A machine learning pipeline for automated registration and classification of 3D lidar data

Rajagopal, Abhejit, Chellappan, Karthik, Chandrasekaran, Shivkumar, Brown, Andrew

Abhejit Rajagopal, Karthik Chellappan, Shivkumar Chandrasekaran, Andrew P. Brown, "A machine learning pipeline for automated registration and classification of 3D lidar data," Proc. SPIE 10199, Geospatial Informatics, Fusion, and Motion Video Analytics VII, 101990D (1 May 2017); doi: 10.1117/12.2262872

Event: SPIE Defense + Security, 2017, Anaheim, California, United States

A machine learning pipeline for automated registration and classification of 3D lidar data

Abhejit Rajagopal[a,b], Karthik Chellappan[b], Shivkumar Chandrasekaran[a], and Andrew P. Brown[b]

[a]Scientific Computing Group, University of California, Santa Barbara, USA 93106
[b]Toyon Research Corporation, Goleta, USA 93117

ABSTRACT

Despite the large availability of geospatial data, registration and exploitation of these datasets remains a persistent challenge in geoinformatics. Popular signal processing and machine learning algorithms, such as non-linear SVMs and neural networks, rely on well-formatted input models as well as reliable output labels, which are not always immediately available. In this paper we outline a pipeline for gathering, registering, and classifying initially unlabeled wide-area geospatial data. As an illustrative example, we demonstrate the training and testing of a convolutional neural network to recognize 3D models in the OGRIP 2007 LiDAR dataset using fuzzy labels derived from OpenStreetMap as well as other datasets available on OpenTopography.org. When auxiliary label information is required, various text and natural language processing filters are used to extract and cluster keywords useful for identifying potential target classes. A subset of these keywords are subsequently used to form multi-class labels, with no assumption of independence. Finally, we employ class-dependent geometry extraction routines to identify candidates from both training and testing datasets. Our regression networks are able to identify the presence of 6 structural classes, including roads, walls, and buildings, in volumes as big as 8000 m^3 in as little as 1.2 seconds on a commodity 4-core Intel CPU. The presented framework is neither dataset nor sensor-modality limited due to the registration process, and is capable of multi-sensor data-fusion.

Keywords: data mining, fuzzy labeling, wide-area imagery, LiDAR point-cloud, deep neural networks

1. INTRODUCTION

Annotated models of large-scale urban environments are desired in many applications involving scene visualization, analysis, and understanding. These applications include both civilian activities, such as urban planning and geomapping, as well as military activities, such as wide area search and object classification. There are several methods for modeling such environments, but recently Light Detection and Ranging (LiDAR) has emerged as a ubiquitous tool due to its fast and high-bandwidth data collection capabilities. LiDAR is typically used to extract range information of point-scatterers in a scene, and can be combined with multi-modality sensor systems to accurately register and recover 3D structural information in the form of point-cloud data (PCD) and digital surface models (DSM).[1–3]

Although sensor data can theoretically be used to detect, segment, and classify objects in a scene, developing a universally robust algorithm (i.e. one that can handle complex model geometries in various cluttered environments with noise) can be challenging in practice. While, historically, low-level clustering, filtering, and template matching algorithms have been used to detect characteristic features of sensor data, such algorithms typically require manual tuning and, even then, are known to be limited in their application to generic signature discrimination problems.[4] In light of this, and enabled by recent advances in machine learning, many practitioners instead choose to construct generic "universal-approximator" algorithms, such as deep neural networks, which leverage so-called "training" data to optimize a large number of algorithm parameters for the intended task.

Further author information: (Send correspondence to A.R.)
A.R.: E-mail: abhejit@ece.ucsb.edu, Telephone: 1 949 232 6195
A.P.B.: E-mail: abrown@toyon.com, Telephone: 1 805 968 6787

Geospatial Informatics, Fusion, and Motion Video Analytics VII, edited by Kannappan Palaniappan, Peter J. Doucette, Gunasekaran Seetharaman, Anthony Stefanidis, Proc. of SPIE Vol. 10199, 101990D · © 2017 SPIE · CCC code: 0277-786X/17/$18 · doi: 10.1117/12.2262872

Generally, such algorithms (e.g. deep neural networks, non-linear SVMs) have demonstrated incredible results in classical 1D and 2D image processing and classification tasks, but require thousands of well-curated examples in the training period to achieve desirable performance on real-world data collections.[5,6] This is not immediately possible for a majority of publicly available wide-area 3D imagery, thereby frustrating the training and computation of geospatial analytics, which could leverage recent computational advancements made in deep learning and cognitive signal processing.

In this paper, we outline a machine learning pipeline for gathering, registering, and classifying initially unlabeled wide-area geospatial data. As an illustrative example, we demonstrate the training and testing of a convolutional neural network to recognize 3D models in the the publicly available OGRIP 2007 LiDAR dataset[7] using fuzzy labels derived from OpenStreetMap,[8] as well as other datasets available on `OpenTopography.org`.[9] In particular, when auxiliary label information is required, various text and natural language processing filters are used to extract and cluster metadata keywords useful for identifying potential target classes. A subset of these keywords are subsequently used to form multi-class labels, with no assumption of independence. Finally, we employ class-dependent geometry extraction routines to identify candidates from both training and testing datasets. Our regression networks are able to identify the presence of 6 structural classes, including roads, walls, and buildings, in volumes as big as 8000 m^3 in as little as 1.2 seconds on a commodity 4-core Intel CPU. The presented framework is neither dataset nor sensor-modality limited due to the registration process, and can support multi-sensor data-fusion.

The remainder of this paper is organized as follows: Section 2 describes a basic software architecture for referencing various geospatial datasets, while Section 3 outlines a technique for 3D intermodal geospatial registration. Results on 3D LiDAR datasets are presented in Section 4, followed by a discussion in Section 5 of neural network topologies suitable for exploiting GIS data.

2. REFERENCING ACQUIRED SENSOR DATA

In this section, we briefly describe a generic methodology for collecting, parsing, and coarsely indexing large geospatial datasets. A primary requirement is that these datasets possess (at least some) local coordinate information, so that datums can be geo-registered in subsequent portions of the pipeline. A variety of publicly available wide-area geospatial datasets exist. In this project, for example, we have written scripts to automatically locate and download (i.e. "scrape") large portions of `OpenTopography.org` (OT).

The first step is to locate the data-source and identify the modality of the data collection. OT hosts a variety of imagery, including airborne LiDAR point-cloud data (PCD), digital elevation models (DEMs), bathymetric data, and satellite imagery. This data is also available in various formats, including ASCII/LAS PCD, raster-scan images, and Google Earth models (KML). It is important here to understand how these files were assembled, since many datasets (e.g. Google Earth models) are derived from a combination of sensor data sources (e.g. airborne LiDAR, satellite imagery, etc), and understanding this hierarchy can be essential to building algorithms that have real-time implementations. In this work, we focus on 3D point-cloud data recovered from single- and multi-return LiDAR sensor systems ($\lambda \sim 400 - 1,000$nm).

One typical issue, often encountered when dealing with multiple datasets, is the differing choice of file-structure or directory layout, which makes it difficult for automatic parsing algorithms to locate and optimally index sensor data. In these cases, we find that linearizing the entire directory tree can often help in understanding the data layout, and picking an optimal indexing scheme. When vital metadata is included in the directory structure, we find that simply annotating terminating nodes with this information prior to linearizing is often preferable to writing custom routines for exhaustively navigating the directory structure.

As an example, in the 2007 OGRIP LiDAR data collection, portions of the geography (recovered as point-cloud data) are blocked into 1.5km x 1.5km tiles, further organized by county. While a trivial implementation would simply concatenate all the tiles together prior to processing, this is computationally prohibitive–and precisely the reason the data was segmented in the first place. To avoid this, we instead develop an indexing scheme using the bounding-box of each tile. This can be extremely useful for quickly identifying the (overlapping) files which contain data of particular geographical location, especially when this data is collected from various sensors. In particular, because 2D maps cannot be naturally ordered, the referencing data of a particular geographic location requires at least 1 expensive search, 1 expensive but strided search, and at most 3 fast local searches if the data-structure implementation uses the tile-size as the basic block (Alg. 1).

Algorithm 1 Simple Geospatial Hash Function

```
procedure FINDDATANODES($bbox$, $d$)                        ▷ Given object bounding-box, and data dictionary
    $x \leftarrow$ findIndex( $d.keys()$, $bbox$.LAT[0] )                          ▷ Locate latitude index
    $y \leftarrow$ findIndex( $d.keys()[x]$, $bbox$.LNG[0] )                     ▷ Locate longitude index
    $neighbors \leftarrow \{ (m,p)$ for $m \in x \pm \{1,0\}$ for $p \in y \pm \{1,0\} \}$   ▷ (Explicitly) identify neighbors
    $nodes \leftarrow$ findOverlap( $d$, $bbox$, $neighbors$ )      ▷ Identify neighbors containing object.
    return $nodes$
end procedure
```

3. REGISTERING MULTIMODAL DATASETS

Once data is coarsely geographically indexed, datums within each block must be finely registered. In particular, in this work we register portions of LiDAR point-cloud data with relatively-coarse text/metadata annotations acquired from OpenStreetMap (OSM). The primary motivation for this is the lack of relevant annotation labels (i.e. ground-truth) for the LiDAR PCD, which prevents the training and testing of machine learning algorithms for tasks such as 3D automatic target recognition (ATR). While this issue is not unique to LiDAR data (and has, in fact, frustrated the comprehensive evaluation of various 3D model recognition and search algorithms[10–12]), here we are exploiting the fact that both LiDAR data and semantic metadata/annotations exist over large areas, providing the large number of samples required for developing deep, robust geospatial analytics algorithms.

To derive these annotations, we first query, collect, and parse nodes on OpenStreetMap as follows (Alg. 2):

Algorithm 2 Assign labels to OpenStreetMap nodes.

```
procedure GATHEROSMNODES($data$, $bags$)     ▷ Given local datum, and a bag of keywords for each class.
    $bbox \leftarrow$ project( $data$, 'WSG-84' )              ▷ Transform local coordinates to LAT,LNG
    $query \leftarrow$ formQuery( $bbox$, 'PLACE','WAY' )                   ▷ Form SQL/API query string
    $nodes \leftarrow$ queryOSM( $query$ )                         ▷ GET request to OpenStreetMap.org
    for $n_k \in all\_nodes$ do
        $keywords \leftarrow$ filterNLP( $n_k$.keys() + $n_k$.values() )            ▷ Extract relevant keywords
        $pdf \leftarrow$ computeFraction( $keywords$, $bags$)                ▷ Compute fuzzy class membership
        $y_{k1} \leftarrow$ identifyPeaks( $pdf$ )                            ▷ Assign most likely "fuzzy" label
        $y_{k2} \leftarrow$ identifyCategories( $n_k$.keys() )                  ▷ Assign label based on OSM keys
        if $|y_{k1}| + |y_{k2}| \ngtr 0$ then                                ▷ If no label can be determined,
            $nodes \leftarrow nodes - n_k$                       ▷ then remove node from consideration
        end if
    end for
    return $nodes$, $y$
end procedure
```

The natural language processing (NLP) filter in the function above (`filterNLP`) can be a tailored to extract a user's classes of interest. In our work, we developed this routine by extracting keywords from thousands of nodes across several counties in Ohio, and clustering them to identify the most common among them. Upon manual inspection, this gives an indication of which keywords should be used for identifying relevant target classes, and how many such examples can be expected to be collected. Of course, various other manual and automatic NLP algorithms are possible, but the basic idea is that an OSM node's keys (e.g. database columns) and values can be parsed to extract either hard (column-based) or soft (value-based) annotations, depending on the application.

Once the nodes in a given region are determined, we densely annotate the imagery that corresponds to the geo-referenced node by leveraging any reference coordinates or information in the OSM node's "geometry" attribute. For example, many OSM nodes include both a reference coordinate (`LAT`,`LNG`), as well as Shapefile information, which can be used to define small bounding boxes along or within the interior of the referenced shape and densely annotate corresponding geographically registered sensor-data or imagery.

We note here that a similar approach was taken by Johnson et al. in the 2016 *DeepOSM* project, although their filtering and label assignment routine relies on the OSM node's designation as a *place*, *way*, or *relation*.[13] Specifically, *DeepOSM* registered overhead satellite imagery with OSM metadata to determine whether a node's OSM designation as a *way* was indeed accurate (binary classification). Here, we have extended this approach to allow the semantic labeling of wide-area imagery for a large number of classes, including *highway/road*, *building*, *bridge*, *waterway*, and *forest*, as well as synthetic objects such as civilian *vehicles*. In particular, we have developed an interface for labeling nodes using custom natural language processing algorithms, and transferring these annotations directly to cross-modality geospatially-registered sensor data.

As an example, for Cuyahoga county in Ohio, we were able to collect and label a more than $10,000$ examples with corresponding LiDAR data. Since the OGRIP data is sampled fairly coarsely (e.g. roughly 2.1m spacing between points), the collected examples were upsampled via bicubic interpolation of their 2D overhead projections. Some examples of various OSM *highway* and *building* nodes are shown in Figure 1, below.

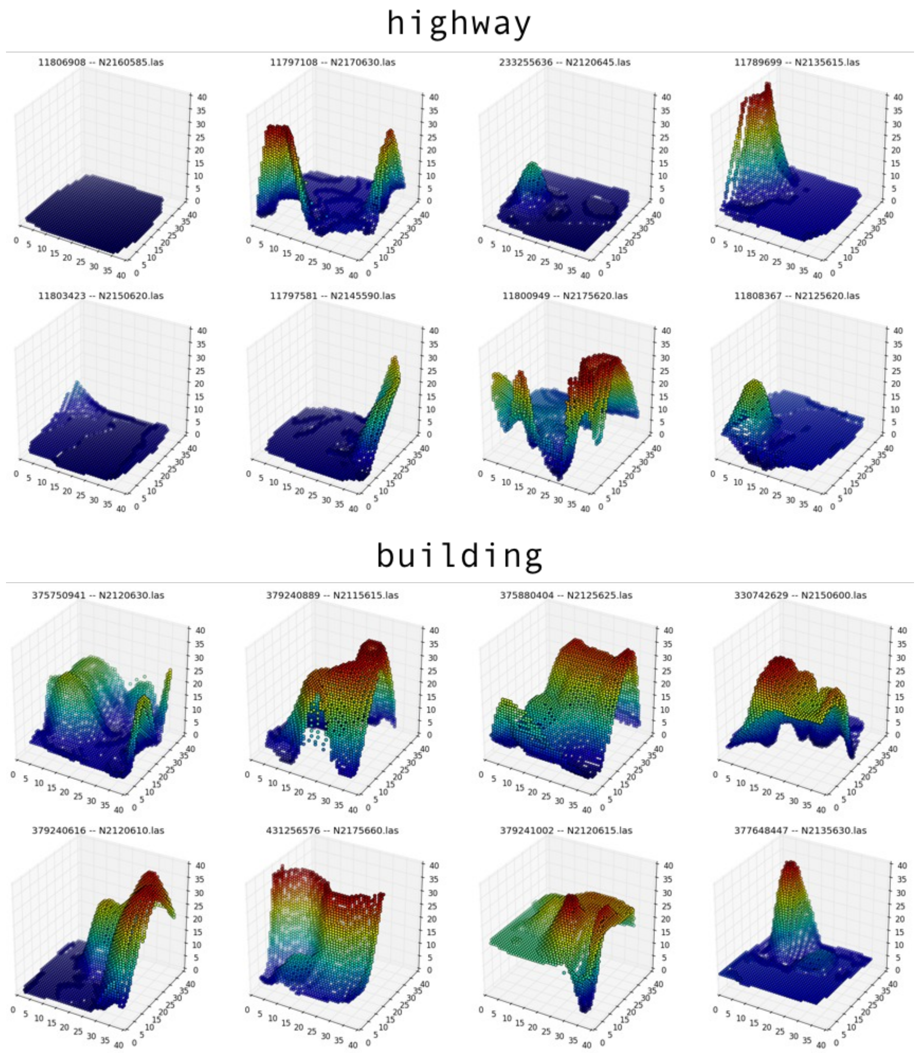

Figure 1. Examples of some *highway* and *building* examples identified automatically from OSM and rendered using interpolated OGRIP LiDAR data.

One issue with the described method, however, is that there is a mismatch between the 2007 OGRIP LiDAR data collection, and the presently available OSM annotations. Although there are mechanisms to filter the annotations by date, OpenStreetMap was only conceived in 2009, and did not include many annotations at the time. A manual sampling of the datset revealed that roughly 7-8% of the LiDAR data was mislabeled, which is an acceptable gure for training a classifier for some applications, but may have implications on the upper bound of achievable classifier performance. Some examples of mislabelled buildings and roads are shown in Figure 2 below.

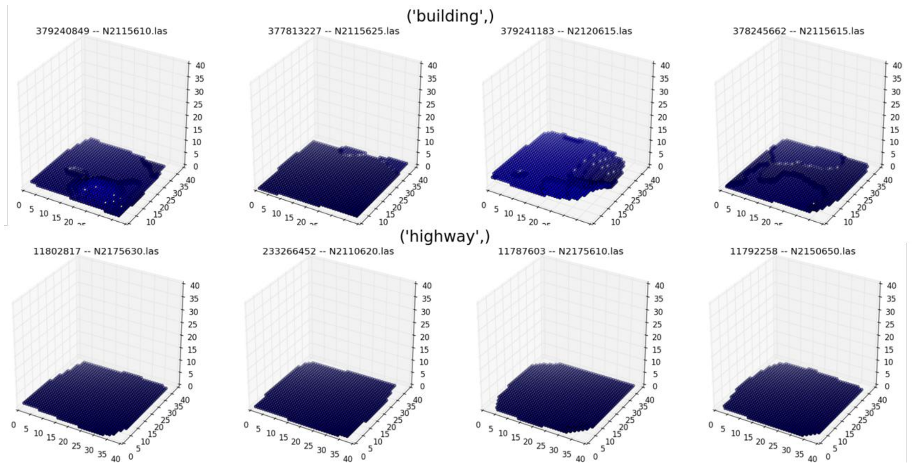

Figure 2. Some examples from the *highway* and *building* classes do not contain any structural information. While this is obvious for the *building* class, even the highway class must contain some degree of structure (e.g. curbs, center-dividers, etc) to be classified as a highway, as opposed to flat-ground or clutter.

4. 3D COGNITIVE LIDAR

To the demonstrate the efficacy of this machine learning pipeline, we train and test a deep neural network on a multi-label 3D classification task. Our network is built in `Tensorflow` and composed of 8 layers (4 convolutional, and 4 fully-connected), resulting in a network of over $150,000$ parameters. A desirable result was reached for a number of classes after training for 5000 iterations using stochastic gradient descent with the Adam's Optimizer, and attempting a few different initial learning rates. Figure 3 depicts convergence for the *highway*, *building*, and *vehicle* classifiers, while Figure 4 depicts the results for the *vehicle* classifier using three different initial learning rates. It is interesting to note that while the largest learning rate (1e$-$3) seems the most stochastic initially, it also achieves the best performance over the training period for all three presented classes.

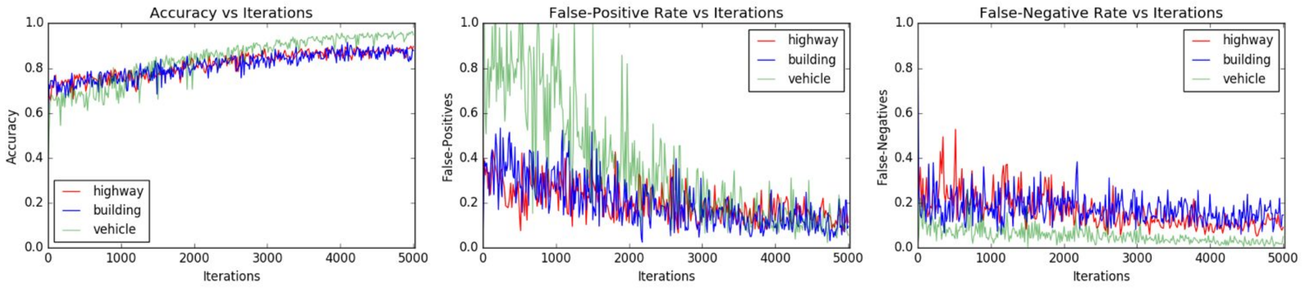

Figure 3. Training a deep convolutional neural network to classify voxelized 3D LiDAR blocks covering $20m$ x $20m$ geographically. Shown here is the network performance over 5000 iterations for the classes: *highway*, *building*, and *vehicle*.

While the training accuracy can be somewhat hard to gauge, at least initially, we developed an additional technique to analyse the general behavior of neural networks on a given classification task. If the classification network can be thought of as a map from the input space ($R^{[20,20,20]}$) to the output space ($R^1 \in [0,1]$) for binary

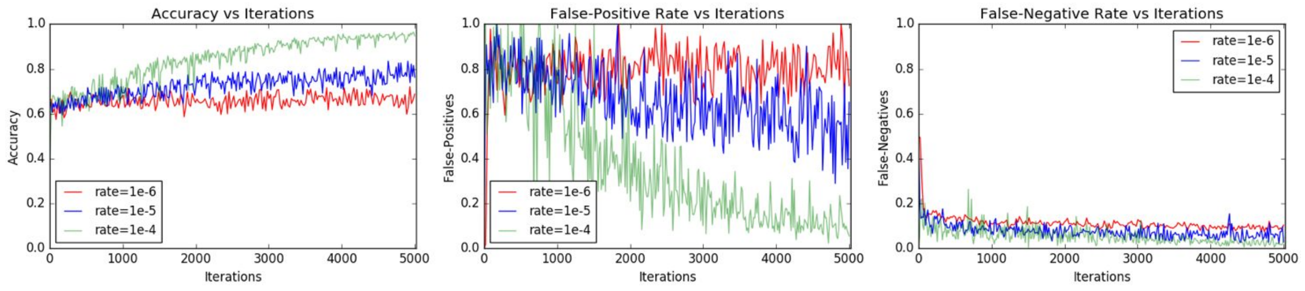

Figure 4. The performance of the neural network *vehicle* classifier varies as much as 15% when starting with different learning rates. The Adam's Optimizer is used with stochastic gradient descent on mini-batches of 200-300 examples each.

classification, then we can measure the relative confidence of the neural network by examining its continuous-valued output. In particular, if a 50-50 batch of samples (i.e. 50% positive label, 50% negative label) are fed to the network, we expect that a perfectly confident network will return a list distributed as 50% positive (1) labels and 50% negative (0) labels. Further, output values that lie in the range $(0, 1)$ can indicate a relative level of confidence in the output decision. One such interpretation might yield the following formula:

$$c(x) = 100\% \times \frac{|nn(x) - 0.5|}{0.5} \tag{1}$$

where $c(\cdot)$ is the confidence in percent, $nn(\cdot)$ is the neural network, and 0.5 is the arbitrary threshold chosen to divide positive classifications from negative classifications. This, and similar, expressions have been used in various instances to gauge the accuracy of the classifier, and in some cases is even used as a metric in the optimization algorithm.[14]

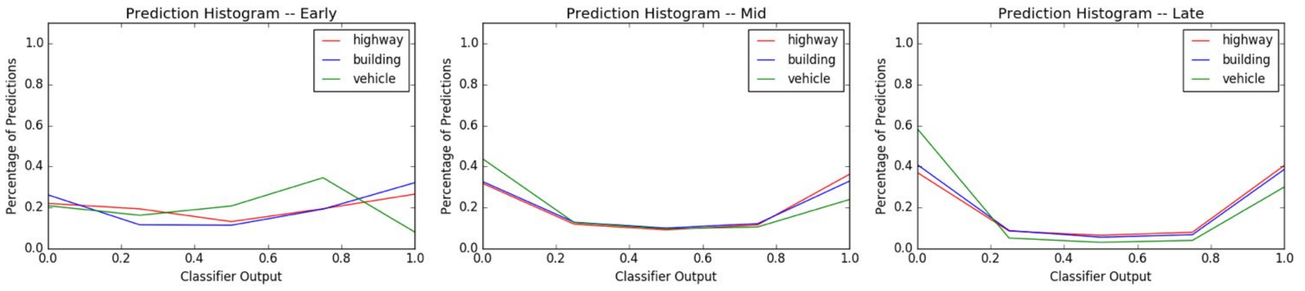

Figure 5. The output prediction histogram of a 50-50 batch can indicate the relative performance or confidence of the classification network. Shown here is the average histogram for the first 500 iterations (left), iterations 2000-3000 (middle), and final 500 iterations (right) of a 5000 iteration training routine. The tool can be used to identify under-confident networks (left), as well as moderately trained networks (middle), and over-trained networks (right, *vehicle* class).

Here, we extend this technique by computing a histogram of the continuous valued outputs produced by the network in response to a 50-50 batch. As mentioned, an ideal classifier will have a two discrete delta functions at 0 and 1, while suboptimal networks will split this energy over the various intermediate values. In our work, we computed the output "prediction" histogram on large validation batches during training to visualize the relative network performance. As expected, the network tends towards the ideal distribution as the training accuracy increases (Fig. 5). One interesting feature of this qualitative analysis is the ability to gauge the level of over-training by noting the asymmetry in the histogram distribution after the network has attained high accuracy.

5. CONCLUSION

In summary, we have presented a method of deriving and classifying initially unannotated wide-area imagery, such as LiDAR point-cloud data, using publicly accessible metadata from OpenStreetMap (OSM) servers. The sensor-data geo-registration process can be sufficiently accurate, but annotations are eventually limited by the granularity of OSM annotations. In particular, when OSM nodes contain Shapefile information, every sensor datapoint can be annotated with a set of labels. When Shapefile information is not available, data must be instead annotated at the tile or sub-block level, which can be thought of as defining an arbitrarily-sized box around which the annotation is valid–this approach corresponds exactly to the case of 3D LiDAR PCD. There is benefit in both levels of granularity, however; although the dense annotation may enable the use of fully convolutional neural networks, which can provide powerful target localization capabilities, the coarser block-level annotation might be more resilient to noise, since target classes are forced to be identified in the presence of a significant amount of ambiguously-labelled clutter. In the case of sparse OSM annotations, the block-level approach is recommended, since this generally is sufficient to match the resolution of available coordinate/geometry information.

ACKNOWLEDGMENTS

The authors would like to thank the Air Force Research Laboratory for supporting this work under contract number FA8650-16-M-1784, and would like to thank Dr. Benjamin Robinson and Dr. Bernard Abayowa for their helpful suggestions and guidance.

REFERENCES

[1] B. Petzold, P. Reiss, and W. Stössel, "Laser scanning-surveying and mapping agencies are using a new technique for the derivation of digital terrain models," *ISPRS Journal of Photogrammetry and remote Sensing* **54**(2), pp. 95–104, 1999.

[2] B. O. Abayowa, *Automatic Registration of Optical Aerial Imagery to a LiDAR Point Cloud for Generation of Large Scale City Models*. PhD thesis, University of Dayton, 2013.

[3] S. Sun and C. Salvaggio, "Aerial 3d building detection and modeling from airborne lidar point clouds," *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal of* **6**(3), pp. 1440–1449, 2013.

[4] R. Ohbuchi, K. Osada, T. Furuya, and T. Banno, "Salient local visual features for shape-based 3d model retrieval," in *Shape Modeling and Applications, 2008. SMI 2008. IEEE International Conference on*, pp. 93–102, IEEE, 2008.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[6] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, *et al.*, "Large scale distributed deep networks," in *Advances in neural information processing systems*, pp. 1223–1231, 2012.

[7] I.-C. Lee, L. Cheng, and R. Li, "Optimal parameter determination for mean-shift segmentation-based shoreline extraction using lidar data, aerial orthophotos, and satellite imagery," in *ASPRS Conference*, 2010.

[8] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive Computing* **7**(4), pp. 12–18, 2008.

[9] S. Krishnan, C. Crosby, V. Nandigam, M. Phan, C. Cowart, C. Baru, and R. Arrowsmith, "Opentopography: a services oriented architecture for community access to lidar topography," in *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, p. 7, ACM, 2011.

[10] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On visual similarity based 3d model retrieval," in *Computer graphics forum*, **22**(3), pp. 223–232, Wiley Online Library, 2003.

[11] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool, "Hough transform and 3d surf for robust three dimensional classification," in *European Conference on Computer Vision*, pp. 589–602, Springer, 2010.

[12] M. Carlberg, P. Gao, G. Chen, and A. Zakhor, "Classifying urban landscape in aerial lidar using 3d shape analysis," in *Image Processing (ICIP), 2009 16th IEEE International Conference on*, pp. 1701–1704, IEEE, 2009.

[13] A. Johnson, "Deeposm." `https://github.com/trailbehind/DeepOSM`, 2016.

[14] J. Snoek, O. Rippel, K. Swersky, R. Kiros, N. Satish, N. Sundaram, M. M. A. Patwary, M. Prabhat, and R. P. Adams, "Scalable bayesian optimization using deep neural networks.," in *ICML*, pp. 2171–2180, 2015.