Estimating the Spatial Resolution of Very High-Resolution Overhead Imagery

Haolin Liang
University of California, Merced
hliang27@ucmerced.edu

Shawn Newsam
University of California, Merced
snewsam@ucmerced.edu

ABSTRACT

We investigate the problem of estimating the spatial resolution of overhead imagery. More overhead imagery is becoming available without such meta-data either because it was not collected in the first place or was not preserved with the imagery. Knowing the spatial resolution can be important for a range of automated image understanding tasks such as object detection, semantic segmentation, etc. In this paper, we explore a regression framework with a feature extraction frontend and a dilated convolution backend to estimate the spatial resolution of an overhead image. We show that a stacked auto-encoder frontend outperforms a standard convolution neural network feature extractor. In order to demonstrate our approach, we construct an evaluation dataset consisting of a large collection of very high-resolution overhead images with spatial resolutions ranging from 0.15 to 1.0 meters per pixel.

CCS CONCEPTS

• **Computing methodologies** → *Scene understanding*; *Supervised learning by regression*; *Neural networks*.

KEYWORDS

spatial resolution estimation, deep learning regression, stacked auto-encoder

ACM Reference Format:
Haolin Liang and Shawn Newsam. 2019. Estimating the Spatial Resolution of Very High-Resolution Overhead Imagery. In *3rd ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery (GeoAI'19), November 5, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3356471.3365241

1 INTRODUCTION

Knowing the spatial resolution, in terms of meters per pixel, for example, of overhead imagery acquired from satellite, aerial, or, more recently, drone platforms is important for the automated analysis of the imagery. Most approaches to object detection or scene classification assume the spatial resolution is known or, at least, that the spatial resolution of the training and target images are the same. It is a far more difficult task to design methods that work on images with unknown and possibly widely varying resolution; i.e., the methods need to be scale invariant.

We therefore focus on the problem of estimating the spatial resolution of overhead imagery. This imagery has traditionally been acquired in a formal, structured way so that the acquisition meta-data is known and preserved. However, the increased ability to capture overhead imagery, particularly by the public using drones, means that the data's provenance is often not known. Spatial resolution is an important component of overhead image meta-data.

We take a bottom-up, data-driven approach and formulate the problem as one of regression. The input is an overhead image and the output is the estimated resolution in terms of meters per pixel. Key to this approach are the features that are extracted from the image in the regression pipeline. Deep learning approaches have shown remarkable ability to learn features that are effective for a range of image analysis tasks. We take motivation from this and use a stacked auto-encoder (SAE) frontend to learn and extract features which are then fed to a dilated convolution backend. We show the SAE outperforms a standard convolutional neural network (CNN) frontend for our regression problem. An overview of our model is shown in figure 1. We evaluate our approach using a sizable dataset of very high-resolution overhead images with a range of resolutions.

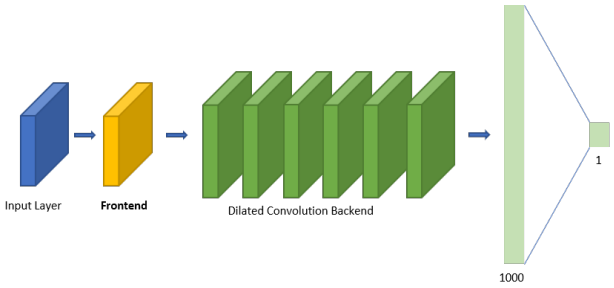

Figure 1: Our deep learning regression model takes an overhead image as input and outputs an estimate of its spatial resolution. We compare two different feature extraction frontends, a stacked auto-encoder (SAE) and a standard convolutional neural network (CNN).

2 RELATED WORK

There has been very little work on estimating the spatial resolution of overhead imagery likely, again, due to this being a novel problem.

Our regression framework is inspired by deep learning regression approaches to other image analysis problems such as estimating a person's age [9] or the orientation of their head [1]. Deep learning regression has also been applied to crowd counting in single surveillance images [5, 10, 12, 14, 15].

GeoAI'19, November 5, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6957-2/19/11…$15.00
https://doi.org/10.1145/3356471.3365241

SAEs have shown to be effective feature extractors for remote sensing image analysis. A recent survey [4] lists a number of works in which SAEs are used for remote sensing image classification.

To our knowledge, the only previous work on estimating the spatial resolution of overhead imagery is our own [6]. That work also uses deep learning regression but does not use an SAE frontend for improved feature extraction. It also targets lower-resolution imagery and therefore is not appropriate for the very high-resolution imagery from drones which is a primary objective of this paper.

3 METHODOLOGY

We formulate estimating the spatial resolution of an overhead image, in meters per pixel, as a single value regression problem. Our approach is data driven in that our regression model is trained on a set of labelled data with the labels being the spatial resolution. Below, we describe our model, noting, in particular, the two different frontend feature extractors we consider. Please see the overview of our model in figure 1 for reference.

3.1 VGG-16 Frontend

For our basline frontend feature extractor, we use a VGG network [11] which is a deep CNN originally designed for image classification. We use a VGG-16 CNN pretrained on the ImageNet dataset. Despite being trained for image classification using the ImageNet dataset, this network has been shown to have learned effective image features for a number of image analysis problems. It therefore represents a good baseline for our work. We feed the feature maps from final the convolutional layer of the VGG-16 network to the dilated convolution backend.

3.2 Auto-Encoder Frontend

Auto-encoders are unsupervised CNN models that have been shown to be effective for dimensionality reduction or feature learning. As shown in figure 2, they consist of an encoder, which reduces the dimensionality of the input to produce an embedding, and a decoder, which reconstructs the input from this embedding. The idea is that this embedding, often referred to as the hidden layer, is a faithful representation of the input if it can be used to reconstruct the output. A major advantage of auto-encoders is that they can be trained in an unsupervised fashion since the training data does not need to be labelled. The loss function used during iterative learning is based on the similarity between the input and its reconstruction. This allows auto-encoders to be trained using large collections of unlabelled data.

3.3 Stacked Auto-Encoder Frontend

SAEs are formed using multiple nested auto-encoders. In particular, the hidden layer (embedding) of the outermost auto-encoder is treated as the input to another auto-encoder. This is then repeated for the desired depth. The hidden layer of the deepest auto-encoder is used as the final embedding. This representation has been shown to be more effective than that of single layer auto-encoders. SAEs are trained in a greedy fashion. Each sub-AE is trained independent of its encapsulating layers.

We use an SAE with three sub-AEs as shown in figure 3. X and $\hat{X}$ represent the input and reconstructed images respectively. The h_i represent the hidden layers. These are input to the next sub-AE and reconstructed as $\hat{h_i}$. For clarity, we do not show the encoder and decoders in this figure. The encoders consist of one convolutional layer with a kernel of size 3 followed by a maxpooling layer with a kernel size of 2. The decoders reverse this through deconvolution. The goal is to train the SAE so that h_4 is an effective representation of the input image.

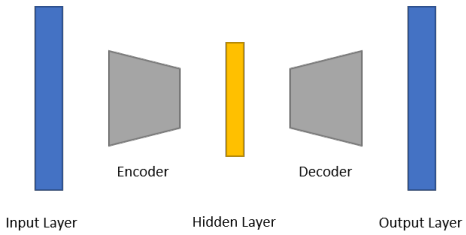

Figure 2: Auto-encoder network.

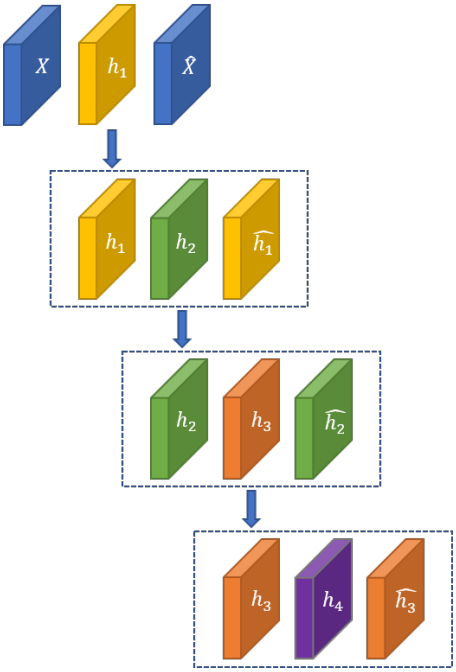

Figure 3: Our SAE model. See the text for details.

3.4 Dilated Convolution Backend

The output of the feature extractor frontend is input to a dilated convolution backend. The motivation here is that the receptive field increases quadratically with layer depth in standard convolution whereas it increases exponentially in dilated convolution. The receptive field is how much a particular feature map element "sees" of the two-dimensional input. A larger receptive field has been shown to improve performance for a number of tasks such as semantic segmentation [2]. Please see [13] for more on dilated convolution.

As shown in figure 1, our model contains five deconvolutional layers. The output of the final layer is input to a fully-connected layer to complete the regression.

4 EVALUATION DATASET

We compiled an evaluation dataset of very high-resolution overhead imagery with four difference spatial resolutions, 0.15, 0.3, 0.6, and 1.0 meters per pixel, from four different sources.

4.1 COWC

The Cars Overhead With Context (COWC) [8] dataset consists of overhead imagery with a spatial resolution of 0.15 m. It is annotated with positive and negative car instances but we do not utilize this information. The images are of six different locations: Toronto (Canada), Selwyn (New Zealand), Potsdam and Vaihingen (Germany), and Columbus and Utah (United States). We use 26 out of 27 image scenes in this dataset. The images vary in size from 2220×2220 to 18075×18400 pixels. The top row of figure 4 shows sample COWC subimages.

4.2 INRIA

The Inria Aerial Image Labeling Dataset (INRIA) [7] consists of aerial orthorectified color imagery with a spatial resolution of 0.3 m. It has ground truth labeling two semantic regions, building and non-building, but we do not utilize this information. The dataset covers a range of urban scenes in the United States in Europe ranging from densely populated areas to small towns. We use the entire dataset which consists of 360 images of size 1500×1500 pixels. The second row of figure 4 shows sample INRIA subimages.

4.3 NAIP

The United States Geological Survey National Agriculture Imagery Program (NAIP) collection consists of country-wide color aerial imagery of the US. We use 189 NAIP scences, 109 that have a spatial resolution of 0.6 m and 80 with a resolution of 1.0 m. The images vary in size from 5408×7616 to 10320×12190 pixels. The last two rows of figure 4 shows sample NAIP subimages.

4.4 Cross Sampling Between Datasets

To prevent our model from simply learning the one-to-one correspondence between data source and spatial resolution, our evaluation dataset consists of the original as well as resampled images.

We first tile the original images into 256×256 native resolution patches using 50% overlap. Table 1 shows the number of native resolution patches. We then form additional patches at each resolution by resampling random image regions from the other three resolutions. We use bilinear downsampling and upsampling. We realize that using upsampled images is not ideal but we were not able to find multiple sources for each resolution. We generate an additional 50000 256×256 resampled patches for each resolution.

Our final evaluation dataset consists of 20000 training patches and 2000 test patches for each resolution randomly sampled from the native and resampled sets. This is summarized in table 1.

5 EXPERIMENTS & RESULTS

5.1 Training

The model training is perfomed in two steps, first the SAE frontend and then the dilated convolution backend along with the regressor.

Figure 4: Sample subimages. Top row: 0.15 m COWC images; second row: 0.3 m INRIA images; third row: 0.6 m NAIP images; bottom row: 1.0 m NAIP images. Columns from left to right represent parking lot, vegetation, housing, and road regions to illustrate the variation in spatial resolution.

5.1.1 Training the SAE Frontend. The SAE frontend feature extractor is trained using all the image patches, both the native and resampled resolutions. Again, the objective during training is for the SAE to reconstruct the image patches from the learned embedding. We train the SAE for 100 epochs.

5.1.2 Training the Deconvolution Backend and Regressor. Once the frontend feature extractor is fixed as either the trained SAE or the pretrained VGG-16, we then train the rest of the model. This is done using the 80000 training image patches. The loss function during training is the L1 loss between the true and estimated spatial resolution of the training images. Two models are trained, one with the SAE frontend and the other with the VGG-16 frontend. We train each model for 100 epochs using the Adam optimizer [3] with a learning rate of 0.001. We also apply dropout with rate 0.5 to the fully connected layer to help prevent overfitting.

5.2 Evaluation

We measure performance by computing the average L1 error between the true and estimated spatial resolution over the 8000 test image patches

$$L1_{error} = \frac{1}{m} \sum_{i=1}^{m} |Y_i - \hat{Y}_i|$$

where Y_i denotes the true resolution of test image i, $\hat{Y}_i$ is the estimated resolution provided by the model, and $m = 8000$ is the number of test images.

5.3 Results

Table 2 compares the average error of the baseline VGG-16 and proposed SAE regression models. While both models do well at

Table 1: The evalution dataset.

Resolution (m/pixel)	# Native Patches	# Resampled Patches	# Training Set	# Testing Set
0.15	70496	50000	20000	2000
0.3	225000	50000	20000	2000
0.6	348222	50000	20000	2000
1.0	101520	50000	20000	2000

Table 2: Results on the test set.

Model	Average L1 error (m)
VGG-16	0.0246
SAE	0.0224

Table 3: Results broken down by resolution. The middle column is the average estimated resolution. All values are in meters.

True resolution	Estimated resolution	L1 error
0.15	0.16	0.014
0.3	0.29	0.013
0.6	0.60	0.030
1.0	0.97	0.035

estimating the spatial resolution, both are within 0.03 m of the true resolution on average over the entire test set, the SAE model results in an 8.9% *relative* performance improvement.

Table 3 breaks down the performance of the SAE model by resolution. The middle column shows the average estimated resolution which is shown to be very close to the true resolution in all cases. The average L1 error is shown to increase with resolution. One interesting observation is that the model tends to be biased at the limits of the range of resolutions. It overestimates at 0.15 m and underestimates at 1.0 m. It actually shows no bias at 0.6 m. We suspect this is a limitation of the regression model that would need to be accounted for by training on a dataset that extends beyond the range of target resolutions.

Finally, we show examples of image patches on which the model performs well (top row) and on which the model performs poorly (bottom row). As expected, the model performs worse on homogeneous images whose content provides little information regarding scale. This is a challenge not only for our method but any that relies only on the image content.

6 CONCLUSION

We investigated deep learning regression for estimating the spatial resolution of overhead imagery. This is a new but increasingly important problem as more imagery is collected through informal means. We showed an SAE feature extractor frontend improves performance over a general CNN frontend. We demonstrated the effectiveness of our approach using a dataset of very high-resolution images range from 0.15 to 1.0 meters per pixel.

Figure 5: Images on which our approach performs well (top) and performs poorly (bottom).

7 ACKNOWLEDGEMENT

This work was funded in part by a National Science Foundation grant, #IIS-1747535. We gratefully acknowledge the support of NVIDIA Corporation through the donation of the GPU card used in this work.

REFERENCES

[1] Byungtae Ahn, Jaesik Park, and In So Kweon. 2014. Real-time head orientation from a monocular camera using deep neural network. In *ACCV*.
[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 2018. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *ITPIDJ* 40, 4 (2018).
[3] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
[4] Ying Li, Haokui Zhang, Xizhe Xue, Yenan Jiang, and Qiang Shen. 2018. Deep learning for remote sensing image classification: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8, 6 (2018).
[5] Yuhong Li, Xiaofan Zhang, and Deming Chen. 2018. CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes. In *CVPR*.
[6] Haolin Liang and Shawn Newsam. 2019. Estimating The Spatial Resolution of Overhead Imagery Using Convolutional Neural Networks. In *ICIP*.
[7] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. 2017. Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark. In *IGARSS*.
[8] T Nathan Mundhenk, Goran Konjevod, Wesam A Sakla, and Kofi Boakye. 2016. A large contextual dataset for classification, detection and counting of cars with deep learning. In *ECCV*.
[9] Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. 2016. Ordinal regression with multiple output CNN for age estimation. In *CVPR*.
[10] Deepak Babu Sam, Shiv Surya, and R Venkatesh Babu. 2017. Switching convolutional neural network for crowd counting. In *CVPR*, Vol. 1.
[11] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556* (2014).
[12] Vishwanath A Sindagi and Vishal M Patel. 2017. Generating high-quality crowd density maps using contextual pyramid CNNs. In *ICCV*.
[13] Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv:1511.07122* (2015).
[14] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. 2015. Cross-scene crowd counting via deep convolutional neural networks. In *CVPR*.
[15] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. 2016. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*.