# A Machine Learning Approach to Image Geographical Localization for USA Cities

Scott Stewart[1], David Helminiak[1], and German Holguin[1]
Machine Learning for Image Processing EECE 5890, Dr Dong Hye Ye[1]

***Abstract*— The problem of geographical localization for images has been gaining attention in recent years; in an era where personal electronic devices with high resolution cameras have become commonplace, the act of actually labeling image contents and metadata has quickly led to a challenging problem, one that machine learning approaches may be best able to handle. A comparative study of several techniques, showcasing the progressive development of the field's capabilities for solving this task is herein presented. Experimentation was performed on crowd-sourced images throughout the United States using hand-crafted features, simple Convolutional Neural Networks (CNNs), and Deep CNNs (DCNNs) with measurable improvements and clear benefits for each identified along the way. Overall, DCNNs are shown to be the most accurate, with additional potentials for development noted.**

## I. INTRODUCTION

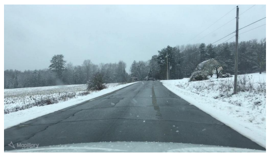
Services such as Flickr™, or Google Street View™ have massive image databases, leading to increased interest for algorithms that can infer their geotags (latitude and longitude). This was the main motivation for the ICCV'05 contest: "Where am I?", whose winner used feature matching through RANSAC and SIFT [1]. Recent work includes both, hand-crafted feature approaches, such as SIFT matching [2], and Deep Convolutional Neural Networks (DCNNs), as in PlaNet [3]. However, given the difficulty level, it is unlikely that a more efficient solution is in the near future [4].

This work uses a finite set of examples to model the appearance of a scene and then uses classification and/or regression based machine learning techniques to perform geolocalization. In particular, algorithms using traditional hand-crafted GIST feature descriptors, a simple CNN, and DCNN are each tested with two metrics: 1) % of test cases with a difference between predicted and actual labels less than 200 km and 2) $R^2$ loss to examine the overall fit between models and the training data.

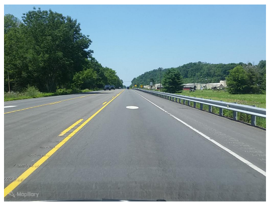
## II. DATA

A total of $5,623$ images were collected from the publicly available Mapillary database [5] each labeled by: 1) longitude, 2) latitude, and 3) region in the United States (distribution visualized in Fig. 2). The set comprises crowd-sourced images taken from vehicular dashboard cameras (Fig. 1). While all of the images are in color, their resolutions vary (though generally 720p/1080p); also sparsely observed are panoramic captures.

[1]Electrical, Electronics and Computer Engineering at Marquette University. Milwaukee, WI.



(a) $44°10'11.4''$N $69°19'26.3''$ W, Warren, ME, USA



(b) $41°01'25.3''$N $76°22'10.4''$ W, Bloomsburg, PA, USA

Fig. 1: Example images from the generated database

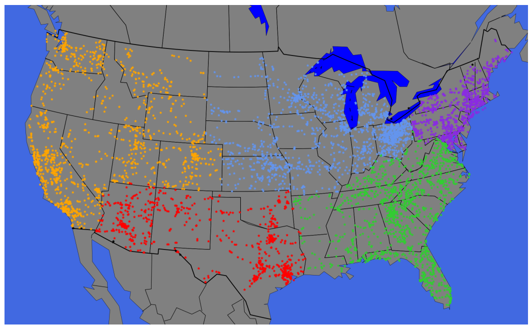


Fig. 2: Location for all images in the generated database with color mappings - Northeast: violet, Midwest: blue, Southeast: green, Southwest: red, and West: orange

## III. METHOD 1: HAND-CRAFTED FEATURES

Inspired by the winning entry of the ICCV 2005 "Where am I?" contest [1], it was decided to test using GIST feature descriptors. During the contest, GIST was used in combination with color histograms. However, while this combination helps for the urban context in the contest, it will not help here, since the images were taken in a more diverse manner and during different seasons.

### A. The GIST descriptor

The GIST descriptor came to prominence back in 2005 along with other prominent feature descriptors, such as HOG [6] and SIFT [7], which are strongly rotation/scale invariant and robust to illumination variation. While descriptors such as GIST are now known as "hand-crafted" features, GIST's inner structure has many similarities with current deep feature maps. Authors of GIST [8] claim its structure is biologically inspired, as the basic human vision system is based on 3 prominent representations (channels): orientation (shape), color (texture), and intensity (lightness). The combination of these channels achieves local object representation and a model for the "gist" of an image that the brain summarizes as a cognition result.

Following a similar strategy, Fig. 3 represents the GIST feature extraction process, detailed in [8]. The image is divided into cells using a grid of a fixed size. Per each cell, 3 representation channels are computed separately. For the orientation channel, 4 prominent orientations are computed using gradient orientations in a multi-scale pyramid. At each scale, several layers of Gabor filters (known to be good texture features) are concatenated into a single feature vector. A similar process is done for the color channel, using RGB histograms, and for the intensity channel. All representations for the cells are then concatenated into a single feature space. The output dimensionality of the GIST filter is then reduced using Principal Component Analysis and Independent Component Analysis with a fixed number of features: 960, kept for the representation. For both classification and regression see Section VI for final results.

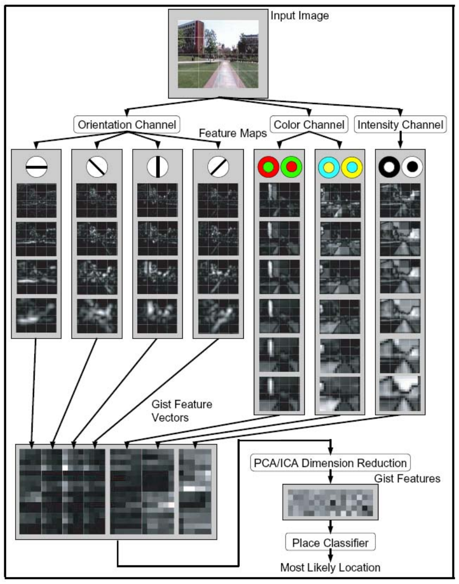


Fig. 3: Typical GIST feature extraction process

## B. *Classification using GIST features*

For the classification task, a Support Vector Machine (SVM) was implemented using `Python 3.7`, along with packages `scikit-learn` and `gist`. Looking to compare the effectiveness of classification using several typical kernels maps, the regularization parameter was fixed to 1.0 for all tested kernels.

$k$-fold cross-validation was employed with $k = 30$ to evaluate model performance. In each fold, the database was randomly split into 70% training and 30% testing sets. A total of 7 kernels were used: 1) linear, 2) polynomial of degree $n = 10$, 3) polynomial of degree $n = 50$, 4) Gaussian, $\gamma = 0.22$, 5) Gaussian, $\gamma = 0.54$, 6) Gaussian, $\gamma = 1/N\sigma$, and 7) sigmoid, $\gamma = 2$, where $\gamma$ is the kernel coefficient. Figure 4 shows the performance of each class and standard deviation among the folds.

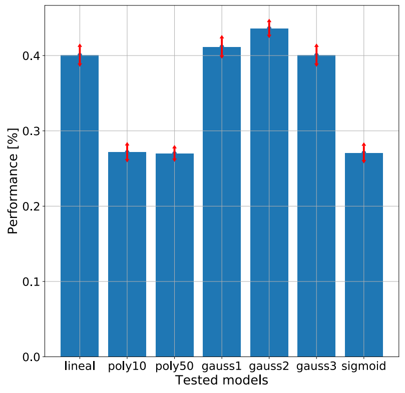
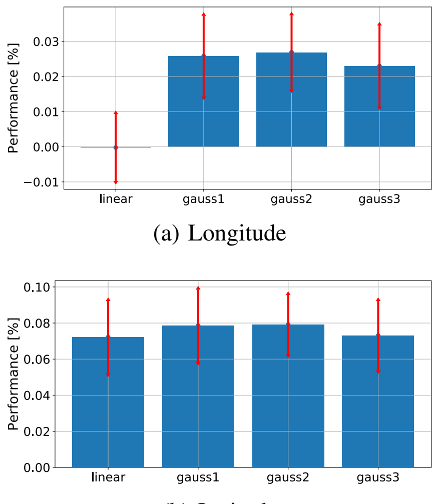


Fig. 4: Tested kernels for the classification task

## C. *Regression using GIST features*

For the regression task, separate regressors for latitude and longitude were implemented using Support Vector Regressors (SVRs). Again values for regularization were fixed to 1.0 and tolerance to $1 \times 10^{-3}$ for all kernels. A total of 4 kernels were used: 1) linear, $\epsilon = 0.1$, 2) Gaussian, $\gamma = 1/N\sigma$, $\epsilon = 0.2$, 3) Gaussian, $\gamma = 1/N\sigma$, $\epsilon = 0.5$, and 4) Gaussian $\gamma = 1/N\sigma$, $\epsilon = 0.1$, where $\epsilon$ is the threshold where no penalty is associated with the training loss. Figure 5 shows the performance and standard deviation for each fold in a 30-fold cross-validation, where each again used random selections of 70%/30% for training/testing.



(a) Longitude

(b) Latitude

Fig. 5: Tested kernels for the regression task

## IV. METHOD 2: SIMPLE CNN

Method 2 uses a simple CNN (Fig. 7), implemented using `Keras` with `Tensorflow 1.6`. The images were rescaled/padded to $1024 \times 1024$ for regression/classification respectively, with the GIST feature vector optionally used as a secondary input (Section III-A) [1].

## A. *Pre-Processing*

The data was pre-processed with latitudes/longitudes regularized between $-1$ to 1 for the regression task, and one-hot-encoded for classification. Images were scaled to between 0 and 1. Finally, the data was randomly split to 80%/20% training/testing sets.

### B. *Various Configurations*

The simple CNN was run in four configurations each for 25 epochs: 1) regression, 2) classification, 3) regression using GIST features (Fig. 7), and 4) classification using GIST features (Fig. 6). See Section VI for method 2's results.

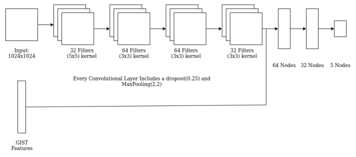


Fig. 6: Method 2 as classification with GIST features

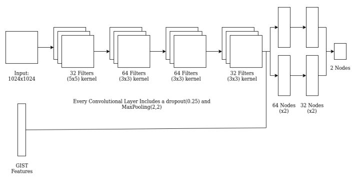


Fig. 7: Method 2 as a regression with GIST features

### C. *Visualizing Results*

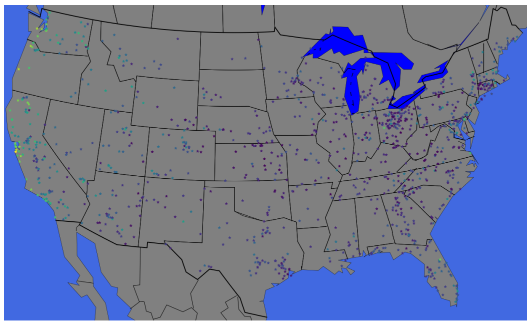


Fig. 8: Method 2 results for the Regression configuration results where darker coloration is more accurate

As can be seen in Fig. 8, when configured as a regression, the simple NN generally avoids labeling images as belonging to the west coast of the United States. Though it will still do so occasionally, this is obviously less than desirable. Notably, there was a lot more overfitting when using GIST features as can be seen in the experimental comparisons. Unlike the first configuration, the regression GIST features captured the west coast labels better.

## V. METHOD 3: RESNET50

The third method utilized was the DCNN using the ResNet50 architecture (Fig. 9), which uses the understanding that increased depth in a traditional feed-forward NN allows for more levels of feature representation and better performance. ResNet was formed in 2015 by He et al. [9] in order to deal with the vanishing gradient problem, where the gradients formed for back propagation continually grow smaller due to repeated use of multiplication between layers. While this eventually leads to either a limit on, or even a reduction to performance, ResNet uses so called shortcuts, or residual connections between the layers, where the input to each block of the CNN is added back to the output, meaning that each layer only learns the features from its predecessor.

### A. *Visualizing the Network*

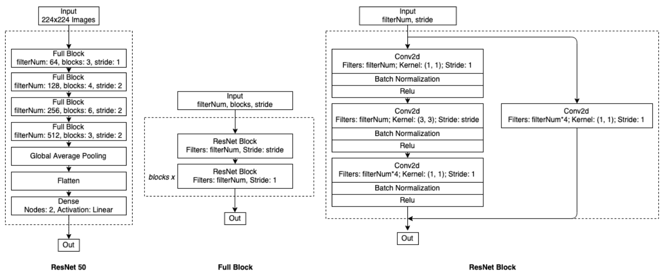

Input
224x224 Images
Full Block
filterNum: 64, blocks: 3, stride: 1
Full Block
filterNum: 128, blocks: 4, stride: 2
Full Block
filterNum: 256, blocks: 6, stride: 2
Full Block
filterNum: 512, blocks: 3, stride: 2
Global Average Pooling
Flatten
Dense
Nodes: 2, Activation: Linear
Out
ResNet 50
Input
filterNum, blocks, stride
ResNet Block
Filters: filterNum, Stride: stride
blocks x
ResNet Block
Filters: filterNum, Stride: 1
Out
Full Block
Input
filterNum, stride
Conv2d
Filters: filterNum; Kernel: (1, 1); Stride: 1
Batch Normalization
Relu
Conv2d
Filters: filterNum; Kernel: (3, 3); Stride: stride
Batch Normalization
Relu
Conv2d
Filters: filterNum*4; Kernel: (1, 1); Stride: 1
Batch Normalization
Relu
Conv2d
Filters: filterNum*4; Kernel: (1, 1); Stride: 1
Out
ResNet Block

Fig. 9: Finalized regression based implementation for the ResNet50 architecture, showcasing the full model on the left and its constituent components on the right

Adopting ResNet for regression, the typical softmax layer was replaced by a dense layer with linear activation. The implementation was formed in `TensorFlow 2.0` with an Adam optimizer and an initial learning rate $\lambda = 1 \times 10^{-5}$. Data augmentation was performed, randomly cropping the inputs to $224 \times 224$ (preventing resolution degradation), with a 50% change of flipping horizontally and/or vertically, as well as a 25% of rotating 0°, 90°, 180°, or 270°. This effectively doubled the inputs, from which a 80/20% split was performed to obtain the respective training/testing sets.

### B. *Visualizing Training*

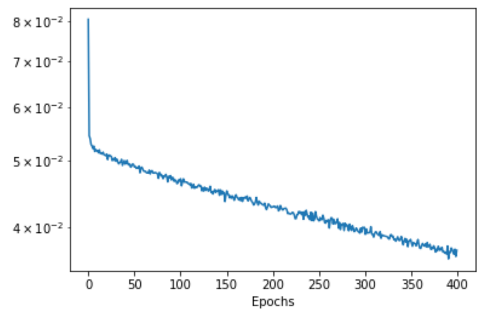


Fig. 10: Training losses for the ResNet50 regression model

The network ran for 400 epochs with a batch size of 50 prior to evaluation. While the model continued to improve its fit to the training data, (Fig. 10) the actual network's ability to generalize well for the test set was observed to begin to decrease thereafter.

### C. Visualizing Results

The visualization of the relative distances between actual and predicted labels can be seen in Fig. 11, with final results shown in Section VI. The minimum distance observed was only 4 miles, with the average being 571 miles.

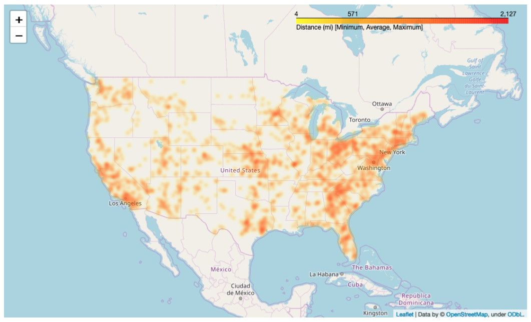


Fig. 11: Heat map of the distances (miles) between the actual and predicted labels with the ResNet50 regression model

## VI. EXPERIMENTAL COMPARISONS

| | **Regression** | | **Classification** |
|---|---|---|---|
| | R-Squared | $\leq$ 200 km of Label | |
| Method 1 | $2.6 \pm 1.1\%$ | N/A | **43.60±0.83%** |
| Method 2 No GIST | 12.9% | 5.77% | 36.5% |
| Method 2 With GIST | 7.26% | 3.9% | 39.1% |
| Method 3 | **32.26%** | **6.0%** | N/A |

TABLE I: Comparison of the employed methods' performance

Combination of the simple CNN approach with the extracted GIST features decreased overall accuracy. This is likely the result of overfitting from the GIST features early and learning to ignore the deep features. Overall, the CNN models performed better than the independent hand-crafted GIST features approach, with ResNet50 giving the best accuracy results. However, the state-of-the-art results reported by Weyland et al. were still significantly better, even though applied to the problem globally [3].

The general observation in the testing results was that urban areas, correlated with greater population densities, are more likely to have similar visual feature sets, such as architecture, or road markings, which in turn lead to more mislabeling. Rural areas where there are visually distinctive weather and/or biodiversity allow for better overall machine learning performance. For example, regions with snowfall/deserts, deciduous/coniferous plant life are generally geospatially distinct and visually distinguishable allowing for better representation and accuracy. It is also noteworthy that given that the set is more likely to contain data from more densely populated urbanized areas (being crowdsoured), it also means that the presence of the "Mapillary" watermark may be being used as a feature. This might in turn lead to misclassifications of rural regions as urban, though to keep to the database's terms of service they were not removed.

## VII. CONCLUSIONS

This work has empirically shown the progression of how machine learning approaches have developed to solve the problem of image geolocation without metadata. There is a clear benefit to moving from hand-crafted features to CNNs to DCNNs, as noted by their respectively increasing levels of accuracy. Further work is still required to develop the chosen networks up to the performance levels reached by current state-of-the-art methods. Particularly, it would be desirous for the ResNet50 implementation to employ a validation set to prevent undue increases in overall bias, removal of skip connections to verify the effect of vanishing gradients, as well as usage of additional data and data augmentation. Ideally this project in general should ensure greater consistency in the input data in terms of image resolution and types (panoramic/flat), or at least perform verification that the varieties are balanced throughout the set. It would be useful in terms of training to understand better what features are being identified by the architecture which may be done through feature map visualization at different layers for the CNN approaches, which would also alleviate concerns that the watermark may be affecting end performance. Additionally, combination of GIST and other feature descriptors such as HOG, RGB, NRI, and NDVI with the CNN architecture might further improve performance.

## SUPPORTING INFORMATION

Code for this project has been made available at: http://github.com/scottstewart1234/EECE5890ProjectCode

## ACKNOWLEDGMENT

The authors would like to thank Mapillary for making their data available and Dr. Ye for the research opportunity.

## REFERENCES

[1] W. Zhang and J. Kosecka. Image based localization in urban environments. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 33–40, June 2006.

[2] Y. Li, N. Snavely, and D. Huttenlocher. Location recognition using prioritized feature matching. In *European conference on computer vision*, pages 791–804. Springer, 2010.

[3] T. Weyand, I. Kostrikov, and J. Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016.

[4] J. Hays and A. Efros. Large-scale image geolocalization. In *Multimodal location estimation of videos and images*, pages 41–62. Springer, 2015.

[5] G. Neuhold, T. Ollmann, Samuel Rota B., and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*, 2017.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society.

[7] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[8] C. Siagian and L. Itti. Rapid biologically-inspired scene classification using features shared with visual attention. *IEEE transactions on pattern analysis and machine intelligence*, 29(2):300–312, 2007.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.