1i} \in \mathcal{X}_1$ and object class labels. In $\mathcal{P}_2$, we have a nighttime light intensity prediction problem with target domain data $D_2 = \{(x_{2i}, y_{2i}\})$ that consists of daytime satellite images $x_{2i} \in \mathcal{X}_2$ and nighttime light intensity labels. Previous work on image domains fundamentally different from natural images typically curate a new, specific dataset for their problem, such as Places205 for scene classification [Zhou et al., 2014]. The transfer learning approach does not require humans to create a new dataset and is therefore much more scalable. Additionally, unsupervised approaches such as autoencoders are not guided in any way towards the final problem and may waste representation capacity learning irrelevant features. In the transfer learning approach, the nighttime light problem guides the network to learn features relevant to economic development.

The National Oceanic and Atmospheric Administration's National Geophysical Data Center (NOAA-NGDC) provides annual nighttime images of the world with 30 arc-second resolution, or about 1 square kilometer, through satellites deployed by the United States Air Force Defense Meteorological System (DMSP) [NOAA National Geophysical Data Center, 2014]. The light intensity values are processed over the entire year so that ephemeral light sources are not included. Light intensity values are normalized integers ranging from 0 to 63.

The daytime satellite imagery used to train the nighttime light prediction model are sampled near locations in the 2015 Demographic Health Survey (DHS). The DHS program conducts nationally representative surveys in Africa that focus mainly on health outcomes [ICF International, 2015]. The DHS offers one of the most geographically comprehensive surveys in Africa, and thus we use it to choose a representative set of locations of populated areas. Satellite images are downloaded using the Google Static Maps API, each with $400 \times 400$ pixels at zoom level 16. This roughly corresponds to 1 square km per pixel, matching the NOAA nighttime lights data. The initial sample consists of over 330,000 images, each labeled with an integer nighttime light intensity value. In order to simplify the task, we bin the data into three nighttime light intensity classes obtained by fitting a 1-dimensional mixture of Gaussians model to the relative frequencies of the nighttime light intensity values from our sample. The three classes semantically correspond to areas of near-zero, low, and high intensity respectively. By observing the histogram of nighttime light intensities in initial image samples around DHS locations, we find that there are three dominant modes of nightlight intensities, and the Gaussian mixture model provides a principled way of binning the classes. The resulting class bins consist of a class corresponding to near-zero intensity, a class corresponding to intensity in the 3-34 range, and a class corresponding to 35-63. Since there are many more near-zero intensity locations than the others, we upsample the other classes (and downsample the near-zero class) to create a balanced dataset. The 3-class balanced dataset consists of 150,000 training images and

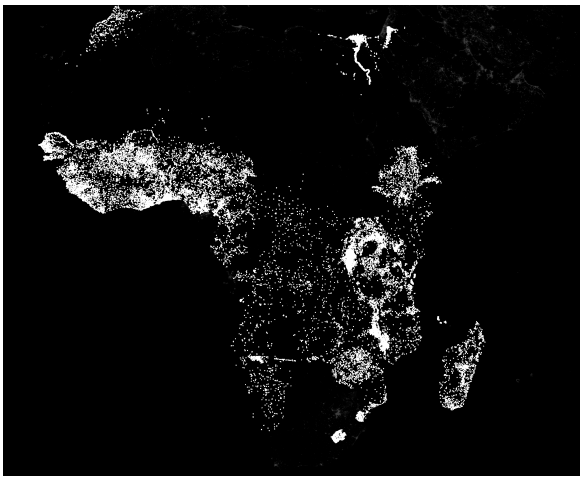

Figure 2.2: Locations (in white) of 330,000 sampled daytime images for the nighttime light intensity prediction problem.

8,000 validation images.

The Google Static Maps API does not directly provide temporal data with its images. However, each image has a watermark denoting the year in which the image was taken, and the images are generally between 2013-2015. Therefore, we use nighttime light intensities from 2013 (the most recent available from this source at the time) to minimize the temporal differences between the imagery and the intensity labels.

Note that, beginning in 2014, the NOAA-NGDC has provided a separate nighttime lights dataset collected from the Visible Infrared Imaging Radiometer Suite (VIIRS). The VIIRS data has 15 arc-second resolution, twice the resolution of the older DMSP data. However, the data has not been processed or aggregated annually to remove ephemeral light sources and also does not exist for many of the survey years in our current dataset. Although we do not use VIIRS data here, this new nighttime light data source can potentially increase the spatial resolution of the transfer learning method in the future.

### Fully Convolutional Model

Spatial context is important when designing a model for satellite imagery. For example, a crop field that is very close to an urban area may be more economically well off than a crop field that is very far from any urban areas. For this reason, satellite imagery can be much larger than natural images like those found in ImageNet. In our case, the satellite images are $400 \times 400$ pixels, which is significantly larger than the $224 \times 224$ images used in ImageNet. To still leverage knowledge from a model trained on ImageNet, we use the translational invariance property of the convolutional layers of the model. We build a fully convolutional model by converting any fully connected layers of the CNN into

convolutional layers. Fully convolutional models have been used successfully for spatial recognition and image segmentation [Wolf and Platt, 1994, Long et al., 2014]. A fully convolutional architecture allows the network to effectively slide the original model across the larger image and make multiple evaluations of different parts of the image, incorporating the spatial context and allowing for the model to handle images of any size. This is more efficient than evaluating the model multiple times over the larger image, since this only requires one forward pass and takes advantage of the heavily optimized convolution operation. The multiple resulting feature vectors are then averaged into one feature vector (an average pooling operation). Since fully connected layers are typically the last layers of the CNN, re-initializing these as convolutional layers allows for high-level features to be relearned. The parameters of remaining convolutional layers from the original model are reused so that lower-level features do not have to be relearned.

We convert fully connected layers to convolutional layers as follows. Given an unrolled $h \times w \times d$-dimensional input $x \in \mathbb{R}^{hwd}$, fully connected layers perform a matrix-vector product

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

where $W \in \mathbb{R}^{k \times hwd}$ is a weight matrix, $b$ is a bias term, $f$ is a nonlinearity function, and $\hat{x} \in \mathbb{R}^k$ is the output. In the matrix multiplication, there are $k$ inner products with the unrolled $x$ vector. 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 uses less 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 $\hat{x} \in \mathbb{R}^{1 \times 1 \times k}$. Further fully connected layers are converted to convolutional layers with filter size $1 \times 1$, matching the new input $\hat{x} \in \mathbb{R}^{1 \times 1 \times k}$. Instead of a scalar output, the new output is a 2-dimensional map of feature vectors.

For our task, we use a fully convolutional model converted from the `VGG F` model trained on ImageNet [Chatfield et al., 2014]. The `VGG F` model has 8 total convolutional and fully connected layers and accepts a fixed input image size of $224 \times 224$ pixels, which is smaller than our $400 \times 400$ input. In our `VGG F`-based model, the $400 \times 400$ input produces an output of size $2 \times 2 \times 4096$ that 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.

### 2.3.2 Nighttime Lights to Poverty

#### Problem Setup

We seek to transfer knowledge from the nighttime light prediction problem $\mathcal{P}_2$ to the poverty prediction problem $\mathcal{P}_3$. The target domain data $D_3$ consists of satellite images from the country or countries of interest. In our case, we are predicting poverty measures for five African countries. This implies that the source domain data, which are sampled across Africa, is roughly from the same distribution as the target domain data.

The poverty measure data $D_3$ comes from survey data collected from the field. We test the approach on two measures of poverty, average household consumption expenditures and asset indices. The average household consumption expenditures are taken from the Living Standards Measurement Study (LSMS) survey [Uganda Bureau of Statistics, 2012]. The asset index is a composite measure capturing the possessions of a household. Asset index data is taken from DHS survey data [ICF International, 2015]. All monetary measures are normalized for purchasing power parity (PPP) and converted to the US dollar using the exchange rate of the survey year.

We transfer knowledge from the nighttime light prediction problem to the poverty prediction problem by using the CNN model trained on nighttime light prediction as a feature extractor for satellite images. The features are then used as input to a logistic regression model for classification problems and a ridge regression model for regression problems, both of which are fit to poverty measure data.

## 2.4 Experiments

### 2.4.1 Nighttime Light Prediction step

We generally follow the hyperparameters from the `VGG F` model and decrease the initial learning rate [Chatfield et al., 2014]. The `VGG` model parameters are obtained from the Caffe Model Zoo and all networks are trained with Caffe [Jia et al., 2014]. The fully convolutional model is fine-tuned from the pre-trained parameters of the `VGG F` model, but randomly initializes the convolutional layers that replace fully connected layers.

We also consider random cropping as an alternative to a fully convolutional model to handle large images. In random cropping, we keep the fully connected layers of the original network and randomly crop the larger input to $224 \times 224$. This is a reasonable approach since the original model is trained by randomly cropping from a $256 \times 256$ input. By cropping, the random cropping model throws away over 68% of the input image when predicting the class scores, losing much of the spatial context. The random cropping model achieved a validation accuracy of 70.04% after 400,200 iterations. In comparison, the fully convolutional model achieved 71.58% validation accuracy after only 223,500 iterations, and both models were trained in roughly three days. Despite reinitializing

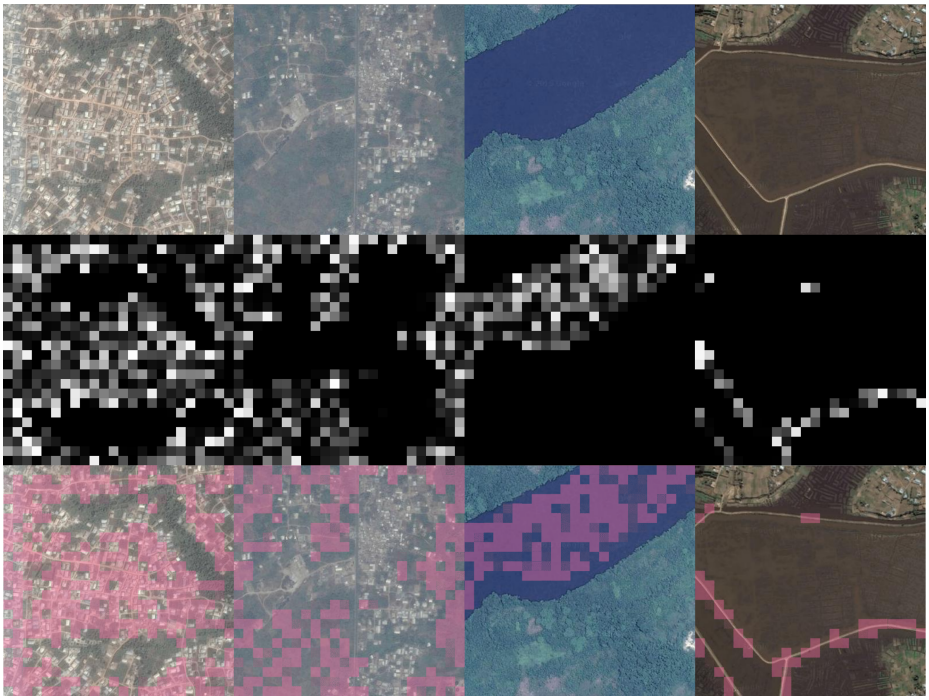

Figure 2.3: Example satellite imagery, filter activation maps, and overlays for 4 different filters (one per column). The first filter seems to activate for urban areas, the second activates for forested lands, the third activates on bodies of water, and the fourth activates for roads.

the final convolutional layers from scratch, the fully convolutional model exhibits faster learning and better performance. The final fully convolutional model achieves a validation accuracy of 71.71%, trained over 345,000 iterations.

### 2.4.2 Visualization of Extracted Features

We would like to have confirmation that training on the nighttime light intensity task guides the network to learn features that are generally applicable beyond predicting nighttime light intensity. The semantic meaning of convolutional filters is generally difficult to ascertain precisely. One method for peering into the black box is to analyze the filter activation maps for each filter, which are the outputs of the convolutions [Zeiler and Fergus, 2013]. High values in the activation map correspond to areas in the image where the image feature(s) that the filter has learned to match are present in the image. We find that many filters learn to identify semantically meaningful features such as roads, urban areas, barren land, forests, farmland, and bodies of water. These features are learned without direct supervision beyond nighttime light intensity. In contrast, other efforts to extract features from satellite imagery have relied on large amounts of expert-labeled data, such as labeled examples of roads [Mnih and Hinton, 2010, 2012]. These findings suggest that the transfer learning

| | Survey | ImgNet | Lights | ImgNet +Lights | Transfer |
|---|---|---|---|---|---|
| Accuracy | 0.754 | 0.686 | 0.526 | 0.683 | **0.716** |
| F1 Score | 0.552 | 0.398 | 0.448 | 0.400 | **0.489** |
| Precision | 0.450 | 0.340 | 0.298 | 0.338 | **0.394** |
| Recall | 0.722 | 0.492 | 0.914 | 0.506 | 0.658 |
| AUC | 0.776 | 0.690 | 0.719 | 0.700 | **0.761** |

Table 2.1: Cross validation test performance for predicting aggregate-level poverty measures. Survey is trained on survey data collected in the field. All other models are based on satellite imagery. Our transfer learning approach outperforms all non-survey classifiers significantly in every measure except recall, and approaches the survey model.

method may be able to generalize to the poverty estimation problem.

### 2.4.3 Poverty Line Classification

Our first task is to classify whether the average consumption expenditure in a cluster of households is over or under the international poverty line of $1.90 per person per day in Uganda.

We use data from the LSMS survey, which has data from 2,716 households in Uganda, which are grouped into 643 unique clusters. Only the average latitude and longitude location of the households within each cluster is given, with added noise of up to 5km in each direction in order to preserve anonymity of the individual households. Therefore, there is significant noise in the input that can reduce the performance of the model. In addition, each household has a binary poverty label based on expenditure data from the survey. We use the majority poverty classification of households in each cluster as the overall cluster poverty label. For each cluster, we sample approximately 100 1km$\times$1km images tiling a 10km $\times$ 10km area centered at the noisy average household location as input. This defines the probability distribution $P(\mathcal{X}_3)$ of the input images for the poverty classification problem $\mathcal{P}_3$.

We compare against baseline methods such as those using only the distribution of nighttime light intensity in the 10km $\times$ 10km grid as input features (Lights), using only a ImageNet-trained CNN as the feature extractor for satellite imagery (ImgNet), and concatenating nighttime light-derived features and ImageNet features (ImgNet+Lights). As a gold standard, we compare against the predictive performance of other data found in the survey. This data has two advantages over our satellite imagery. First, the data for each household is collected from the field, which ensures that the data is matched to an actual observation of a household. Since we have noisy locations, the satellite images may only be in the vicinity of the actual household. Second, the survey data contains information from looking inside the household and talking to the household members, something that satellites cannot do. We attempt to normalize this aspect by taking only survey fields that

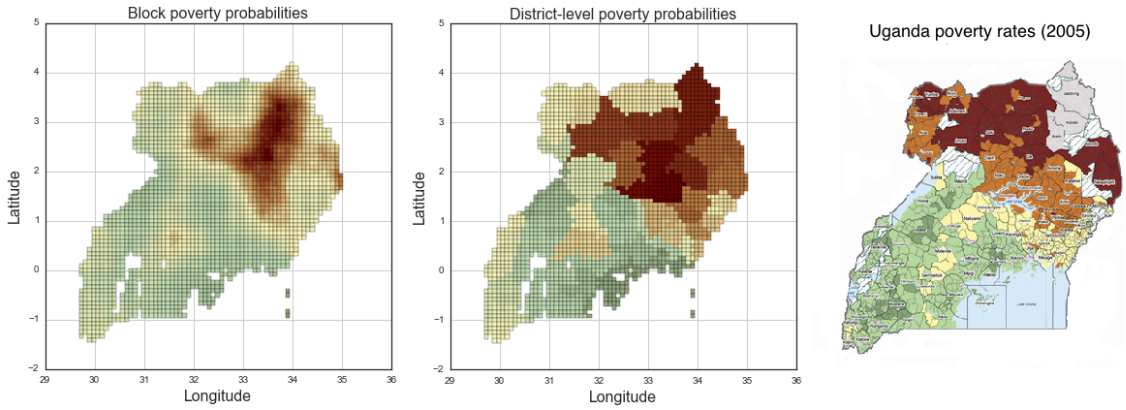


Figure 2.4: **Left:** Predicted poverty probabilities at a fine-grained 10km × 10km block level. **Middle:** Predicted poverty probabilities aggregated at the district-level. **Right:** 2005 survey results for comparison [World Resources Institute, 2009].

can possibly be detected from satellite imagery, such as the number of rooms in the house and the distance to the nearest population center. We train a logistic regression model using nested 10-fold cross validation, and the average results over 10 folds are reported in Table 2.1.

We find that the transfer learning step boosts performance beyond the information from the individual components (ImageNet and nighttime light intensity data). To understand the high recall of the nighttime light model, we note that the nighttime light model predicts "under" the poverty line nearly 100% of the time in cluster with near-zero average nighttime light, although only 51% of household clusters with near-zero average nighttime light are actually under the poverty line. However, only 6% of clusters in the two higher nighttime light intensity categories are below the poverty line, which explains the high recall of the lights model. In contrast, the transfer learning model predicts "under" the poverty line in 52% of clusters where the average nighttime light intensity is 0, more accurately reflecting the true distribution.

We can use this model to produce a poverty map of the probability of the average household expenditure in any 10km × 10km block to be under the international poverty line, seen in Figure 2.4. Since there are no such fine-grained poverty maps as ground truth to compare against, we qualitatively compare against the most recent available poverty rate map in Uganda, which is from 2005 and is reported on a district level [World Resources Institute, 2009]. Aggregating our results to the district level, we see the our generated map loosely corroborates with the map from 2005, notably that northern Uganda suffers from higher poverty rates [Ministry of Finance, 2014]. There are some differences between the maps that may be due to the 10 year temporal disparity or inaccuracies in either map. While current maps are very coarse and outdated, the transfer learning method can offer much finer spatial and temporal resolution at scale.

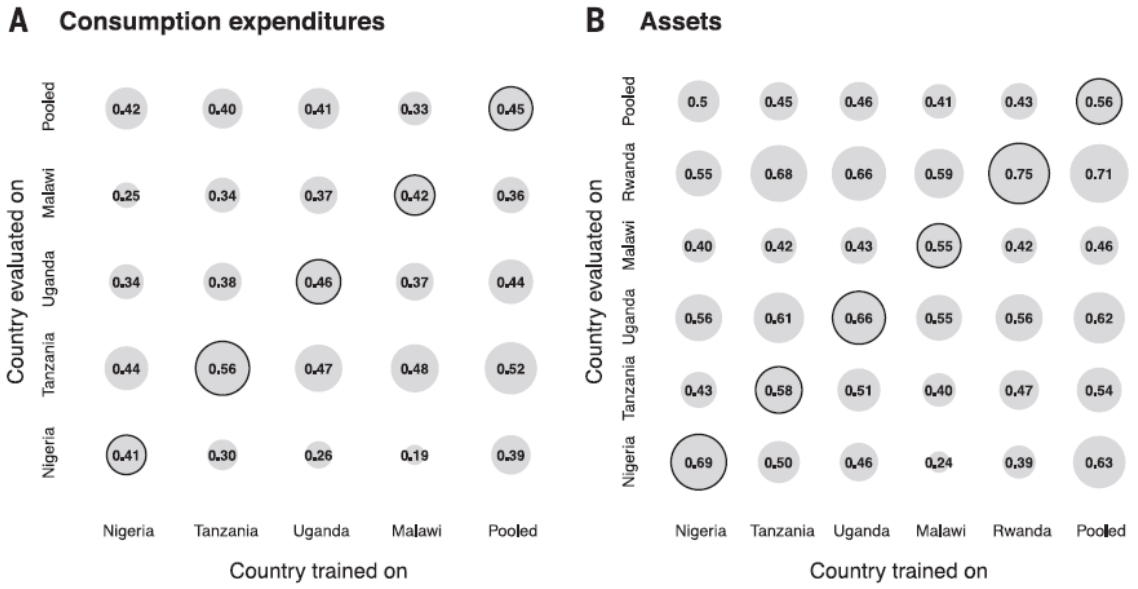


Figure 2.5: **Left:** Cross-border generalization $r^2$ of the transfer learning model for the consumption expenditure prediction problem. **Right:** Cross-border generalization $r^2$ for the asset prediction problem.

### 2.4.4 Consumption Expenditure and Asset Index Regression

We detail our further experiments on predicting real-valued consumption expenditure and asset indices, from the LSMS and DHS surveys respectively, as in [Jean et al., 2016a]. Asset indices are measures of material wealth computed as the first principal component of survey responses to questions about asset ownership [ICF International, 2015, Filmer and Pritchett, 2001]. We use the transfer learning CNN model as a feature extractor for the input images and fit a ridge regression model on the image features. All results are averaged over the outer 10 folds of a nested cross-validation scheme. We test this model on survey data from five African countries: Nigeria, Tanzania, Uganda, Rwanda, and Malawi. The countries were chosen because they have relatively recent survey data available. We also form a pooled dataset which combines the data from the five countries. We conduct experiments on all combinations of training and testing datasets, where the ridge regression model can either be trained and tested on the same country's data (in-country), or the cross-border generalization is evaluated by testing the ridge regression model on data from a different country from which it was trained (Figure 2.5). Lastly, we also evaluate the performance of training on a pooled dataset and testing the predictions of this model on poverty measures from individual countries.

For average household consumption expenditures, the ridge regression models trained in-country explain 37 to 55% of the variation. For average household asset wealth, the in-country models

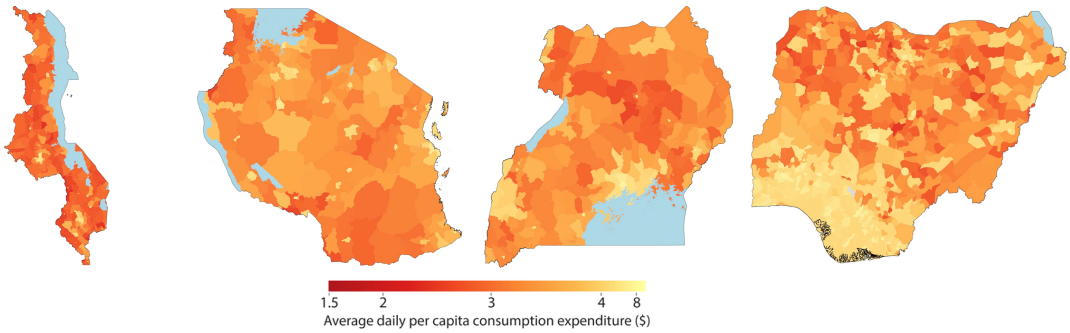


Figure 2.6: Estimated daily per capita consumption expenditures (2012-2015) [Jean et al., 2016b]. From left: Malawi, Tanzania, Uganda, Nigeria.

explain 55 - 75% of the variation. The pooled model explains 44 to 59% of the variation in both average household expenditures and assets. The models generally perform better on the asset index prediction task, possibly due to the more direct nature of the task; many assets may actually be possible to see from satellite imagery. The asset index measure may also be a better measure of long-run economic status, which may correlate more with landscapes and infrastructure that change slowly with time and are prevalent in satellite imagery [Filmer and Pritchett, 2001, Sahn and Stifel, 2003].

Similarly to the results from the poverty line classification problem, we find that despite learning to extract image features according to the nighttime light intensity prediction task, the transfer learning model achieves better performance than using nighttime light intensity as an input directly. Other baseline methods of feature extraction from images such as color histograms and histograms of oriented gradients (HoG) perform drastically worse than the transfer learning model [Jean et al., 2016a]. Amazingly, the transfer learning model has comparable performance to using data from past surveys to predict outcomes [Jean et al., 2016a]. In comparison to other remote sensing methods proposed such as using private cell-phone record data as in Blumenstock et al. [2015], our approach uses publicly available data, is more directly scalable due to the global scale of satellite imagery, and generates more accurate predictions.

We produce poverty estimation maps of consumption expenditures in Figure 2.6. The expenditure predictions are aggregated to the district level.

## 2.5 Using Multiple Resolutions

In the transfer learning method, we use satellite imagery from one resolution. There are often multiple resolutions of satellite imagery available for the same area, where lower resolutions allow for more spatial context around the area to be seen and higher resolutions can reveal more fine-grained details. Here, we detail some further work in progress on incorporating data with multiple

| Country | VGG | Zoom14 | Zoom16 | Zoom18 | End-to-end | Neural | Survey |
|---|---|---|---|---|---|---|---|
| Nigeria | 0.6917 | 0.7440 | 0.7684 | 0.7579 | 0.7747 | **0.7841** | 0.7648 |
| Tanzania | 0.5746 | 0.7084 | 0.6984 | 0.6974 | 0.7468 | **0.7615** | 0.6866 |
| Uganda | 0.6614 | 0.7226 | 0.7579 | **0.7610** | 0.7447 | 0.7517 | 0.8252 |
| Malawi | 0.5470 | 0.6027 | 0.6221 | 0.6039 | 0.6252 | **0.6380** | 0.8218 |
| Rwanda | 0.7438 | 0.7701 | 0.7732 | 0.7722 | 0.7786 | **0.7788** | 0.5870 |

Table 2.2: $r^2$ for in-country models on asset index prediction. The VGG models are trained following Xie et al. [2016]. The Zoom14, Zoom16, Zoom18 models are single-resolution ResNet-50 models. The End-to-end and Neural models are multi-resolution models. The Survey model does not use satellite imagery, and is built using features from the DHS surveys.

| Country | VGG | Zoom14 | Zoom16 | Zoom18 | End-to-end | Neural | Survey |
|---|---|---|---|---|---|---|---|
| Nigeria | 0.3968 | 0.4854 | 0.5623 | 0.5299 | 0.5702 | **0.5799** | 0.5686 |
| Tanzania | 0.4499 | 0.5001 | 0.5870 | 0.5870 | 0.6106 | **0.6365** | 0.6307 |
| Uganda | 0.5671 | 0.4298 | 0.5857 | **0.6325** | 0.6098 | 0.6218 | 0.7417 |
| Malawi | 0.4162 | 0.4093 | 0.4579 | 0.4374 | 0.4667 | **0.5019** | 0.5895 |
| Rwanda | **0.6216** | 0.5417 | 0.5799 | 0.5950 | 0.5971 | 0.6054 | 0.4860 |

Table 2.3: $r^2$ for out-of-country models on the asset index prediction problem.

resolutions into the model [Kim et al., 2016].

In this work, we use deep residual learning as in He et al. [2016] to train deeper CNN networks through the transfer learning method, with the intermediate problem being nighttime light intensity prediction to the VIIRS dataset, using images from 3 resolutions to make one prediction. We use zoom levels $14, 16, 18$, which correspond to resolutions of 9.55 m, 2.39 m, and 0.60 m per pixel respectively. Therefore, 224-by-224 pixel images cover areas of 2.14 km by 2.14 km, 0.54 km by 0.54 km, and 0.13 km by 0.13 km respectively. The VIIRS dataset is also binned into three classes, as in the DMSP dataset. Using multiple resolutions allows for both spatial context and smaller sub-meter objects such as cars to be seen. Indeed, when visualizing the features learned from the networks, the low resolution network finds large features such as bodies of water and landscape features, while the high resolution network finds features such as individual buildings and swimming pools. The three $224 \times 224$ images are inputted to 3 separate 50-layer ResNet, each of which produce a feature vector. These feature vectors are then concatenated into a single large feature vector. The test the in-country and cross-border generalization of two such models. One model (**End-to-End**) is trained end-to-end using backpropagation, using the concatenated vector as the input to the final nighttime light intensity softmax classifier. The second model (**Neural**) is trained by first fixing the three ResNet networks after training on the nighttime light prediction problem separately, then training a separate shallow neural network to learn how to combine the 3 feature vectors to predict nighttime light intensity. We find that both methods give a boost in the transfer performance of ridge regression models for the poverty prediction task.

# Chapter 3

# Semi-supervised Learning and Spatial Models

This chapter introduces a general method for training semi-supervised regression models through modeling structure in some learned feature space as in Jean et al. [2016c] and Jean et al. [2017]. Semi-supervised learning refers to the use of both data with labels and data without labels to train a predictive model. Problems in remote sensing typically have a surplus of remote sensing data with respect to the availability of the labels. In the poverty mapping problem, we have satellite imagery available across the globe, while survey data is typically scarce in developing countries. Making use of the abundant unlabeled satellite imagery, as in a semi-supervised method, is crucial in closing the data gap. Motivated by the spatial structure of poverty measures, we present a general semi-supervised method for learning regression models by learning a structured feature mapping using a deep Gaussian process model and optimizing a semi-supervised objective. Specifically, we augment maximum marginal likelihood learning with maximizing the posterior variance of predictions on unlabeled data through the posterior regularization framework [Zhu et al., 2014]. To our knowledge, this is the first application of posterior regularization to formulate semi-supervised learning objectives, which also generalizes earlier uncertainty minimization techniques in classification such as entropy minimization [Grandvalet and Bengio, 2005, Zhao and Zhai, 2015]. We show that this model can use unlabeled data to boost generalization performance in a set of real-world regression tasks and a local poverty measure prediction problem using satellite imagery.

Section 3.1 gives the motivation for this approach. Section 3.2 introduces background information on Gaussian processes, deep kernel learning, and the posterior regularization framework [Wilson et al., 2015, Zhu et al., 2014]. Section 3.3 introduces our semi-supervised regression model and grounds the model in the posterior regularization framework. Section 3.4 gives experimental results on real-world regression datasets and a poverty measure prediction problem. Section 3.5 introduces

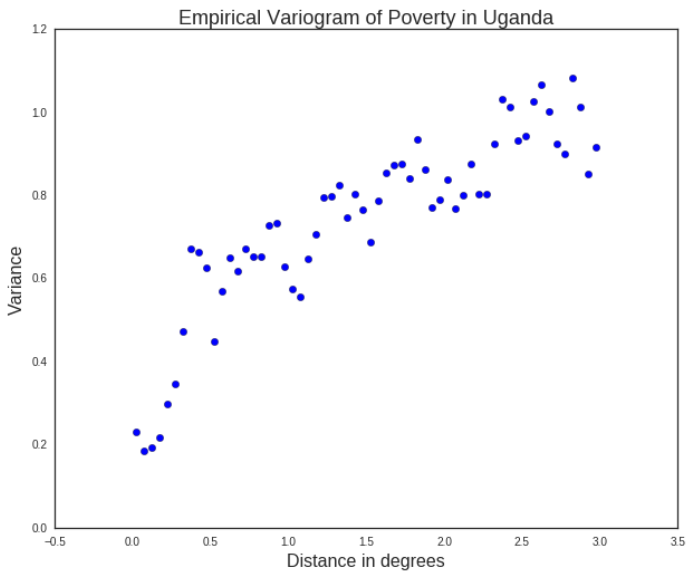


Figure 3.1: Empirical variogram of average household consumption expenditures in Uganda as in the 2012 LSMS survey. Households that are close together geographically tend to have similar levels of consumption expenditures.

some results on incorporating geographic coordinates into the model to take advantage of the spatial correlations in poverty measures.

## 3.1 Motivation

In the poverty mapping problem, we have access to satellite imagery for essentially any location in the world, but have survey data for only hundreds to thousands of households per country. We expect that the satellite imagery themselves are also an abundant source of information, even without accompanying survey data. In Chapter 2, we described a transfer learning method which uses the abundant unlabeled satellite imagery through learning a data-rich proxy task. However, finding proxy tasks that are both data-rich and are close enough to the final desired problem is hard in general. We seek a semi-supervised learning technique that can use both labeled and unlabeled data to boost the generalization performance of a model while training directly on the problem with limited labeled data.

As a motivating observation, we find by taking empirical variograms of poverty measure data from the LSMS surveys that households that are closer together geographically will tend to have more similar levels of consumption expenditures (Figure 3.1). This information can easily be added to the SSDKL model as prior knowledge to develop semi-supervised technique that takes advantage of the spatial correlation of poverty measures to relate labeled and unlabeled data.

We generalize this to arbitrary feature spaces using a deep Gaussian process model and the new semi-supervised objective of minimizing posterior variance. We use a deep learning model to learn a transformation of the input data into a feature space. We learn a Gaussian process model in this feature space, which provides both predictions and an estimate of prediction uncertainty as a probabilistic model. Using the fact that the model can quantify uncertainty in its predictions, we augment end-to-end maximum likelihood learning with posterior variance minimization in the unlabeled data.

Gaussian processes, as commonly used in machine learning, are powerful non-parametric models where pairwise covariance relationships are specified through a kernel function [Rasmussen and Williams, 2006, Ghahramani, 2015]. In geospatial statistics, Gaussian process regression is known as *kriging* [Haran, 2011, Cressie, 2015, Stein, 2012]. Interest in the connection between neural networks and Gaussian processes began when Neal [1996] showed that a Bayesian neural networks with an infinite number of hidden units converges to a Gaussian process. Deep Gaussian processes, as proposed by Damianou and Lawrence [2013], learn a hierarchy of Gaussian processes by modeling the outputs of a Gaussian process in one layer with another Gaussian process and continuing this pattern. Their model is used for unsupervised learning, trained using approximate variational inference, and does not integrate deep neural networks and Gaussian processes. Wilson et al. [2015] introduce the deep kernel learning method, which uses a deep neural network to learn a feature mapping and a Gaussian process to learn structure in the feature space. Their model is trained end-to-end with backpropagation, but is used for fully supervised learning.

Our semi-supervised objective of minimizing prediction uncertainty is motivated by transductive experimental design from the active learning community [Yu et al., 2006]. Transductive experimental design aims to choose the most informative set of training points (experiments) to use by seeking the points that have high predictive variance. In classification settings, prediction uncertainty methods such as minimum entropy and minimum variance regularization have been previously explored [Grandvalet and Bengio, 2004, Zhao and Zhai, 2015].

## 3.2 Background

In our semi-supervised problem setup, we will assume that we have a $d$-dimensional training set of $n$ labeled examples and $m$ unlabeled examples such that $\mathbf{x}_i \in \mathbb{R}^d, \forall i \in \{1, \ldots, n+m\}$ and there are $n$ real-valued targets $y_i \in \mathbb{R}, \forall i \in \{1, \ldots, n\}$. Let $X_L, \mathbf{y}_L, X_U$ be the aggregated training features and targets, where $X_L \in \mathbb{R}^{n \times d}$, $\mathbf{y}_L \in \mathbb{R}^n$, and $X_U \in \mathbb{R}^{m \times d}$. Let $X_T \in \mathbb{R}^{t \times d}, \mathbf{y}_T \in \mathbb{R}^n$ be aggregated testing features and targets.

### 3.2.1 Gaussian Processes

Gaussian processes are stochastic processes where the finite dimensional distributions are Gaussian distributed [Rasmussen and Williams, 2006]. In machine learning, the covariance matrix of these Gaussian distributions are specified using a covariance kernel function. A useful interpretation is that a Gaussian process defines a distribution over functions $f : \mathbb{R}^d \to \mathbb{R}$ from inputs to target values. Using the notation of Wilson et al. [2016b],

$$f(\mathbf{x}) \sim \mathcal{GP}\left(\mu(\mathbf{x}), k_\phi(\mathbf{x}_i, \mathbf{x}_j)\right)$$

with mean function $\mu(\mathbf{x})$ and covariance kernel function $k_\phi(\mathbf{x}_i, \mathbf{x}_j)$ parameterized by $\phi$. We have that a finite collection of function values are Gaussian distributed,

$$f(X) = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^T \sim \mathcal{N}(\boldsymbol{\mu}, K_{X,X}),$$

with $\boldsymbol{\mu}_i = \mu(\mathbf{x}_i)$ and $(K_{X,X})_{ij} = k_\phi(\mathbf{x}_i, \mathbf{x}_j)$. We can compute the posterior distribution at any set of points in closed form by standard Gaussian conditioning formulas on the observed data $(X_L, \mathbf{y}_L)$.

A commonly used kernel function is the squared exponential or radial basis function (RBF) kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi_f^2 \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\phi_l^2}\right)$$

where $\phi_f^2$ is the signal variance and $\phi_l^2$ is the characteristic length scale which specifies the extent of strong spatial correlations.

In practical applications, the observed data is noisy, such that $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon(\mathbf{x})$ for some function $f$ and noise function $\epsilon$. If we assume that $\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \phi_n^2)$, then the covariance kernel becomes $k(\mathbf{x}_i, \mathbf{x}_j) + \phi_n^2 \delta_{ij}$, where $\delta_{ij} = \mathbf{1}(i = j)$. For applications with spatially informative features such as location coordinates of satellite images, this information can easily be added in the form of a Gaussian process kernel.

### 3.2.2 Deep Kernel Learning

Wilson et al. [2015] introduces deep kernel learning, which uses a deep neural network to learn a feature mapping and simultaneously learns a Gaussian process on this feature space. Given input data $\mathbf{x} \in \mathcal{X}$, a neural network parameterized by $w$ is used to extract features $h_w(\mathbf{x}) \in \mathbb{R}^p$. The outputs are modeled as

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(h_w(\mathbf{x})), k_\phi(h_w(\mathbf{x}_i), h_w(\mathbf{x}_j)))$$

for some mean function $\mu(\cdot)$ and base kernel function $k_\phi(\cdot, \cdot)$ with parameters $\phi$.

All parameters of the model, including neural network weights and kernel parameters, are optimized end-to-end via backpropagation to maximize the marginal likelihood of the labeled data, such

that the loss is

$$L_{likelihood}(\theta) = -\log p(\mathbf{y}_L|X_L, \theta). \tag{3.1}$$

### 3.2.3 Posterior Regularization

The *posterior regularization* framework allows for direct manipulation of posterior distributions through changing the objective of the variational formulation of Bayes' rule. While knowledge is often encoded in a Bayesian prior, which reflects changes in the Bayesian posterior, one can directly encode posterior distributions via posterior regularization that may not be possible or straightforward as Bayesian posteriors [Ganchev et al., 2010].

Let $\mathcal{D} = (X_L, \mathbf{y}_L)$ be a collection of observed data. Zhu et al. [2014] present a regularized optimization formulation called *regularized Bayesian inference*, or RegBayes. In this framework, the regularized posterior is the solution of the following optimization problem:

$$\textbf{RegBayes:} \inf_{q(M|\mathcal{D})\in\mathcal{P}_{prob}} \mathcal{L}(q(M|\mathcal{D})) + \Omega(q(M|\mathcal{D})) \tag{3.2}$$

where $\mathcal{L}(q(M|\mathcal{D}))$ is defined as the KL-divergence between the desired post-data posterior $q(M|\mathcal{D})$ over models $M$ and the standard Bayesian posterior $p(M|\mathcal{D})$.

## 3.3 Semi-supervised Deep Kernel Learning

### 3.3.1 Posterior Variance Minimization

We propose a compound optimization objective that combines maximizing marginal likelihood of the observed data and minimizing predictive variance on the unobserved data, such that

$$L_{semisup}(\theta) = \frac{1}{n} L_{likelihood}(\theta) + \frac{\alpha}{m} L_{variance}(\theta)$$

$$L_{variance}(\theta) = \sum_{x\in X_U} \mathrm{Var}(f(x))$$

where $n$ and $m$ are the numbers of labeled and unlabeled training examples and $\alpha$ is a hyperparameter controlling the trade-off between supervised and unsupervised components.

Intuitively, the model should have the smallest predictive variance at labeled examples, since labels were given. Therefore, in order to reduce the predictive variance, the model will try to move the unlabeled examples closer to labeled examples in feature space, effectively clustering the data around labeled examples when possible. Moving an unlabeled example closer to a labeled example in feature space requires that the the model find commonalities between the original unlabeled and labeled examples and ignore some differences so that the mappings can converge. Therefore, the

variance minimization objective intuitively seeks generalizable features present in both labeled and unlabeled examples and relates the unlabeled data to labeled data according to these features.

Another interpretation is that reducing the variance is equivalent to reducing the expected squared error between the predictions $f(X_U)$ and the true unobserved labels $\mathbf{y}_U$ under the assumption that the finite dimensional distributions of the feature space learned by the network are indeed Gaussian. If this holds, then $f$ is an unbiased estimator of the true unobserved labels. Then we see that the sum of predictive variances on the unobserved data

$$\sum_{x \in X_U} \mathrm{Var}(f(x)) = \mathbb{E}_{q^*} \left[ \sum_{x \in X_U} (f(x) - \mathbb{E}_{q^*}[f(X_U)])^2 \right],$$

is the expected squared error of the predictions with respect to the posterior distribution $q^*$.

### 3.3.2 Grounding in Posterior Regularization

Let $X = (X_L, X_U)$ be the observed input data and $\mathcal{D} = (X, \mathbf{y}_L)$ be the input data with labels for the labeled data $X_L$. Let $\mathcal{F}$ denote a space of functions where for $f \in \mathcal{F}$, $f : \mathbb{R}^d \to \mathbb{R}$ maps from the inputs to the target values, and let $q(f|\mathcal{D}, \theta) \in \mathcal{P}_{prob}$ be a posterior distribution over the functions, where $\theta \in \Theta$ are parameters of $q$ from some parameter space $\Theta$ and $\mathcal{P}_{prob} = \{q : q(f, \theta|\mathcal{D}) = q(f|\theta, \mathcal{D})\delta_{\bar{\theta}}(\theta|\mathcal{D})\}$ is a family of distributions where $q(\theta|\mathcal{D})$ is restricted to be a Dirac delta centered on $\bar{\theta} \in \Theta$. We assume that a likelihood density $p(\cdot|X)$ exists and let $q^*(f, \theta|\mathcal{D}) = p(f|\theta, \mathcal{D})\delta_{\bar{\theta}}(\theta|\mathcal{D})$, where $p$ is the Bayesian posterior.

Instead of maximizing the marginal likelihood of the labeled training data in a purely supervised approach, we train our model in a semi-supervised fashion by minimizing the following compound objective:

$$L_{semisup}(\bar{\theta}) = -\frac{1}{n} \log p(\mathbf{y}_L|X, \bar{\theta}) + \frac{\alpha}{m} \sum_{x \in X_U} \mathrm{Var}_{f \sim q^*}(f(x)) \tag{3.3}$$

where $\alpha$ controls the trade-off between supervised and unsupervised components. In a Gaussian process, $p(\mathbf{y}_L|X, \theta) = p(\mathbf{y}_L|X_L, \theta)$.

This semi-supervised variance minimization objective can be viewed as a specific form of posterior regularization in the RegBayes framework [Zhu et al., 2014]. As in Zhu et al. [2014], we assume that $\mathcal{F}$ is a complete separable metric space and $\Pi$ is a absolutely continuous probability measure (with respect to background measure $\eta$) on $(\mathcal{F}, \mathcal{B}(\mathcal{F}))$, where $\mathcal{B}(\mathcal{F})$ is the the Borel $\sigma$-algebra, such that a density $\pi$ exists where $d\Pi = \pi d\eta$.

**Theorem 1.** *Given $\mathcal{D}$, a suitable space of functions $\mathcal{F}$, and parameter space $\Theta$, the semi-supervised variance minimization problem (3.3)*

$$\inf_{\bar{\theta}} L_{semisup}(\bar{\theta})$$

*is equivalent to the RegBayes optimization problem (3.2)*

$$\inf_{q(f,\theta|\mathcal{D})\in\mathcal{P}_{prob}} \mathcal{L}(q(f,\theta|\mathcal{D})) + \Omega(q(f,\theta|\mathcal{D}))$$

*where*

$$\Omega(q(f,\theta|\mathcal{D})) = \alpha' \sum_{i=1}^{m} \Big( \int_f (f(X_U)_i - \mathbb{E}_p[f(X_U)_i])^2 \cdot \int_\theta q(\theta|\mathcal{D}) p(f|\theta,\mathcal{D}) d\eta(f,\theta) \Big).$$

*where* $\alpha' = \frac{\alpha n}{m}$ *and* $\mathcal{P}_{prob} = \{q : q(f,\theta|\mathcal{D}) = q(f|\theta,\mathcal{D})\delta_{\bar{\theta}}(\theta|\mathcal{D})\}$ *is a family of distributions where* $q(\theta|\mathcal{D})$ *is restricted to be a Dirac delta centered on* $\bar{\theta} \in \Theta$*.*

A formal derivation of is included in the Appendix. It can be shown that solving the variational optimization objective

$$\inf_{q(f,\theta|\mathcal{D})} D_{KL}(q(f,\theta|\mathcal{D}) \| p(f,\theta|X)) - \int_{f,\theta} q(f,\theta|\mathcal{D}) \log p(\mathbf{y}_L|f,\theta,X) d\eta(f,\theta)$$

is equivalent to minimizing the first term of the RegBayes objective in Theorem 1, and the minimizer is precisely the Bayesian posterior $p(f,\theta|\mathcal{D})$. When we restrict $q \in \mathcal{P}_{prob}$, given any value of $\bar{\theta}$, the optimizing value is in the form $q^*(f,\theta|\mathcal{D}) = p(f|\theta,\mathcal{D})\delta_{\bar{\theta}}(\theta|\mathcal{D})$ where $p$ is the Bayesian posterior.

In general, it is possible for the optimal post-data posterior $q$ to be in a different form than the Bayesian posterior after adding the regularization term to the objective. However, note that the variance regularizer only depends on $q(\theta|\mathcal{D}) = \delta_{\bar{\theta}}(\theta|\mathcal{D})$. In this case, the optimal post-data posterior $q$ in the regularized objective is still of the form $q^*(f,\theta|\mathcal{D}) = p(f|\theta,\mathcal{D})\delta_{\bar{\theta}}(\theta|\mathcal{D})$, and $q$ is modified by the regularization function only through changing $\bar{\theta}$. From this we can recover the variance minimization objective. In the case where the Bayesian posterior is the posterior of a Gaussian process parameterized by parameters $\bar{\theta}$ (such as in SSDKL), this allows for modifying the posterior through optimizing the model parameters via closed-form differentiable inference formulas.

Since the labeled data are not separable in this model, we must use full batch gradient descent on the labeled data to compute exact gradients of the objective. However, since the sum of predictive variances are separable, we can use minibatched gradient descent for the unlabeled data. Previous work on using minibatched stochastic training has been explored using approximations to the inference procedure such as the use of sparse structured stochastic matrices [Wilson et al., 2016a]. This can also be used in our model to train the model in a fully minibatched fashion.

## 3.4 Experiments

We first introduce two other semi-supervised models we compare to as baselines, co-training regressors and virtual adversarial training (VAT). We then show results on 8 real-world regression tasks from the UCI repository, as well as a asset index prediction problem from satellite imagery.

In our results, we use the standard squared exponential or radial basis function (RBF) kernel.

We also experimented with polynomial kernel functions $k(\mathbf{x}_i, \mathbf{x}_j) = (\phi_f \mathbf{x}_i^T \mathbf{x}_j + \phi_l)^p$, $p \in \mathbb{Z}_+$, but they resulted in poorer performance for both Gaussian process-based models. To enforce positivity constraints and ensure positive definiteness of the covariance, we train the kernel parameter $\sigma_n$ in the log domain. The neural networks are regularized with L2 weight decay and models are trained in TensorFlow using the ADAM optimizer [Abadi et al., 2016, Kingma and Ba, 2015].

### 3.4.1 Baselines

Coreg, or co-training regressors, uses two $k$-nearest neighbor ($k$NN) regressors. During the training process, the regressors train and generate labels for the other regressor using its most confident predictions in alternating fashion [Zhou and Li, 2005]. Coreg ensures that the regressors do not have the same outputs by using different distance metrics in its regressors.

Virtual adversarial training (VAT) is a general training mechanism which enforces *local distributional smoothness* (LDS) by optimizing the model to be less sensitive to adversarial perturbations of the input [Miyato et al., 2015]. The VAT objective augments the marginal likelihood with an LDS objective:

$$\frac{1}{n}\sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \theta) + \frac{\lambda}{n}\sum_{i=1}^{n} \mathrm{LDS}(\mathbf{x}_i, \theta)$$

where

$$\mathrm{LDS}(\mathbf{x}_i, \theta) = -\Delta_{KL}(r^{(i)}_{\text{v-adv}}, \mathbf{x}_i, \theta)$$

$$\Delta_{KL}(r, \mathbf{x}_i, \theta) = D_{KL}(p(y_i|\mathbf{x}_i, \theta) \| p(y_i|\mathbf{x}_i + r, \theta))$$

$$r^{(i)}_{\text{v-adv}} = \arg\max_r \{\Delta_{KL}(r, \mathbf{x}_i, \theta); \|r\|_2 \leq \epsilon\}.$$

Note that the LDS objective does not require labels to compute virtual adversarial perturbations, so unlabeled data can be incorporated. The experiments in the original paper are for classification, although VAT is general. We use VAT for regression by choosing $p(y_i|\mathbf{x}_i, \theta) = \mathcal{N}(h_\theta(\mathbf{x}_i), \sigma^2)$ where $h_\theta : \mathbb{R}^d \to \mathbb{R}^H$ is a parameterized mapping (a neural network), and $\sigma$ is fixed. Optimizing the likelihood term is then equivalent to minimizing the squared error; the LDS term is the KL-divergence between the model's Gaussian distribution and a perturbed Gaussian distribution, which is also in the form of a squared difference. The adversarial perturbation $\Delta_{KL}$ is calculated with the second-order Taylor approximated at each step using a first dominant eigenvector calculation of the Hessian. The eigenvector calculation is done via a finite-difference approximation to the power iteration method. As in Miyato et al. [2015], one step of the finite-difference approximation was used in all of our experiments.

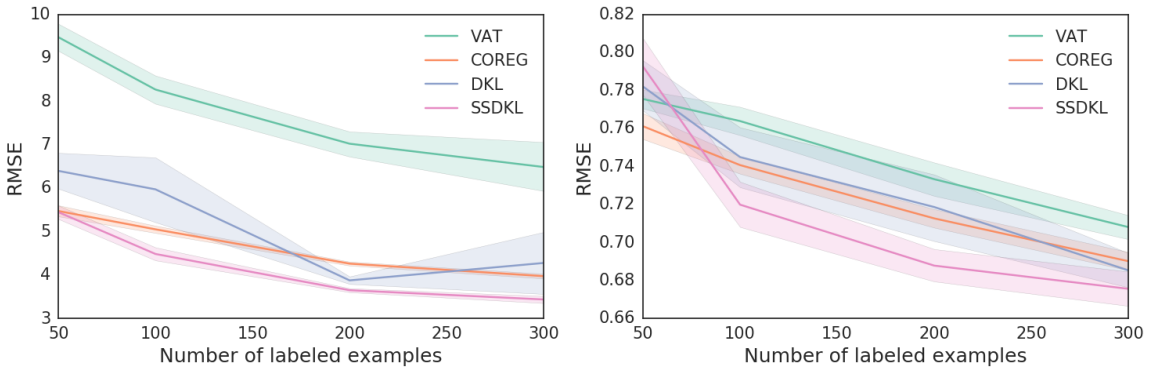


Figure 3.2: Average test RMSE vs. number of labeled examples, $n = \{50, 100, 200, 300\}$, for UCI Parkinsons **(left)** and Protein **(right)** datasets. Results are averaged over 10 trials of randomly sampled data and the shading reflects 1 standard deviation over the results distribution.

| | | | Percent reduction in RMSE | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | $n = 100$ | | | $n = 300$ | | |
| Dataset | $N$ | $d$ | SSDKL | Coreg | VAT | SSDKL | Coreg | VAT |
| Skillcraft | 3,325 | 18 | **7.7** | 7.5 | 1.4 | **1.7** | −6.4 | −7.0 |
| Parkinsons | 5,875 | 20 | **24.9** | 15.4 | −38.5 | **19.6** | 7.1 | −51.6 |
| Elevators | 16,599 | 18 | **1.9** | −6.7 | −3.5 | **1.8** | −8.5 | −10.7 |
| Protein | 45,730 | 9 | **3.4** | 0.6 | −2.5 | **1.4** | −0.7 | −3.3 |
| Blog | 52,397 | 280 | 3.7 | **8.6** | 8.4 | −0.3 | 6.8 | **7.3** |
| CTslice | 53,500 | 384 | 15.5 | **36.7** | 18.6 | 32.9 | **42.2** | 27.5 |
| Buzz | 583,250 | 77 | 13.7 | **24.8** | −2.4 | 15.7 | **17.9** | −21.3 |
| Electric | 2,049,280 | 6 | **43.7** | −17.5 | −120.6 | **17.8** | −76.8 | −241.6 |
| Average | | | **14.3** | 8.7 | −17.4 | **11.3** | −2.3 | −37.6 |

Table 3.1: Percent reduction in RMSE compared to baseline supervised deep kernel learning (DKL) model for semi-supervised deep kernel learning (SSDKL), Coreg, and virtual adversarial training (VAT) models. Results are averaged across 10 trials for each UCI regression dataset. Here $N$ is the total number of examples, $d$ is the input feature dimension, and $n$ is the number of labeled training examples. Final row shows average reduction in RMSE achieved by using unlabeled data.

### 3.4.2 UCI Regression Datasets

We evaluate SSDKL on eight regression datasets from the UCI repository. The number of labeled examples are varied between $n = \{50, 100, 200, 300\}$. For each $n$, we use 1000 examples as the hold out test set, and the remaining data is used as unlabeled examples. The resulting training set and validation set are split from the labeled examples, with 80% of the labeled examples used as training examples. For all results, we conduct 10 trials of randomly sampled data splits. The model is a small neural network with a [$d$-100-50-50-2] architecture, where the final feature embedding is two-dimensional, following Wilson et al. [2015]. Following Wilson et al. [2015], the DKL model

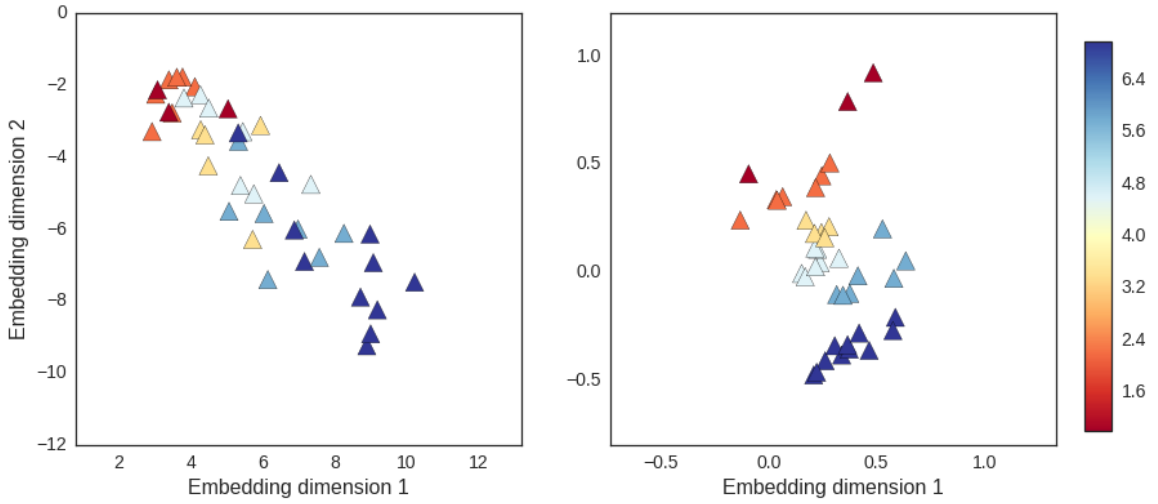


Figure 3.3: **Left:** Two-dimensional feature embeddings for labeled data learned by DKL on the Skillcraft dataset using $n = 50$ labeled training examples. **Right:** Embeddings learned by SSDKL using the same 50 labeled training examples plus an additional $m = 1000$ unlabeled examples. The addition of unlabeled data guides the model to learn a better structure for the labeled data, resulting in a desired gradient of color over the labeled data.

is initialized by sequential optimization of the neural network and the Gaussian process. For the SSDKL model, we search over $\alpha = \{0.1, 1, 10\}$, and model selection is done using the validation set. Full results can be found in Table 3.1. We use the same hyperparameters and initializations across *all* UCI datasets, showing the robustness of the approach. Specifically, we use learning rates of $1 \times 10^{-3}$ and 0.1 for the neural network and GP parameters respectively, and the GP parameters are initialized to the constant 1. As shown in Figure 3.2, SSDKL has marked advantages over DKL and baseline models especially in settings with lower labeled examples by using unlabeled examples effectively.

### 3.4.3 Visualizing the Feature Space

To gain some intuition about how the unlabeled data helps in the learning process, we visualize the neural network embeddings learned by the DKL and SSDKL models on the Skillcraft dataset. In Fig. 3.3 (**left**), we first train DKL on $n = 50$ labeled training examples and plot the two-dimensional neural network embedding that is learned. In Fig. 3.3 (**right**), we train SSDKL on the same $n = 50$ labeled training examples along with $m = 1000$ additional unlabeled data points and plot the resulting embedding. The addition of unlabeled data guides the model to learn a better structure for the labeled data, where points with a similar magnitude of labels are grouped together and not mixed, resulting in a desired gradient of color over the labeled data. When labeled data is scarce, complex models such as neural networks are prone to overfitting to the training set

and learning feature representations that fail to generalize well to unseen data. By incorporating unlabeled data and requiring that the learned features also try to minimize the predictive variance on these examples, the SSDKL model is able to use the unlabeled examples as a form of regularization to learn a more generalizable representation of the data. Using a nearest neighbors regressor with only the top nearest labeled neighbor in this embedding shows a reduction in MSE from 2.51 to 2.35 from the embedding learned by DKL to the SSDKL embedding.

## 3.5 Spatial Models through a Location Kernel

Although the SSDKL model is inspired by the spatial correlations in poverty measures, we do not directly make use of spatial coordinates in the above experiment. We observed previously (Figure 3.1) that poverty measures are indeed correlated over space such that households that are geographically closer together will be more likely to have similar measures of poverty. We can use location coordinates as additional inputs into the kernel function of the Gaussian process, which naturally incorporates locations into the model. Here, we test 2 plausible models for integrating features that are known to have spatial structure.

The Independent Kernel model has two independent Gaussian process priors, one which models the structure in the transformed image features and one which models the structure in the location coordinates. These Gaussian processes do not share parameters such as the length scale. These priors are then multiplied together to form the a new Gaussian process prior, which has a kernel which is the sum of the two independent kernels. This has the advantage of directly using the spatial structure of the location coordinates and having directly interpretable Gaussian process parameters. However, this may ignore any interactions between the location coordinates and the image data.

The Concatenated Kernel model allows for representation of interactions between the image features and the location coordinates. Such an interaction could occur, for example, for locations in the Sahara desert - many of the images in a certain latitude would depict harsh desert habitats. The image features are concatenated with the location coordinates and the vector is then normalized. In our experiments, we use static normalization by computing the statistics of the training set; however, batch normalization can also be used in this context. The concatenated vector is then transformed by a shallow neural network into the embedding space, in which the Gaussian process learns structure.

### 3.5.1 Asset Index Prediction

We apply the spatial SSDKL models, SSDKL, and DKL to a complex real-world task of predicting asset indices from satellite images in Africa, which are measures of material wealth. As described in Chapter 2, the dataset consists of $3,066$ household clusters across five African countries: Nigeria, Tanzania, Uganda, Malawi, and Rwanda. These countries include some of the poorest in the world

| | Percent reduction in RMSE ($n = 300$) | | | |
|---|---|---|---|---|
| Country | Independent Kernel | Concatenated Kernel | SSDKL | DKL |
| Malawi | 13.7 | 14.8 | **16.4** | 15.7 |
| Nigeria | **17.9** | 4.9 | 4.6 | 1.7 |
| Tanzania | 10.0 | 12.8 | **15.5** | 9.2 |
| Uganda | **25.2** | 12.6 | 12.1 | 13.8 |
| Rwanda | **27.0** | 26.7 | 25.4 | 21.3 |
| Average | **18.4** | 14.4 | 14.9 | 12.5 |

Table 3.2: Percent RMSE reduction compared to baseline supervised ridge regression model used in Jean et al. [2016a]. Final row shows average RMSE reduction of each model, where we find that the independent location kernel SSDKL model performs the best on average. Results are averaged over 10 trials.

(Malawi, Rwanda) as well as regions of Africa that are relatively better off (Nigeria). The raw satellite inputs consist of $400 \times 400$ pixel RGB satellite images downloaded from Google Static Maps at zoom level 16 as in Xie et al. [2016], Jean et al. [2016a]. In our experiments, we use a pre-trained VGG model trained on nighttime light intensity as an initial feature extractor Xie et al. [2016]. The image features are transformed in the Independent kernel, SSDKL, and DKL models by a shallow neural network with [$d$-100-50-50-2] architecture. The concatenated image features and location coordinates are transformed by a network with [($d$+2)-100-50-50-2] architecture. For 10 trials, we randomly sample 300 training examples, 1000 testing examples, 75 examples for the validation set, and the rest is used as unlabeled examples for semi-supervised methods and unused in DKL. We find that $\alpha \approx 1$ tend to result in the best performance, suggesting that hyperparameter tuning is not especially important for SSDKL models. We find that the Independent model outperforms other models in most of the experiments, and all the models outperform the ridge regression model from Chapter 2, as detailed in Table 3.2. This suggests that the geographic location is directly spatially informative and only loosely correlated with the image features.

# Chapter 4

# Achieving Global Scale Mapping

As satellite companies mature and public satellite data sources such as the Sentinel-2 satellite increase the spatial and temporal resolution of the imagery available, satellite data will evolve to become increasingly rich with information [European Space Agency (ESA) and United States Geological Survey (USGS), 2017]. Within ten years, it is expected that commercial sources will be able to provide daily to weekly global snapshots at a sub-meter resolution [Murthy et al., 2014]. Satellites are increasingly providing data for frequency bands outside the visible spectrum that may give extra information on reflectance, vegetation, and heat sources. Therefore, the remote sensing data that can be leveraged is always updating and changing, beckoning the development of new models to incorporate this data. While we have seen that machine learning models for the remote sensing and poverty mapping domain are still at the beginning of their development, there are already improvements upon current methods in terms of performance, cost-effectiveness, and scalability. We can expect that these models will continue to improve as more research is conducted.

The first outcome from this project is a way to estimate poverty measures with a high spatial resolution without a large survey effort. Simply providing a poverty map that is periodically updated with new data would be useful for many organizations in directing targeted aid efforts and policy decisions. This suggests a need for a system which makes the process of producing poverty maps as easy as possible, while being general and flexible enough to handle constantly evolving models and data sources. The ultimate goal is to be able to produce up-to-date poverty maps automatically as new satellite images are taken and models change, automatically downloading and processing the new satellite images, running the models, and uploading the results to a service such as Google Earth Engine. Here we present the first version of such a system, which is designed to handle large amounts of data and the rapid cycle of data and model development.

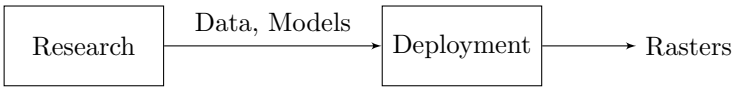


Figure 4.1: Research and Deployment as black boxes. Research is a human-in-the-loop process that produces new datasets (and ways to preprocess them) and models. Deployment takes these as input and automates the process of producing poverty map rasters as much as possible.

## 4.1 Research and Deployment

The high-level picture in Figure 4.1 describes interaction between two components, Research and Deployment. We can view Research as a black box which produces new predictive models and data sources.

Here, we focus on the implementation of the Deployment system, which takes the models and data produced by Research and creates a raster of poverty outcomes as the output. In order for the Deployment system to work on arbitrary models and data sources produced by Research, the notions of models and data must be generalized and fit within a set format. This is a common theme in many systems working with data transformations, such as MacroBase, where different components of the system must agree on a fixed communication format [Bailis et al., 2016]. As long as the inputs and output formats are obeyed, the internal implementations of each component of the system can be arbitrary.

## 4.2 Deployment Pipeline Architecture

The core architecture of the deployment pipeline consists of Batcher, Model, Saver, and Rasterizer modules. On a high level, the Batcher reads the input data (satellite images) and transforms the data into a format that the Model can use. The data is usually processed in small chunks or batches in order to process a large amount of data without running out of computer memory. The Model produces a batch of outputs (poverty outcomes) from the batch of input data. The Saver writes the outputs to disk and consolidates the batched outputs. Finally, the Rasterizer reads the saved output from disk and produces a raster or map by associating the outputs with their geographic locations.

The modules are abstract classes which specify the necessary functions that must be implemented, which standardizes the interface for these modules. The parameters, inputs, and outputs of the system are specified with a configuration text file. The modules are dynamically loaded and executed with a separate runner that processes the configuration file. Therefore, the pipeline is fully defined by the configuration file and the implementations of the Batcher, Model, and Saver modules. For use cases where the model does not change, such as applying the same model to data from multiple different countries at a time, the different jobs can be run by simply changing the configuration file, avoiding changes to the code as much as possible.

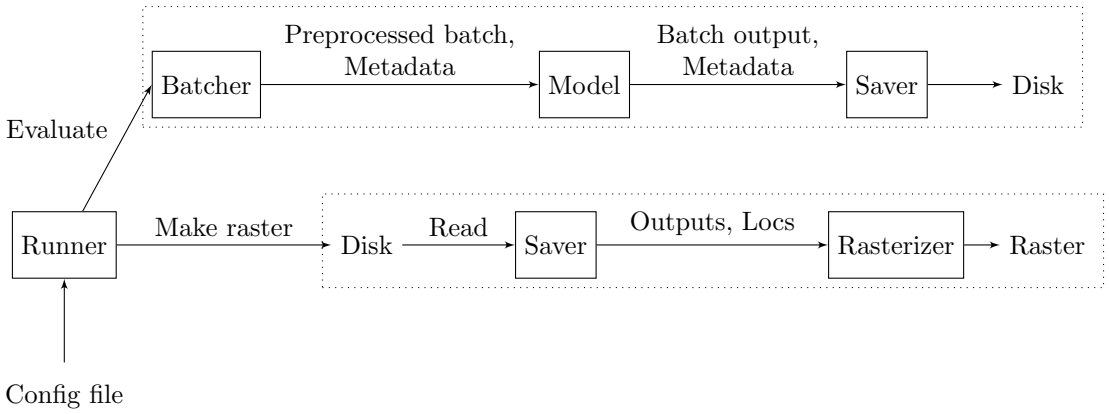


Figure 4.2: Deployment pipeline architecture. The configuration file specifies the implementations of each module (Batcher, Model, Saver, Rasterizer) to use, as well as any relevant parameters and paths. The runner uses this information to dynamically instantiate the pipeline. The runner evaluates the model in batch-wise fashion over the dataset and saves the outputs of the model to disk, aggregating the batch-wise outputs after finishing. The Saver is responsible for all disk I/O of model outputs. The Rasterizer then produces a raster from the results and locations saved on disk.

We choose TensorFlow as the main deep learning library for this project [Abadi et al., 2016]. TensorFlow has a rich set of features with active support and development that is quickly becoming an industry standard. TensorFlow also has developing capabilities for parallelism that can allow for further scaling in the future.

### 4.2.1 Runner and Configuration

The configuration file is a text file written in the YAML language, which provides a convenient way of specifying key-value pairs that can easily be loaded into a Python dictionary. The configuration file contains all the parameters for each of the Batcher, Model, and Saver modules, including which specific implementation of each module to use, the input and output paths, and additional parameters required by the modules. Below is an example configuration file:

```
batcher:
    name: poverty_pipeline.batchers.ImageBatcher
    args:
        img_mean:
            - 103.334
            - 107.8797
            - 107.4072
        data_files:
            -/path/to/malawi_replication.tfrecords
```

```
        batch_size: 32

model:
    name: poverty_pipeline.models.VGGFWithLinear
    path: /path/to/pretrained_weights
    args:
        pkl_path_model: /path/to/models/ridge.pkl
        pkl_path_scaler: /path/to/models/scaler.pkl

saver:
    name: poverty_pipeline.savers.OutputSaver2D
    args:
        out_dir: /path/to/malawi/features_dir

rasterizer:
    name: poverty_pipeline.rasterizers.RasterizerBasic
    args:
        out_dir: /path/to/malawi/rasters
```

For each module, the configuration file specifies the `name` of the specific implementation to use and the arguments to initialize the object under `args`. The runner uses this information to dynamically load and instantiate the modules. After loading the class using the class name, the runner matches entries from `args` with arguments to the constructor of the class. In this design, the system expects that the writer of the configuration file knows the arguments that must be passed to the particular implementation used.

### 4.2.2 Batcher

The Batcher is responsible for reading the data and transforming it into a format that can be used by the Model. As a running example, take the task of producing a poverty map over the entire continent of Africa using 3-meter resolution images and the transfer learning method described in Chapter 2. This dataset would consist of roughly 10TB of satellite images, which is too large to fit within random-access memory for most computers. The transfer learning model using the VGG-inspired convolutional neural network requires that the images must have each RGB pixel subtracted by a predefined mean RGB pixel. In this case, the Batcher reads the image data in batches, computes preprocessing steps on the batches such as subtracting the mean pixel, and produces batches of image data for the model.

The input image data that the Batcher reads from is standardized to use the TFRecords format, which is a binary file format developed as a part of TensorFlow. Records in the TFRecords file can

contain arbitrary metadata associated with the data. We use this format to store satellite images with their latitude-longitude coordinates, which are needed to produce the final map.

### 4.2.3 Model

The Model defines a transformation of a batch of input data to predicted outputs, such as poverty outcomes. In our running example, the model is a VGG-inspired convolutional neural network that was trained to predict nighttime light intensity using daytime satellite imagery, along with a ridge regression model on the features of the convolutional network which predicts a poverty outcome such as consumption expenditures. The parameters of the model are defined during the training process, which occurs in the Research process. These saved parameters are used to initialize the model. Our models are typically written using the TensorFlow library, although the implementation can be general. In this example, the convolutional network is implemented in TensorFlow while the ridge regression model is implemented using the Python machine learning library scikit-learn [Buitinck et al., 2013]. While the infrastructure for saving/loading saved model parameters is best used with the TensorFlow model saving format, other saved model parameters can be used. In the transfer learning model, we save the parameters for the ridge regression model separately as Python pickle files. An alternative method for integrating non-TensorFlow components is to save its parameters as TensorFlow variables and use the TensorFlow parameter loading routines to fetch the values.

### 4.2.4 Saver

The Saver handles input-output routines for saving the outputs from the Model to disk. As batches of outputs are created by the Model, the Saver intermittently writes the results to disk so that CPU memory will not be exhausted. Since the outputs are intermittently written, the outputs will be fragmented into different files on disk. After all the data has been processed by the Model, the Saver consolidates the saved output files by appending to a new file. In our running example, the Saver will ultimately produce a 3-column NumPy array with columns for latitude, longitude, and the poverty outcome prediction from the Model.

Since the Saver defines how the outputs are written on disk, it also provides the interface for loading the saved outputs. This reduces the communication needed between modules by eliminating the need for metadata on how to read the saved output files.

### 4.2.5 Rasterizer

The Rasterizer produces raster files, which are images represented directly as arrays of numbers. Using the Saver, the Rasterizer loads the saved outputs from disk. The Rasterizer then uses the location coordinates associated with the Model outputs to create a map of the outputs with the

desired map projection using geographic transformation routines from the Geospatial Data Abstraction Library (GDAL) [GDAL Development Team]. In our example, the Rasterizer outputs a raster with the same projection as the NOAA nighttime light intensity data [NOAA National Geophysical Data Center, 2014]. Rasters generated by the system have been verified to automatically reproduce the results that produce the maps in Chapter 2.

# Chapter 5

# Conclusions and Future Work

We have introduced a way to obtain high-resolution poverty maps using publicly available data in the transfer learning approach, refined this model through the our knowledge of the spatial correlations of poverty, and provided a first implementation of a system for having up-to-date poverty maps always available for the relevant organizations to use in their decision making process. The models obtain results that are competitive with those obtained from directly surveying the households in the field, and therefore using this method we can potentially save millions of dollars per survey effort to measure poverty.

While this initial work has been successful in giving a first method for attacking the poverty mapping problem, further work must be done to incorporate all the different sources of data available. Our initial work on using multiple resolutions of satellite imagery is a step in this direction (Chapter 2). Nighttime light intensity data, high-resolution population density data such as LandScan, and other spectral bands outside the visible spectrum that may give more information on vegetation or metallic reflectance can be valuable sources of data that are not used here [Bright et al., 2013]. Incorporating other data sources will necessitate new models, such as those that can train using multi-objective learning and convolutional neural network models that can handle arbitrary numbers of image bands effectively and efficiently.

Temporal data is the most important other source of data. While poverty mapping for the present is already useful, predicting the development of poverty would help decision-makers be one step ahead. Learning temporal trends also helps the model improve its current predictions. Overall, we have a complex spatio-temporal inference problem with interesting work in both dimensions.

Finally, a general problem with deep learning approaches is that the results are difficult to interpret. Models that can capture what exactly the model itself is looking at in the satellite imagery, such as attention-based models, may give a mechanism for better understanding the models [Mnih et al., 2014, Gregor et al., 2015]. At the same time, attention models can limit the amount of an image that a model must examine at a time to make an overall prediction. Since satellite imagery

can tend to be very large, this potentially allows for models that can process very large images to capture both spatial context and selected attention to detail.

Further development of the deployment pipeline will be aimed at increasing the reliability of the system and extending the two ends of the pipeline so that publishing a new poverty map is truly a one-click process. Production-grade systems are robust to failures and simply work. Due to a number of reasons, some step in the pipeline may fail. A fully robust system can recover from a failure automatically and continue where it left off. By virtue of processing data in small chunks and being built to handle very large datasets, the current system is also built to handle failure recovery very well. Since the outputs are saved intermittently, we can dump the output to disk when an error occurs and restart from where the system left off. The progress of the system can be saved as part of the TensorFlow architecture for checkpointing models during the training process. While we have succeeded in building a general pipeline from input images to poverty maps, there are still steps on either end of the current pipeline that would be better suited as part of the Development ecosystem. For example, consider the task of producing up-to-date poverty maps intermittently as satellites produce updated images without updating the model. In order to produce a poverty map, a researcher currently must download the satellite images manually, process the images, and convert the images into the TFRecords format. The image downloading and processing step is nontrivial for some image sources and may include tiling and mosaicking of images and producing cloud-free composites. After producing a poverty map, the raster must then be uploaded manually to a service such as Google Earth Engine. These are both tasks that can be automated, making them good candidates for incorporating into the Deployment framework. We see that this may easily blossom into a directed acyclic graph of data processing tasks with all require a large amount of computation time. AirFlow is a framework that may prove useful in this case, as it is specialized for scheduling jobs described as directed acyclic graphs, which communicate over some shared data store (such as local disk memory or cloud storage). AirFlow can run arbitrary bash commands as operators in its graph and automatically handles retries in case of a failure in any step. Encapsulating the current pipeline as well as any additional pre- and post-processing steps as AirFlow operators allows the entire process to be run end-to-end with reliability features built in.

Parallelism strategies in the deployment pipeline should also be explored to increase the scale at which poverty maps can be produced. So far, we have described a system which runs on one machine. Distributing the computation and taking advantage of parallel computing capabilities of large computing clusters can provide a boost in speed. We currently achieve *data parallelism* by splitting our data at the country level and processing different countries in different nodes in our cluster. Such a data parallelism strategy can easily be automated in a preprocessing operator. Other strategies for parallelism should be explored, including the capability for one model to run on multiple machines (*model parallelism*) using the distributed computation capabilities of TensorFlow.

# Bibliography


Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Brian Abelson, Kush Varshney, and Joy Sun. Targeting direct cash transfers to the extremely poor. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1563–1572. ACM, 2014.

Peter Bailis, Deepak Narayanan, and Samuel Madden. Macrobase: Analytic monitoring for the internet of things. *CoRR*, abs/1603.00567, 2016.

Joshua Blumenstock, Gabriel Cadamuro, and Robert On. Predicting poverty and wealth from mobile phone metadata. *Science*, 350(6264):1073–1076, 2015.

Jake Bouvrie. Notes on convolutional neural networks. 2006.

Edward Bright, Phillip Coleman, and Amy Rose. Landscan 2012 global population database, 2013.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

Noel Cressie. *Statistics for spatial data*. John Wiley & Sons, 2015.

Andreas C. Damianou and Neil D. Lawrence. Deep gaussian processes. *The Journal of Machine Learning Research*, 2013.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

European Space Agency (ESA) and United States Geological Survey (USGS). Sentinel-2 multispectral instrument (msi), 2017.

Deon Filmer and Lant H Pritchett. Estimating wealth effects without expenditure data or tears: An application to educational enrollments in states of india. *Demography*, 38(1):115–132, 2001.

Kuzman Ganchev, Jennifer Gillenwater, Ben Taskar, et al. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11(Jul):2001–2049, 2010.

GDAL Development Team. *GDAL - Geospatial Data Abstraction Library*. Open Source Geospatial Foundation.

Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553): 452–459, 2015.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2004.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 529–536. MIT Press, 2005.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1462–1471, 2015.

Murali Haran. Gaussian random field models for spatial data. *Handbook of Markov Chain Monte Carlo*, pages 449–478, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

ICF International. Demographic and health surveys (various) [datasets], 2015.

Neal Jean, Marshall Burke, Michael Xie, W Matthew Davis, David B Lobell, and Stefano Ermon. Combining satellite imagery and machine learning to predict poverty. *Science*, 353(6301):790–794, 2016a.

Neal Jean, Marshall Burke, Michael Xie, W. Matthew Davis, David B. Lobell, and Stefano Ermon. Sustainability and artificial intelligence lab: Combining satellite imagery and machine learning to predict poverty, 2016b. URL `sustain.stanford.edu/predicting-poverty`.

Neal Jean, Michael Xie, and Stefano Ermon. Semi-supervised deep kernel learning. *NIPS 2016 Bayesian Deep Learning Workshop*, 2016c.

Neal Jean, Michael Xie, and Stefano Ermon. Semi-supervised deep regression through posterior regularization. *In submission at Neural Information Processing Systems (NIPS)*, 2017.

Morten Jerven. Benefits and costs of the data for development targets for the post-2015 development agenda. *Data for Development Assessment Paper Working Paper, September. Copenhagen: Copenhagen Consensus Center*, 2014.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *CoRR*, abs/1408.5093, 2014.

Jae Hyun Kim, Michael Xie, Neal Jean, and Stefano Ermon. Incorporating spatial context and fine-grained detail from satellite imagery to predict poverty. Technical report, Stanford University, 2016.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *3rd International Conference for Learning Representations*, 2015.

Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. In *International Conference on Machine Learning*, 2012.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

Ministry of Finance. Poverty status report 2014: Structural change and poverty reduction in uganda, Nov 2014.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

Volodymyr Mnih and Geoffrey E. Hinton. Learning to detect roads in high-resolution aerial images. In *Computer Vision–ECCV 2010*, pages 210–223. Springer, 2010.

Volodymyr Mnih and Geoffrey E. Hinton. Learning to label aerial images from noisy data. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 567–574, 2012.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and koray kavukcuoglu. Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2204–2212. Curran Associates, Inc., 2014.

Kiran Murthy, Michael Shearn, Byron D. Smiley, Alexandra H. Chau, Josh Levine, and Dirk Robinson. Skysat-1: very high-resolution imagery from a small satellite. In *SPIE Remote Sensing*, pages 92411E–92411E. International Society for Optics and Photonics, 2014.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. ISBN 0387947248.

NOAA National Geophysical Data Center. F18 2013 nighttime lights composite, 2014.

Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1717–1724, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.222.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, Oct 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191.

Vladan Radosavljevic, Slobodan Vucetic, and Zoran Obradovic. Continuous conditional random fields for regression in remote sensing. *ECAI*, 2010.

Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. *CoRR*, abs/1403.6382, 2014.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, pages 1–42, 2014.

David E Sahn and David Stifel. Exploring alternative measures of welfare in the absence of expenditure data. *Rev. Income Wealth*, 49(4):463–489, 1 December 2003.

Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

Uganda Bureau of Statistics. Uganda national panel survey 2011/2012, 2012.

United Nations. Transforming our world: The 2030 agenda for sustainable development, 2015.

Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016a.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. *The Journal of Machine Learning Research*, 2015.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 370–378, 2016b.

Ralph Wolf and John C. Platt. Postal address block location using a convolutional locator network. In *Advances in Neural Information Processing Systems*, pages 745–752. Morgan Kaufmann Publishers, 1994.

World Bank. Povcalnet online poverty analysis tool, http://iresearch.worldbank.org/povcalnet/, 2015.

World Resources Institute. Mapping a better future: How spatial analysis can benefit wetlands and reduce poverty in uganda, 2009.

Michael Xie, Neal Jean, Marshall Burke, David Lobell, and Stefano Ermon. Transfer learning from deep features for remote sensing and poverty mapping. *30th AAAI Conference on Artificial Intelligence*, 2016.

Kai Yu, Jinbo Bi, and Volker Tresp. Active learning via transductive experimental design. *The International Conference on Machine Learning (ICML)*, 2006.

Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

Chenyang Zhao and Shaodan Zhai. Minimum variance semi-supervised boosting for multi-label classification. In *2015 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1342–1346. IEEE, 2015.

Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems*, pages 487–495, 2014.

Zhi-Hua Zhou and Ming Li. Semi-supervised regression with co-training. In *IJCAI*, volume 5, pages 908–913, 2005.

Jun Zhu, Ning Chen, and Eric P Xing. Bayesian inference with posterior regularization and applications to infinite latent svms. *Journal of Machine Learning Research*, 15(1):1799–1847, 2014.

# Appendix A

# Appendix

## A.1 Proof of Theorem 1

*Proof of Theorem 1.* Let $\mathcal{D} = (X_L, \mathbf{y}_L, X_U)$ be a collection of observed data. Let $X = (X_L, X_U)$ be the observed input data points. As in Zhu et al. [2014], we assume that $\mathcal{F}$ is a complete separable metric space and $\Pi$ is a absolutely continuous probability measure (with respect to background measure $\eta$) on $(\mathcal{F}, \mathcal{B}(\mathcal{F}))$, where $\mathcal{B}(\mathcal{F})$ is the the Borel $\sigma$-algebra, such that a density $\pi$ exists where $d\Pi = \pi d\eta$. Let $\Theta$ be a space of parameters to the model, where we treat $\theta \in \Theta$ as random variables. With regards to the notation in the RegBayes framework, the model is the pair $M = (f, \theta)$. We assume as in Zhu et al. [2014] that the likelihood function $P(\cdot|M)$ is the likelihood distribution which is dominated by a $\sigma$-finite measure $\lambda$ for all $M$ with positive density, such that a density $p(\cdot|M)$ exists where $dP(\cdot|M) = p(\cdot|M)d\lambda$.

We would like to compute the posterior distribution

$$p(f, \theta|\mathcal{D}) = \frac{p(\mathcal{D}|f, \theta)\pi(f, \theta)}{\int_{f,\theta} p(f, \theta, \mathcal{D})d\eta(f, \theta)}$$

which is involves an intractable integral. We introduce a variational approximation $q(f, \theta) \in \mathcal{P}_{prob}$ which approximates $p(f, \theta|\mathcal{D})$, where $\mathcal{P}_{prob} = q : q(f, \theta|\mathcal{D}) = q(f|\theta, \mathcal{D})\delta_{\bar{\theta}}(\theta|\mathcal{D})$ is a family of approximating distributions such that $q(\theta|\mathcal{D})$ is restricted to be a Dirac delta centered on $\bar{\theta}$. We claim that the exact optimal solution of the following optimization problem is precisely the Bayesian posterior $p(f, \theta|\mathcal{D})$:

$$\inf_{q(f,\theta|\mathcal{D})} D_{KL}(q(f, \theta|\mathcal{D})\|p(f, \theta|X)) - \int_{f,\theta} q(f, \theta|\mathcal{D}) \log p(\mathbf{y}_L|f, \theta, X)d\eta(f, \theta).$$

We note that, adding the constant $\log p(\mathcal{D})$ to the objective,

$$\begin{aligned}
&\inf_{q(f,\theta|\mathcal{D})} D_{KL}(q(f,\theta|\mathcal{D})\|p(f,\theta|X)) - \int_{f,\theta} q(f,\theta|\mathcal{D}) \log p(\mathbf{y}_L|f,\theta,X) d\eta(f,\theta) + \log p(\mathcal{D}) \\
&= \inf_{q(f,\theta|\mathcal{D})} \int_{f,\theta} q(f,\theta|\mathcal{D}) \log \frac{q(f,\theta|\mathcal{D})p(X)}{p(f,\theta,\mathbf{y}_L,X)} d\eta(f,\theta) + \log p(\mathcal{D}) \\
&= \inf_{q(f,\theta|\mathcal{D})} \int_{f,\theta} q(f,\theta|\mathcal{D}) \log \frac{q(f,\theta|\mathcal{D})}{p(f,\theta,\mathcal{D})} d\eta(f,\theta) + \log p(\mathcal{D}) \\
&= \inf_{q(f,\theta|\mathcal{D})} \int_{f,\theta} q(f,\theta|\mathcal{D}) \log \frac{q(f,\theta|\mathcal{D})}{p(f,\theta|\mathcal{D})} d\eta(f,\theta) \\
&= \inf_{q(f,\theta|\mathcal{D})} D_{KL}(q(f,\theta|\mathcal{D})\|p(f,\theta|\mathcal{D})) \\
&= \inf_{q(f,\theta|\mathcal{D})} \mathcal{L}(q(f,\theta|\mathcal{D})),
\end{aligned}$$

the claim holds, and we see that the objective is equivalent to the first term of the RegBayes objective. When we restrict $q \in \mathcal{P}_{prob}$,

$$\inf_{q(f,\theta|\mathcal{D})\in\mathcal{P}_{prob}} D_{KL}(q(f,\theta|\mathcal{D})\|p(f,\theta|X)) - \int_{f,\theta} q(f,\theta|\mathcal{D}) \log p(\mathbf{y}_L|f,\theta,X) d\eta(f,\theta) \tag{A.1}$$

$$= \inf_{q(f|\theta,\mathcal{D}),\bar{\theta}} \int_\theta \delta_{\bar{\theta}}(\theta) \int_f q(f|\theta,\mathcal{D}) \left( \log \frac{q(f|\theta,\mathcal{D})\delta_{\bar{\theta}}(\theta)}{p(f,\theta|X)} - \log p(\mathbf{y}_L|f,\theta,X) \right) d\eta(f,\theta) \tag{A.2}$$

$$= \inf_{q(f|\theta,\mathcal{D}),\bar{\theta}} \int_\theta \delta_{\bar{\theta}}(\theta) \int_f q(f|\theta,\mathcal{D}) \left( \log \frac{q(f|\theta,\mathcal{D})}{p(f,\theta|X)} - \log p(\mathbf{y}_L|f,\theta,X) \right) d\eta(f,\theta) \tag{A.3}$$

$$\begin{aligned}
= \inf_{q(f|\theta,\mathcal{D}),\bar{\theta}} \int_\theta \delta_{\bar{\theta}}(\theta) \int_f q(f|\theta,\mathcal{D}) \Big( &\log q(f|\theta,\mathcal{D}) - \log p(f,\theta|X) - \log p(f|\mathbf{y}_L,\theta,X) \\
&- \log p(\mathbf{y}_L,\theta,X) + \log p(f,\theta,X) \Big) d\eta(f,\theta)
\end{aligned} \tag{A.4}$$

$$\begin{aligned}
= \inf_{q(f|\theta,\mathcal{D}),\bar{\theta}} \int_\theta \delta_{\bar{\theta}}(\theta) \int_f q(f|\theta,\mathcal{D}) \Big( &\log q(f|\theta,\mathcal{D}) - \log p(f,\theta|X) - \log p(f|\mathbf{y}_L,\theta,X) \\
&- \log p(\mathbf{y}_L,\theta|X) + \log p(f,\theta|X) \Big) d\eta(f,\theta)
\end{aligned} \tag{A.5}$$

$$= \inf_{q(f|\theta,\mathcal{D}),\bar{\theta}} \int_\theta \delta_{\bar{\theta}}(\theta) (D_{KL}(q(f|\theta,\mathcal{D})\|p(f|\mathbf{y}_L,\theta,X)) - \log p(\mathbf{y}_L,\theta|X)) \tag{A.6}$$

where in equation (A.3) we note that $\int_\theta \delta_{\bar{\theta}}(\theta) \log \delta_{\bar{\theta}}(\theta)$ does not vary with $\bar{\theta}$ or $q$, and can be removed from the optimization, and similarly in equation (A.5) we remove the constant $\log p(X)$. For any $\theta$, the optimizing value is $q^*(f|\theta,\mathcal{D}) = p(f|\mathbf{y}_L,\theta,X) = p(f|\theta,\mathcal{D})$, which is the Bayesian posterior given the model parameters. Substituting this optimal value into (A.6),

$$\inf_{\bar{\theta}} \int_\theta \delta_{\bar{\theta}}(\theta) (D_{KL}(q^*(f|\theta,\mathcal{D})\|p(f|\mathbf{y}_L,\theta,X)) - \log p(\mathbf{y}_L,\theta|X)) \tag{A.7}$$

$$= \inf_{\bar{\theta}} - \int_\theta \delta_{\bar{\theta}}(\theta) \log p(\mathbf{y}_L,\theta|X)) \tag{A.8}$$

$$= \inf_{\bar{\theta}} - \log p(\mathbf{y}_L|\bar{\theta},X) \tag{A.9}$$

using that $D_{KL}(q^*(f|\theta,\mathcal{D})\|p(f|\mathbf{y}_L,\theta,X)) = 0$ in (A.8). The optimization problem over $\bar{\theta}$ reflects maximizing the likelihood of the data.

The regularization term in the RegBayes framework is expressed variationally as

$$\Omega(q(M|\mathcal{D})) = \inf_{\xi} U(\xi)\, s.t. : q(M|\mathcal{D}) \in \mathcal{P}_{post}(\xi),$$

where $\xi$ are slack variables, $U(\xi)$ is a penalty function, and $\mathcal{P}_{post}(\xi)$ is a subspace of feasible distributions satisfying specified constraints. An equivalent formulation of the RegBayes problem is then

$$\textbf{RegBayes:} \inf_{q(M|\mathcal{D})\in\mathcal{P}_{post}(\xi),\xi} \mathcal{L}(q(M|\mathcal{D})) + U(\xi). \tag{A.10}$$

Let the regularization function be

$$\Omega(q(f,\theta|\mathcal{D})) = \inf_{\xi} U(\xi)\ \ s.t. : q(f,\theta|\mathcal{D}) \in \mathcal{P}_{post}(\xi),$$

where $U(\xi) = \alpha' \sum_{i=1}^{m} \xi_i$, $\alpha' = \frac{\alpha n}{m}$, and $q$ is restricted to the family of distributions

$$\mathcal{P}_{post} = \{q : \forall i = 1,\ldots,m, \left(\int_f (f(X_U)_i - \mathbb{E}_p[f(X_U)_i])^2 \int_\theta q(\theta|\mathcal{D})p(f|\theta,\mathcal{D})d\eta(f,\theta)\right) \le \xi_i,\ \ q \in \mathcal{P}_{prob}\}$$

where $p(f|\theta,\mathcal{D})$ is the Bayesian posterior from the unregularized objective. Note that given values of $q$ and $\bar{\theta}$, the optimal $\xi_i^* = \int_f (f(X_U)_i - \mathbb{E}_p[f(X_U)_i])^2 \int_\theta q(\theta|\mathcal{D})p(f|\theta,\mathcal{D})d\eta(\theta)$. Thus the regularization function is

$$\Omega(q(f,\theta|\mathcal{D})) = \alpha' \sum_{i=1}^{m} \left(\int_f (f(X_U)_i - \mathbb{E}_p[f(X_U)_i])^2 \int_\theta q(\theta|\mathcal{D})p(f|\theta,\mathcal{D})d\eta(f,\theta)\right).$$

Note that the regularization function only depends on $q(\theta|\mathcal{D}) = \delta_{\bar{\theta}}(\theta)$. Therefore the optimal post-data posterior $q$ in the regularized objective is still in the form $q^*(f,\theta|\mathcal{D}) = p(f|\theta,\mathcal{D})\delta_{\bar{\theta}}(\theta)$, and $q$ is modified by the regularization function only through $\delta_{\bar{\theta}}(\theta)$.

Thus, augmenting the objective from (A.6) and using the optimal post-data posterior $q^*(f,\theta|\mathcal{D}) =$

$p(f|\theta, \mathcal{D})\delta_{\bar{\theta}}(\theta)$, the regularized optimization objective is

$$
\begin{aligned}
&\inf_{q(f,\theta|\mathcal{D})\in\mathcal{P}_{prob}} \mathcal{L}(q(f,\theta|\mathcal{D})) + \Omega(q(f,\theta|\mathcal{D})) \\
&= \inf_{\bar{\theta}} -\log p(\mathbf{y}_L|\bar{\theta}, X) + \alpha' \sum_{i=1}^{m} \left( \int_f (f(X_U)_i - \mathbb{E}_p[f(X_U)_i])^2 \int_\theta \delta_{\bar{\theta}}(\theta) p(f|\theta, \mathcal{D}) d\eta(f,\theta) \right) \\
&= \inf_{\bar{\theta}} -\log p(\mathbf{y}_L|\bar{\theta}, X) + \alpha' \sum_{i=1}^{m} \int_f p(f|\bar{\theta}, \mathcal{D})(f(X_U)_i - \mathbb{E}_p[f(X_U)_i])^2 d\eta(f) \\
&= \inf_{\bar{\theta}} -\log p(y|\bar{\theta}, X) + \alpha' \sum_{i=1}^{m} \int_f q(f|\bar{\theta}, \mathcal{D})(f(X_U)_i - \mathbb{E}_{q^*}[f(X_U)_i])^2 d\eta(f) \\
&\quad \inf_{\bar{\theta}} -\log p(\mathbf{y}_L|\bar{\theta}, X) + \alpha' \sum_{i=1}^{m} \mathrm{Var}_{q^*}(f((X_U)_i)) \\
&= \inf_{\bar{\theta}} L_{semisup}(\bar{\theta}).
\end{aligned}
$$

where we use $p(f|\bar{\theta}, \mathcal{D}) = q^*(f|\bar{\theta}, \mathcal{D})$ in the second-to-last equality. □

## A.2 Filter Activation Maps for Transfer Learning Method

We provide 25 maximally activating images in the validation set and their activation maps for four filters in our CNN model (Figures A.1, A.2,A.3,A.4,A.5) as in Chapter 2 and Xie et al. [2016]. The activation maps indicate the locations where the filter activated the most. These filters seem to activate to different terrain types, man-made structures, and roads, all of which can be useful socioeconomic indicators.

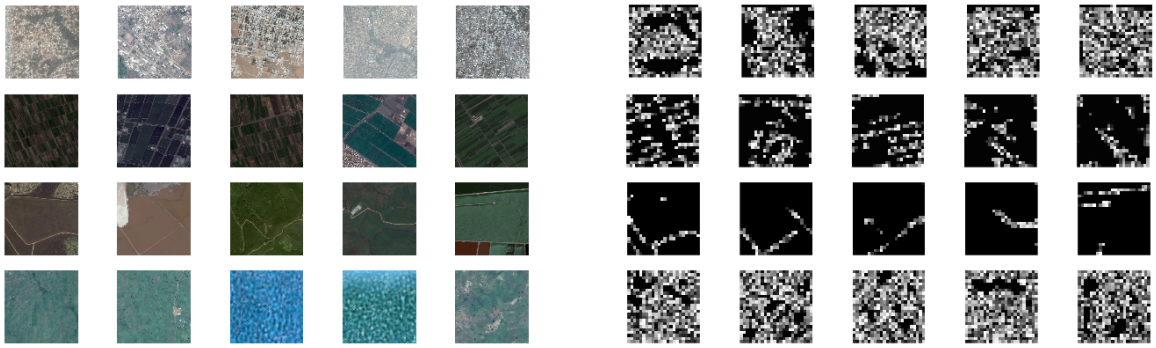

Figure A.1: **Left:** Each row shows five maximally activating images for a different filter in the fifth convolutional layer of the CNN trained on the nighttime light intensity prediction problem. The first filter (first row) activates for urban areas. The second filter activates for farmland and grid-like patterns. The third filter activates for roads. The fourth filter activates for water, plains, and forests, terrains contributing similarly to nighttime light intensity. The only supervision used is nighttime light intensity, i.e., no labeled examples of roads or farmlands are provided. **Right:** Filter activations for the corresponding images on the left. Filters mostly activate on the relevant portions of the image. For example, in the third row, the strongest activations coincide with the road segments. Best in color. Images from Google Static Maps.

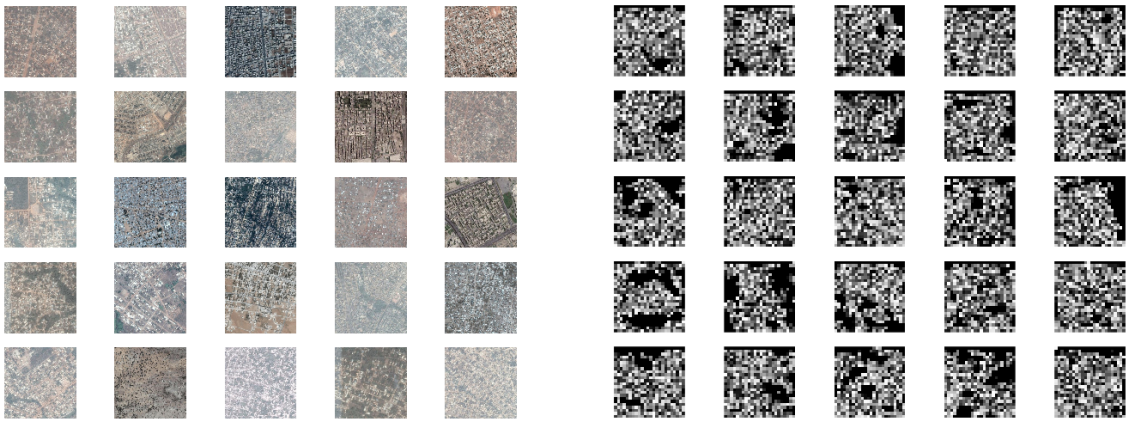

Figure A.2: A set of 25 maximally activating images and their corresponding activation maps for a filter in the fifth convolutional layer of the network trained on the 3-class nighttime light intensity prediction task. This filter seems to activate for urban areas, which indicate economic development.

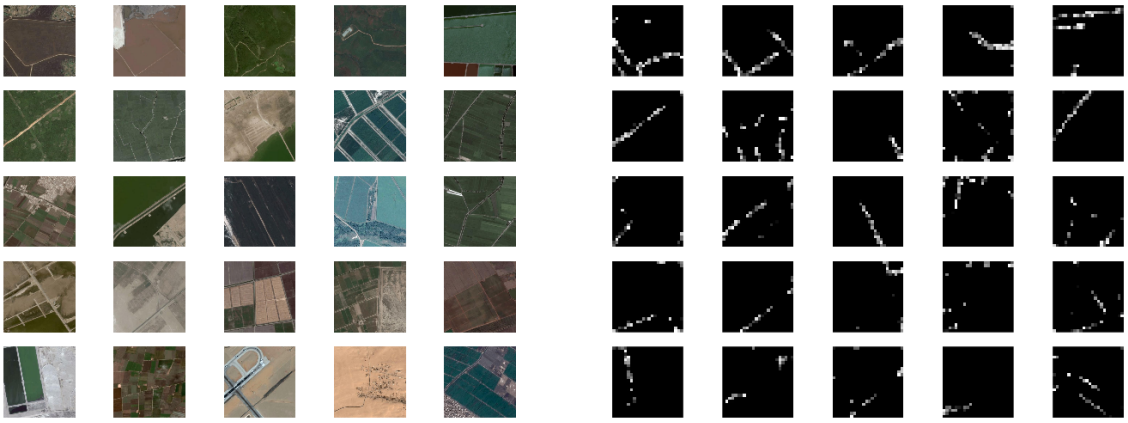

Figure A.3: A set of 25 maximally activating images and their corresponding activation maps for a filter in the fifth convolutional layer of the network trained on the 3-class nighttime light intensity prediction task. This filter seems to activate for roads, which are indicative of infrastructure and economic development.

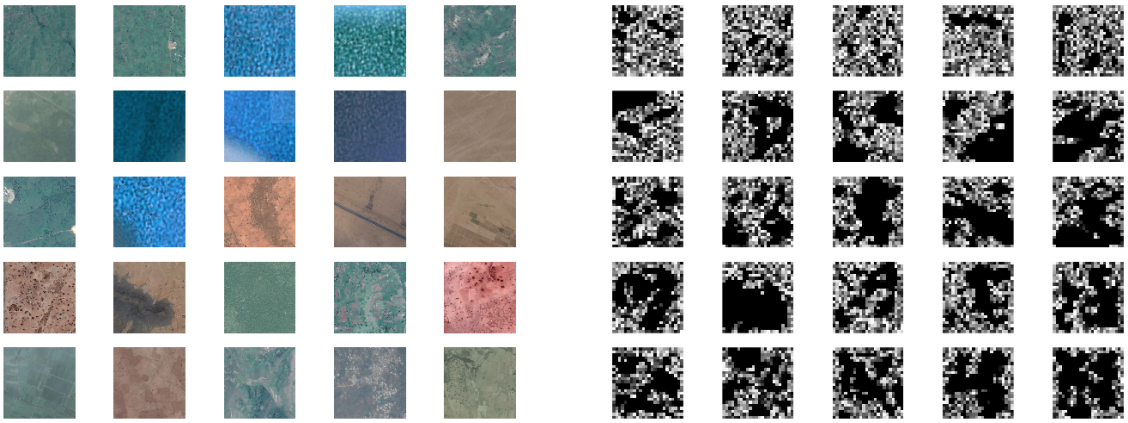

Figure A.4: A set of 25 maximally activating images and their corresponding activation maps for a filter in the fifth convolutional layer of the network trained on the 3-class nighttime light intensity prediction task. This filter seems to activate for water, barren, and forested lands, which this filter seems to group together as contributing similarly to nighttime light intensity.

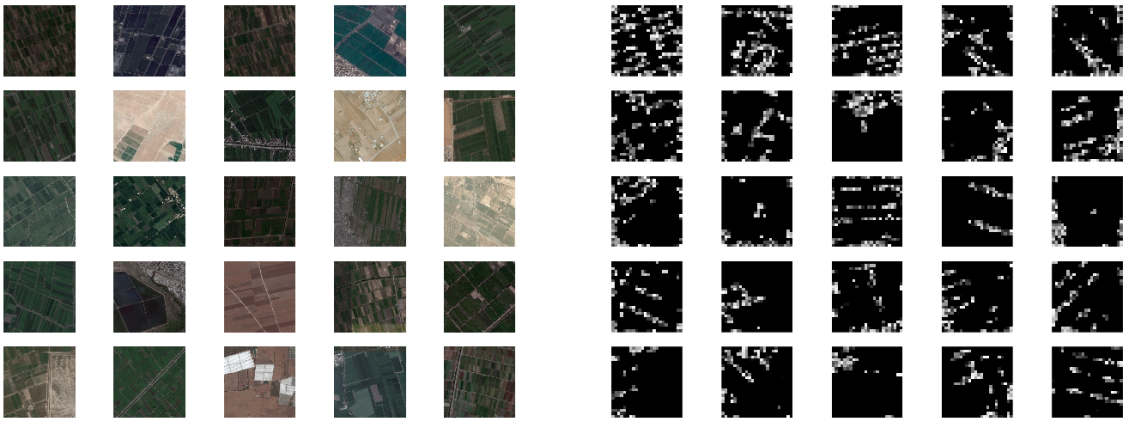

Figure A.5: A set of 25 maximally activating images and their corresponding activation maps for a filter in the fifth convolutional layer of the network trained on the 3-class nighttime light intensity prediction task. This filter seems to activate for farmland and for grid-like patterns, which are common in human-made structures.

## A.3 Other Spatial Models

### A.3.1 Problem Setup

In this spatial modeling problem, we are given a set of locations, each with a given feature vector associated with the location, and partially observed labels, which we know to be spatially correlated. In Xie et al. [2016] and [Jean et al., 2016a], we trained a ridge regression model from these feature vectors, treating each location independently. We explore ways to do better by supplying a joint assignment which takes into account spatial correlations. While Gaussian processes define the covariance matrix directly, solving quadratic optimization problems is a way of defining a precision matrix of a Gaussian that gives a different way of controlling the spatial dependencies captured in the model. One of the assumptions made hereafter is that the relationship between the feature vectors and the label at each location is roughly linear; since we get good performance with the ridge regression models, this is assumed to be reasonable. For differentiable models, we can backpropagate through the spatial model and also tweak the neural network so that the feature vectors are more linear with respect to the labels, as in Jean et al. [2017].

In general, the models here are less applicable than Gaussian process models because it requires all the possible query locations to be in the graph during training. Testing on new data without retraining the parameters is not straightforward because there are new factors introduced into the graph when introducing new data. By directly defining the covariance matrix through the kernel function, Gaussian processes get around this limitation and inherently defines all finite dimensional distributions of query locations.

### A.3.2 Weak Predictor Model

In the first attempt, we try to build directly off of the output of the ridge regression model, regarding the ridge regression outputs as the output of a "weak classifier", which we enhance by taking into account spatial correlations. One drawback is that we must supply this weak classifier, and since it is not a parameter in our model, we must find the parameters through cross-validation.

Let $w(x, \lambda)$ be the output of a ridge regression model with regularization parameter $\lambda$. A conditional random field is a probabilistic graphical model where the random variables are only depend on its neighbors and the evidence. We construct a grid-structured conditional random field model where each label variable is connected to up to 4 neighbors via pairwise potentials; let the set of neighbor pair indices be $N$. Each location's label variable is connected to a feature vector and a ridge regression output. We note that this model is similar to the one seen in Radosavljevic et al. [2010], but with the added twist that the data is partially observed. We incorporate partially observed data by conditioning upon the evidence. We model the conditional probability distribution (given the features) as a Gaussian by specifying the conditional log-likelihood:

$$\log P(y \mid w, x; \beta) = \sum_i (y_i - w^T(x_i))^2 + \beta \sum_{(i,j)\in N} (y_i - y_j)^2 - Z(\beta, w)$$

where $Z(\beta, w)$ is the partition function (involving the determinant) and $y$ are all the locations.

We can now see that the log-likelihood is equivalent to

$$\log P(y \mid w, x; \beta) = -\left(y^T(\beta D^T D + I)y - 2(Xw)^T y + (Xw)^T(Xw)\right) - \log Z(\beta, w).$$

Since this is in a quadratic form (the linear term serves as the mean shift), we can view this as a Gaussian, and perform MAP inference by outputting the mean. The mean of the Gaussian without observing evidence is

$$\mu_0 = (\beta D^T D + I)^{-1}(Xw)$$

which we can solve via conjugate gradients. Let the indices of the observed labels be $E$, the unobserved labels be $U$, and $\bar{y}$ be the observed labels. The conditional precision matrix can be written as a block matrix corresponding to 4 sections:

$$\Sigma^{-1} = \begin{bmatrix} \Gamma_{UU} & \Gamma_{UE} \\ \Gamma_{EU} & \Gamma_{EE} \end{bmatrix}$$

$$\mu_0 = \begin{bmatrix} \mu_U \\ \mu_E \end{bmatrix}.$$

Our prediction is the shifted mean after conditioning on the evidence, which is

$$y_{pred} = \mu_U - \Gamma_{UU}^{-1}\Gamma_{UE}(\bar{y} - \mu_E).$$

We note earlier that the regularization parameter $\lambda$ for our weak classifier (ridge regression) must be found through cross-validation. In order to not peek at the test set, we find $\lambda$ in an inner cross-validation loop. We can train the spatial parameter $\beta$ by optimizing the marginal likelihood or test loss, and do simple grid search over the one parameter $\beta$.

In preliminary results, the weak predictor model attains test $r^2$ values of 0.49 on the consumption expenditure prediction problem in Uganda as described in Jean et al. [2016a], which represents a boost from the 0.46 attained from treating the locations as independent.

### A.3.3 Joint model

Instead of using a weak predictor to loosely base our predictions off of, we would like to insert the parameters for the "linearity" term into our model as well, instead of allowing the ridge regression to model the linearity. We present a possible way to do this here, but do not present results. Let the label for each location be in a vector $y \in \mathbb{R}^m$ and the vector of weights $w \in \mathbb{R}^n$ be the linearity term. The variables in the model are $y$ and $w$. We specify the log-likelihood as follows:

$$\log p(y, w \mid x; \beta, \lambda) = -\left(\beta \sum_{i,j \in N} (y_i - y_j)^2 + \sum_i (y_i - w^T x_i)^2 + \lambda w^T w\right)$$

As before, we have a grid-structured $y$ that is connected at each location to its respective feature vector in $x$ (which could have arbitrary connections within themselves since we condition upon them). We have a shared weight vector $w$ which connects to all $y$ and $x$ via factors of size 3. Notice that the last two terms are of the same form as ridge regression. Converting this to matrix form, we can view this as an inner product

$$\log p(y, w \mid x; \beta, \lambda) = \left(\begin{bmatrix} \sqrt{\beta}D & 0 \\ I & -X \\ 0 & \sqrt{\lambda}I \end{bmatrix} \begin{bmatrix} y \\ w \end{bmatrix}\right)^T \left(\begin{bmatrix} \sqrt{\beta}D & 0 \\ I & -X \\ 0 & \sqrt{\lambda}I \end{bmatrix} \begin{bmatrix} y \\ w \end{bmatrix}\right)$$

where $D$ is the difference matrix as before ($D^T D$ is $(m \times m)$) and $X$ is the $(m \times n)$ matrix of feature vectors. The log-likelihood can be rewritten as

$$\log p(y, w \mid x; \beta, \lambda) = -\begin{bmatrix} y \\ w \end{bmatrix}^T \Sigma^{-1} \begin{bmatrix} y \\ w \end{bmatrix} - \log Z(\beta, \lambda)$$

where

$$\Sigma^{-1} = \begin{bmatrix} \beta D^T D + I & -X \\ -X^T & X^T X + \lambda I \end{bmatrix}$$

which is a Gaussian with mean 0, where the mean shifts as we condition on some subset of $y$.

**Inference and Training**

We notice that unlike in the case of Gaussian Processes, it is relatively hard to find the marginal likelihood of a subset of examples since we must invert $\Sigma^{-1}$ blockwise. However, since we directly specify the precision matrix, conditioning is easy - just delete the rows and columns corresponding to the evidence for the new precision matrix, and shift the mean by solving a linear system involving the evidence. We note that, after finding the new conditional mean, it is also easy to marginalize the mean(our prediction) to test examples; we ignore the other elements of the mean.

We use the fact that we can make predictions conditioned on new evidence relatively easily and attempt to optimize the parameters $\beta$ and $\lambda$ by minimizing test error on leave-one-out folds of training patches (sampled patches of locations). While we can also conduct a simple grid search, we explore the use of Bayesian optimization to learn a setting of these parameters. We will attempt to take a gradient step after each patch, averaging the gradients from the leave-one-out errors. Note that $\beta$ and $\lambda$ are not hyperparameters, but direct parameters of our model that (along with the given feature vectors) fully specifies the Gaussian in the Bayesian optimization point of view.

Let $E$ be the indices of partially observed locations, $U$ be the indices of unobserved locations and the weight vector (also unobserved). Let $\bar{y}$ be the observed labels. The conditional precision matrix can be written as a block matrix corresponding to 4 sections:

$$\Sigma^{-1} = \begin{bmatrix} \Gamma_{UU}(\beta, \lambda) & \Gamma_{UE}(\beta) \\ \Gamma_{EU}(\beta) & \Gamma_{EE}(\beta). \end{bmatrix}$$

$$\mu = \begin{bmatrix} \mu_U \\ \mu_E \end{bmatrix}$$

We write the inner block matrices as functions of the parameters for convenience, to show the dependence of each on each parameter. Using Schur complements, we find that the new conditional mean and precision matrix is

$$\mu_U \mid \bar{y} = \mu_U - \Gamma_{UU}^{-1}\Gamma_{UE}(\bar{y} - \mu_E) = -\Gamma_{UU}^{-1}\Gamma_{UE}\bar{y}$$

$$\Sigma^{-1} \mid \bar{y} = \Gamma_{UU}$$

where we have used that the Gaussian is mean 0 before conditioning.

The objective function for the leave-one-out scheme is a squared error loss. Let $y_T$ be the test

label, where $T$ is the index of the test location. Let the new conditional mean after conditioning on the training data be $\mu'$. Let the training evidence be $\bar{y}$, as before. Then the loss is

$$err(\beta, \lambda) = \frac{1}{2}(y_T - \mu'_T)^2.$$

We take the gradient. Let

$$\delta = y_T - \mu'_T.$$

We have that

$$\nabla_\beta err(\beta, \lambda) = \delta(-\nabla_\beta \mu')_T.$$

Let $Y$ be indices of the unobserved locations, and $W$ be the indices of the weight vector, so that the union of $Y$ and $W$ is $U$. We can view $\mu'$ as a function of $\beta, \lambda$ and differentiate:

$$\begin{aligned}
(-\nabla_\beta \mu') &= \nabla_\beta \Gamma_{UU}^{-1} \Gamma_{UE} \bar{y} \\
&= -\Gamma_{UU}^{-1} (\nabla_\beta \Gamma_{UU}) \Gamma_{UU}^{-1} \Gamma_{UE} \bar{y} + \Gamma_{UU}^{-1} (\nabla_\beta \Gamma_{UE}) \bar{y} \\
&= -\Gamma_{UU}^{-1} \begin{bmatrix} (D^T D)_{YY} & 0 \\ 0 & 0 \end{bmatrix} \Gamma_{UU}^{-1} \Gamma_{UE} \bar{y} + \Gamma_{UU}^{-1} \begin{bmatrix} (D^T D)_{YW} \\ 0 \end{bmatrix} \bar{y} \\
&= \Gamma_{UU}^{-1} \left( - \begin{bmatrix} (D^T D)_{YY} & 0 \\ 0 & 0 \end{bmatrix} \Gamma_{UU}^{-1} \Gamma_{UE} \bar{y} + \begin{bmatrix} (D^T D)_{YW} \\ 0 \end{bmatrix} \bar{y} \right) \\
&= \Gamma_{UU}^{-1} \left( \begin{bmatrix} (D^T D)_{YY} & 0 \\ 0 & 0 \end{bmatrix} \mu' + \begin{bmatrix} (D^T D)_{YW} \\ 0 \end{bmatrix} \bar{y} \right)
\end{aligned}$$

which is a system of linear equations, which we can solve efficiently with conjugate gradients. Therefore

$$\nabla_\beta err(\beta, \lambda) = \delta \cdot \Gamma_{UU}^{-1} \left( \begin{bmatrix} (D^T D)_{YY} & 0 \\ 0 & 0 \end{bmatrix} \mu' + \begin{bmatrix} (D^T D)_{YW} \\ 0 \end{bmatrix} \bar{y} \right)_T.$$

Similarly, we have

$$\nabla_\lambda err(\beta, \lambda) = \delta \cdot \Gamma_{UU}^{-1} \left( \begin{bmatrix} 0 & 0 \\ 0 & I_{WW} \end{bmatrix} \mu' \right)_T$$

which can also be calculated efficiently via conjugate gradients. The gradient updates are in the log-domain, so that

$$\nabla_{\log \beta} err(\beta, \lambda) = \beta \nabla_\beta err(\beta, \lambda)$$

$$\nabla_{\log \lambda} err(\beta, \lambda) = \lambda \nabla_\lambda err(\beta, \lambda).$$

Regularization terms can be added to the error function to ensure that the parameters do not blow up, as in [Radosavljevic et al., 2010].

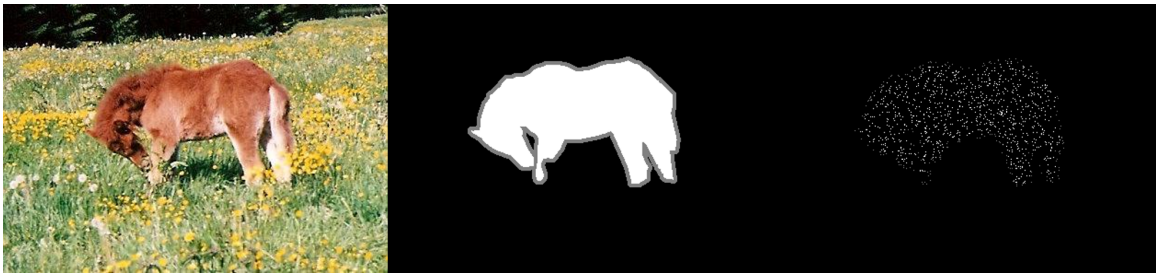

Figure A.6: **Left**:Image used in the toy problem. **Middle**: Ground truth for the toy problem. **Right**: Observed pixel labels for training (5% of total pixels), where we have not shown observed black pixels.

Figure A.7: Ridge regression predictions.

### A.3.4 Toy Segmentation Problem

To visualize what the models are doing, we test on a toy image segmentation problem, where the locations are the pixels, the feature vectors are the RGB values, and the segmentation class is viewed as a continuous-valued output. The original image presents a foreground object-background segmentation task. We allow the model to observe 5% of the pixels in this image, visualized here without the observed black pixels in Figure A.6. We visualize the results from training a ridge regression model to predict the label(continuous valued) from RGB in Figure A.7. Note that the ridge regression is mostly a color filter; the flowers are horse-like in color since the horse's brown fur contains yellow.

We visualize the weak predictor model's predictions for different settings of $\beta$ in Figure A.8. The weak predictor model derives the answer directly from the ridge outputs, smoothing out the outputs as $\beta$ increases.

We visualize the joint model's predictions for different settings of $\beta$ in Figure A.9. The parameter $\lambda$ seems to matter less in this problem, since there may not be a clear linear relationship between

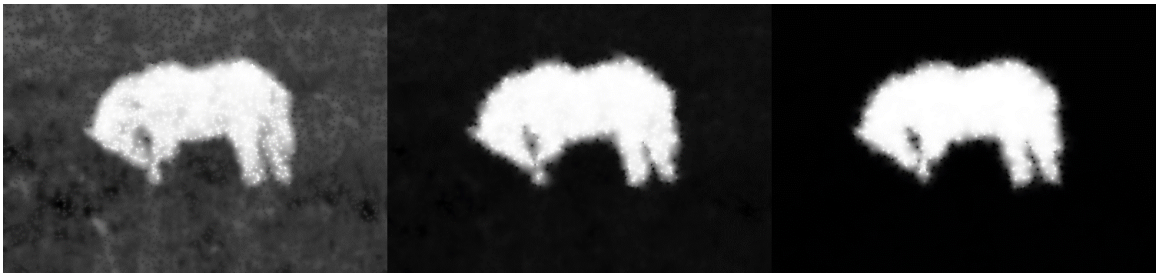

Figure A.8: **Left**: Weak Predictor model predictions with $\beta = 10$. **Middle**: Predictions with $\beta = 100$. **Right**: Predictions with $\beta = 1000$.

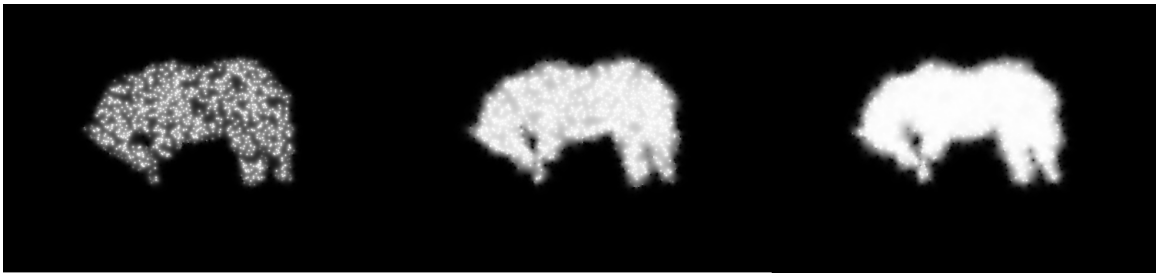

Figure A.9: **Left**: Joint model predictions with $\beta = 10$. **Middle**: Predictions with $\beta = 100$. **Right**: Predictions with $\beta = 1000$.

the evidence and the output. We set $\lambda = 0$ here so that there is no linear component. Training using the method described in the previous subsection is unstable so far, as the model decides to push $\beta$ as high as it can. The computational complexity of inference also scales with $\beta, \lambda$, which may pose a problem for problems requiring large values of $\beta, \lambda$. Despite limitations of the model, we see that indeed, for this toy problem, considering the spatial relationship of the image segmentation is very important for getting a good segmentation that ignores the background.