Learning to Label Aerial Images from Noisy Data

Volodymyr Mnih VMNIH@CS.TORONTO.EDU
Department of Computer Science, University of Toronto

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science, University of Toronto

Abstract

When training a system to label images, the amount of labeled training data tends to be a limiting factor. We consider the task of learning to label aerial images from existing maps. These provide abundant labels, but the labels are often incomplete and sometimes poorly registered. We propose two robust loss functions for dealing with these kinds of label noise and use the loss functions to train a deep neural network on two challenging aerial image datasets. The robust loss functions lead to big improvements in performance and our best system substantially outperforms the best published results on the task we consider.

1. Introduction

Information extracted from photographs of the earth's surface that were taken by airborne sensors has found applications in a wide range of areas including urban planning, crop and forest management, disaster relief, and climate modeling. Relying on human experts for extracting information from aerial imagery is both slow and costly, so automatic aerial image interpretation has received much attention in the remote sensing community. So far, there are only a few, semi-automated systems that operate in limited domains (Mayer, 2008), but recent applications of large-scale machine learning to aerial image interpretation have produced object detectors with impressive levels of accuracy on challenging high-resolution data (Kluckner & Bischof, 2009; Kluckner et al., 2009; Mnih & Hinton, 2010).

In machine learning applications, aerial image interpretation is usually formulated as a pixel labeling task. The goal is to produce either a complete semantic segmentation of an aerial image into classes such as building, road, tree, grass, and water (Kluckner & Bischof, 2009; Kluckner et al., 2009) or a binary classification of the image for a single object class (Dollar et al., 2006; Mnih & Hinton, 2010). In both scenarios, the availability of accurately labeled data for training tends to be the limiting factor. Hand-labeled data tends to be reasonably accurate, but the cost of hand-labeling and the lack of publicly available hand-labeled datasets strongly restricts the size of the training and test sets for aerial image labeling tasks.

At present, maps of many major cities not only provide the locations of most roads and parks, but also the locations of buildings. So one alternative to using hand-labeled data is to use maps from projects such as OpenStreetMap for constructing the labels. For object types covered by these maps, it is now possible to construct datasets that are much larger than the ones that have been hand-labeled. While the use of these larger datasets has improved the performance of machine learning methods on some aerial image recognition tasks (Mnih & Hinton, 2010), datasets constructed from maps suffer from two types of label noise:

- **Omission noise** occurs when an object that appears in an aerial image does not appear in the map. This is the case for many buildings (even in major cities) due to incompleteness of the maps. It is also true for small roads and alleys, which tend to be omitted from maps, often with no clear criterion for when they should be omitted.
- **Registration noise** occurs when the location of an object in a map is inaccurate. Such errors are quite common because not requiring pixel level accuracy makes maps cheaper to produce for human experts without significantly reducing their usefulness for most purposes.

The presence of these kinds of errors in the training labels significantly reduces the accuracy of classifiers trained on this data.

Appearing in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

(a) (b)

Figure 1. Road locations derived from a map are shown in red. (a) Example of omission noise. (b) Example of registration noise.

In this paper, we show how one can deal with the presence of both kinds of noise in the training labels. We present two robust loss functions, one that reduces the effect of omission errors on the resulting classifier, and one that accounts for both omission and registration errors in the training data. After incorporating these improvements into a deep learning framework we obtain a substantial improvement in the state of the art on the largest and most challenging road detection dataset.

2. Problem Formulation

We will consider aerial image labeling tasks with binary labels, where the goal is to label all pixels belonging to an object class of interest with 1's and all other pixels with 0's. Road and building detection are two examples of such problems. We adopt a problem setup that closely resembles the patch-based approach used for road detection in (Mnih & Hinton, 2010).

Let $\mathbf{S}$ be an aerial/satellite image and $\tilde{\mathbf{M}}$ be the corresponding map image of equal size produced from the given map[1]. $\tilde{\mathbf{M}}_{i,j} = 1$ whenever the pixel at location (i, j) contains the object of interest and $\tilde{\mathbf{M}}_{i,j} = 0$ otherwise. The goal is to learn to predict patches of map $\tilde{\mathbf{M}}$ from patches of $\mathbf{S}$, and following a probabilistic approach, we model the distribution

$$P(n(\tilde{\mathbf{M}}_{i,j}, w_m)|n(\mathbf{S}_{i,j}, w_s)), \tag{1}$$

where $n(\mathbf{I}_{i,j}, w)$ is the $w \times w$ patch of image $\mathbf{I}$ centered at location (i, j). Typically w_m is set to be smaller than w_s because some context is required to predict the value of a map pixel. While w_m can be set to 1 to predict one pixel at a time, it is generally more efficient to predict a small patch of labels from the same context.

[1]We use the same approach for generating soft binary maps from vector maps described in (Mnih & Hinton, 2010).

3. Deep Learning Framework

To simplify notation, we will use vectors $\mathbf{s}$ and $\tilde{\mathbf{m}}$ to denote the aerial image patch $n(\mathbf{S}_{i,j}, w_s)$ and the map patch $n(\tilde{\mathbf{M}}_{i,j}, w_m)$ respectively. Following earlier work (Mnih & Hinton, 2010) we assume conditional independence of the map pixels and model the map distribution

$$p(\tilde{\mathbf{m}}|\mathbf{s}) = \prod_{i=1}^{w_m^2} p(\tilde{m}_i|\mathbf{s}) \tag{2}$$

using a neural network. We assume that each $p(\tilde{m}_i|\mathbf{s})$ is a Bernoulli distribution whose mean value is determined by the ith output unit of the neural network. We will refer to this as the noise free model.

3.1. Network Architecture

Unlike earlier work, we use a deep neural network to model the map distribution. The input to the neural network is a w_s by w_s patch of an aerial image encoded in the RGB color space, while the output is a w_m by w_m map patch. The input layer is followed by three hidden layers all of which make use of the rectified linear activation function (Nair & Hinton, 2010), for which the output is defined as $\max(0, input)$. Rectified linear units have been found to be better than logistic units on various image classification tasks and we find that this advantage also exists on image labeling tasks (Nair & Hinton, 2010).

The first two hidden layers in our network are locally connected layers, in which each hidden unit is connected to only a small subset of the input units. To precisely define the connectivity pattern, assume that the input units of a locally connected layer make up a $w_{in} \times w_{in}$ image, possibly consisting of multiple channels. The input image is divided into evenly spaced filter sites by moving a $w_f \times w_f$ window over the image by a stride of w_{str} vertically and horizontally, for a total of $((w_{in} - w_f)/w_{str} + 1)^2$ filters sites. A different set of f filters of size $w_f \times w_f$ and consisting of the same number of channels as the input image is applied at each filter site. Hence, a single locally connected layer results in $f \cdot ((w_{in} - w_f)/w_{str} + 1)^2$ hidden units. The hidden units of one locally connected layer can then act as the input to another locally connected layer by viewing the hidden units as a square image with f channels and width $(w_{in} - w_f)/w_{str} + 1$. Unlike a convolutional or tiled net, there is no weight-sharing of any kind.

The third hidden layer is fully connected, with each unit connected to every unit in the preceding hidden layer. The output layer consists of w_p^2 logistic units for

which the output is $1/(1+\exp(-input))$. Typically, $w_p = w_m$ and each output unit models the probability that the corresponding pixel in the w_m by w_m output map patch belongs to the class of interest.

The values of the parameters such as the number of filters f, their width w_f, and stride w_{str} should vary from problem to problem. The settings we use in our experiments are described in Section 6.

3.2. Preprocessing

We preprocess each input patch by subtracting the mean value of the pixels in that patch from all pixels and then dividing by the standard deviation found over all pixels in the dataset. This type of preprocessing achieves some contrast normalization between different patches.

3.3. Learning

We learn the parameters of the neural network by minimizing the negative log likelihood of the training data. For the model given in Equation 2 the negative log likelihood takes the form of a cross entropy between the patch $\tilde{\mathbf{m}}$ derived from the given map and the predicted patch $\hat{\mathbf{m}}$

$$\sum_{i=1}^{w_m^2} (\tilde{m}_i \ln \hat{m}_i + (1-\tilde{m}_i)\ln(1-\hat{m}_i)) \,. \qquad (3)$$

We optimize this objective function using mini-batched stochastic gradient descent with momentum.

3.4. Discussion of Architecture

Our general architecture has some similarity to various convolutional architectures that have recently become popular (Lee et al., 2009; Kavukcuoglu et al., 2010). In fact, the locally connected layers can be seen as convolutional layers with untied weights. Weight-sharing in convolutional architectures is advantageous on smaller datasets because it helps reduce overfitting by restricting the number of parameters, but we do not need such a restriction because the abundance of labels, combined with random rotations, allows us to avoid overfitting by training on millions of labeled aerial image patches. Like convolutional architectures, our locally connected architecture is computationally and statistically more efficient than a fully connected architecture.

3.5. Unsupervised Pretraining

Initializing neural networks using unsupervised learning methods is known to improve performance on a variety of vision tasks (Hinton et al., 2006; Krizhevsky, 2011; Mnih & Hinton, 2010). We use unsupervised pretraining to initialize the deep neural network following the approach described in (Nair & Hinton, 2010) for training Restricted Boltzmann Machines with rectified linear units.

4. Dealing With Omission Noise

Omission noise, as shown in Figure 1(a), occurs when some map pixels are labeled as not belonging to the object class of interest when they, in fact, do. When trained on data containing a substantial number of such pixels a classifier will be penalized for correctly predicting the value of 1 for pixels affected by omission noise. This will cause a classifier to be less confident and potentially increase the false negative rate.

We propose using a robust loss function that explicitly models asymmetric omission noise in order to reduce its effect on the final classifier. The noise-free model of the data from Equation 2 assumes that the observed labels $\tilde{\mathbf{m}}$ are generated directly from the aerial image $\mathbf{s}$. In order to model label noise, we assume that a true, uncorrupted, and unobserved map patch $\mathbf{m}$ is first generated from the aerial image patch $\mathbf{s}$ according to some distribution $p(\mathbf{m}|\mathbf{s})$. The corrupted, observed map $\tilde{\mathbf{m}}$ is then generated from the uncorrupted $\mathbf{m}$ according to a noise distribution $p(\tilde{\mathbf{m}}|\mathbf{m})$. For simplicity, our omission model assumes that conditioned on $\mathbf{m}$, all components of $\tilde{\mathbf{m}}$ are independent and that each $\tilde{m}_i$ is independent of all m_j for $j \neq i$. The observed map distribution that corresponds to this model can then be obtained by marginalizing out $\mathbf{m}$, leading to

$$p(\tilde{\mathbf{m}}|\mathbf{s}) = \sum_{\mathbf{m}} p(\tilde{\mathbf{m}}|\mathbf{m})p(\mathbf{m}|\mathbf{s}) \qquad (4)$$

$$= \prod_{i=1}^{w_m^2} \sum_{m_i} p(\tilde{m}_i|m_i)p(m_i|\mathbf{s}). \qquad (5)$$

The noise distribution $p(\tilde{m}_i|m_i)$ is assumed to be the same for all pixels i, and is determined by parameters

$$\theta_0 = p(\tilde{m}_i = 1|m_i = 0) \text{ and,}$$
$$\theta_1 = p(\tilde{m}_i = 0|m_i = 1).$$

For modeling omission noise we set $\theta_0 \ll \theta_1$ because the probability that the observed label $\tilde{m}_i$ is 1 given that the true label m_i is 0 should be very close to 0, while the probability that the observed $\tilde{m}_i$ is 0 given that the true label m_i is 1 should still be small but not as close to 0 as θ_0.

We refer to this model as the asymmetric Bernoulli noise model, or the ABN model for short. In the noise-free scenario we described in the previous section, the

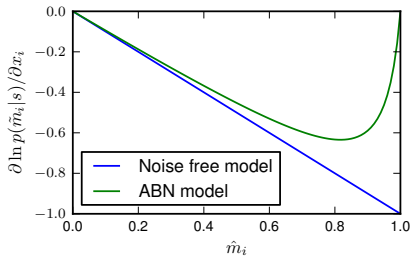


Figure 2. The derivative of the log probability with respect to the input to the ith output unit for varying predictions $\hat{m}_i$ is shown. The observed value $\tilde{m}_i$ is set to 0 while the parameters of the ABN model are $\theta_0 = 0.001$ and $\theta_1 = 0.05$. The noise free model penalizes incorrect predictions more than the ABN model, which penalizes incorrect but confident predictions less.

map distribution in Equation 2 was modelled directly by a deep neural network. In the noisy setting, we can use the neural network to model the true map distribution $p(\mathbf{m}|\mathbf{s})$. Learning can still be done efficiently by minimizing the negative log probability of the training data under the ABN model given in Equation 5. Since the ABN model factorizes over the pixels i and there is only a single Bernoulli latent variable m_i for each pixel i, the derivative of the negative log probability can be found directly.

In the noise-free scenario, the derivative of the negative log probability with respect to the input to the ith output unit of the neural network takes the form $\tilde{m}_i - \hat{m}_i$. The learning procedure is trying to make the prediction $\hat{m}_i$ close to the observed label $\tilde{m}_i$. Under the ABN model, this derivative takes the form $p(m_i = 1|\tilde{m}_i, \mathbf{s}) - \hat{m}_i$. Hence, the learning procedure is trying to make the prediction $\hat{m}_i$ close to the posterior probability that the unobserved true label m_i is 1. This has the effect that the neural network gets penalized less for making a confident but incorrect prediction. Figure 2 demonstrates how the derivatives for the noise-free and the ABN models differ as a function of the prediction $\hat{m}_i$.

5. Dealing With Registration Noise

Registration noise occurs when an aerial image and the corresponding map are not perfectly aligned. As shown in Figure 1(b), the error in alignment between the map and the aerial image can vary over the dataset and cannot be corrected by a global translation. Our approach attempts to eliminate registration errors using local translations of the labels.

We extend the robust loss function we introduced in the previous section for dealing with omission noise to also handle local registration errors. As with the ABN model, we introduce a generative model of the observed map patches. On a high level, the generative model works by first generating an uncorrupted and perfectly registered map from the aerial image, then selecting a random subpatch of the true map, and finally generating the observed map by corrupting the selected subpatch with asymmetric noise. More formally, the generative process is as follows:

1) An uncorrupted and perfectly registered true map patch $\mathbf{m}$ of size $w_{m'} \times w_{m'}$ is generated from $\mathbf{s}$ according to $p(\mathbf{m}|\mathbf{s})$. We set $w_{m'} = w_m + 2t_{max}$ where t_{max} is the maximum possible registration error/translation between the map and aerial image measured in pixels.

2) A translation variable t is sampled from some distribution $p(t)$ over $T+1$ possible values $0, \ldots, T$. In this paper, we use $T = 8$, where $t = 0$ corresponds to no translation while $1, \ldots, T$ index 8 possible translations by t_{max} pixels in the vertical and horizontal directions as well as their combinations (see Figure 3).

3) An observed map is sampled from the translational noise distribution

$$p(\tilde{\mathbf{m}}|\mathbf{m}, t) = p(\tilde{\mathbf{m}}|Crop(\mathbf{m}, t)) \quad (6)$$

$$= \prod_{i=1}^{w_m^2} p_{ABN}(\tilde{m}_i|Crop(\mathbf{m}, t)_i), \quad (7)$$

where $Crop(\mathbf{m}, t)$ selects a w_m by w_m subpatch from the $w_{m'}$ by $w_{m'}$ patch $\mathbf{m}$ according to the translation variable t as shown in Figure 3, and $p_{ABN}(\tilde{m}_i|m_i)$ is the pixelwise asymmetric binary noise model defined in the previous section.

For simplicity we assume that $p(t = i) = (1 - p(t = 0))/T$ for all $i \neq 0$ and parameterize $p(t)$ using only a single parameter $\theta_t = p(t = 0)$. Hence, we use a total of four parameters: t_{max}, θ_t, and two parameters needed to define $p_{ABN}(\tilde{m}_i|m_i)$. We refer to this generative model as the translational asymmetric binary noise model, or the TABN model for short.

5.1. Learning

The observed map distribution under the TABN model is given by

$$p(\tilde{\mathbf{m}}|\mathbf{s}) = \sum_{t=0}^{T} p(t) \sum_{\mathbf{m}} p(\tilde{\mathbf{m}}|\mathbf{m}, t) p(\mathbf{m}|\mathbf{s}). \quad (8)$$

We set the parameters of $p(t)$ and $p(\tilde{\mathbf{m}}|\mathbf{m}, t)$ using a validation set and learn the parameters of $p(\mathbf{m}|\mathbf{s})$ by minimizing the negative log likelihood in Equation 8 using the EM-algorithm. The required EM updates can be performed efficiently.

M-step: Since $p(\mathbf{m}|\mathbf{s})$ is modelled by a neural net-

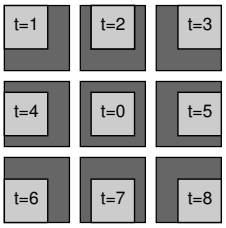


Figure 3. Demonstration of the $Crop(\mathbf{m}, t)$ function used in this paper. For each dark gray patch representing $\mathbf{m}$, the lighter gray subpatch highlights the area cropped by $Crop(\mathbf{m}, t)$ for the stated translation parameter t.

work, we cannot do a full M-step and instead do an approximate partial M-step by doing a single gradient descent update of the neural network parameters on a mini-batch of training cases. The required derivative of the expected log likelihood is

$$\frac{\partial}{\partial x_i} \sum_t \sum_{\mathbf{m}} p(\mathbf{m}, t|\tilde{\mathbf{m}}, \mathbf{s}) \ln p(\mathbf{m}|\mathbf{s}) = p(m_i = 1|\tilde{\mathbf{m}}, \mathbf{s}) - \hat{m}_i$$

where $\hat{m}_i$ is value of the ith output unit of the neural network and x_i is the input to the ith output unit. The updates for all weights of the neural network can be computed from the above equation using backpropagation.

E-step: The role of the E-step is to compute $p(m_i|\tilde{\mathbf{m}}, \mathbf{s})$ for use in the M-step, and as we will show, this computation can be done in time $T \cdot w_m^2$ by exploiting the structure of the noise model.

We first define C_t to be the set of indices of pixels of $\mathbf{m}$ that are cropped for transformation t. Since this set will have w_m^2 entries we will slightly abuse notation and also use it to index into $\tilde{\mathbf{m}}$. By defining

$$P_t = \prod_{i \in C_t} \left(\sum_{m_i} p(\tilde{m}_i|m_i) p(m_i|\mathbf{s}) \right), \tag{9}$$

the observed map distribution can be rewritten as $p(\tilde{\mathbf{m}}|\mathbf{s}) = \sum_t p(t) \cdot P_t$. Now using the identity

$$p(m_i|\tilde{\mathbf{m}}, \mathbf{s}) = \left[\sum_t \sum_{\mathbf{m}_{-i}} p(t) p(\tilde{\mathbf{m}}|\mathbf{m}, t) p(\mathbf{m}|\mathbf{s}) \right] / p(\tilde{\mathbf{m}}|\mathbf{s}), \tag{10}$$

where $\mathbf{m}_{-i}$ denotes all entries of $\mathbf{m}$ other than i, $p(m_i|\tilde{\mathbf{m}}, \mathbf{s})$ can be expressed as

$$\left[\sum_t p(t) \cdot P_t \cdot \frac{p(\tilde{m}_i|m_i) p(m_i|\mathbf{s})}{\sum_{m_i} p(\tilde{m}_i|m_i) p(m_i|\mathbf{s})} \right] / \left[\sum_t p(t) \cdot P_t \right]. \tag{11}$$

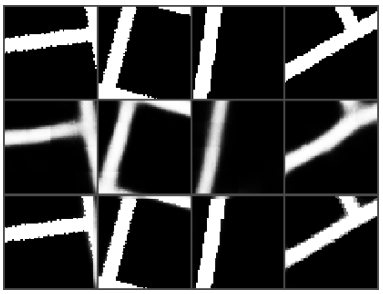


Figure 4. Demonstration of the translational noise model on road detection data. Top row - the target values $\tilde{\mathbf{m}}$ produced from the given map, middle row - model prediction $\hat{\mathbf{m}}$, bottom row - inferred marginal means for the posterior over the uncorrupted labels $p(\mathbf{m}|\tilde{\mathbf{m}}, \mathbf{s})$. Each column corresponds to a training case.

We note that the width of patches to which the noise model is applied ($w_{m'}$) can be different from the width of patches predicted by the neural network (w_p). This allows us to decouple the size of patch for which registration error is assumed to be constant from the size of predicted patch. In this paper we used $w_{m'} = 4w_p$. In this case, we construct the $w_{m'} \times w_{m'}$ patch $\hat{\mathbf{m}}$ out of 16 non-overlapping $w_p \times w_p$ patches predicted by the neural net. We then find the $w_{m'} \times w_{m'}$ patch of posterior marginals $p(m_i|\tilde{\mathbf{m}}, \mathbf{s})$ as described above, break it up into 16 non-overlapping $w_p \times w_p$ subpatches, and backpropagate the derivatives from all the subpatches through the neural network.

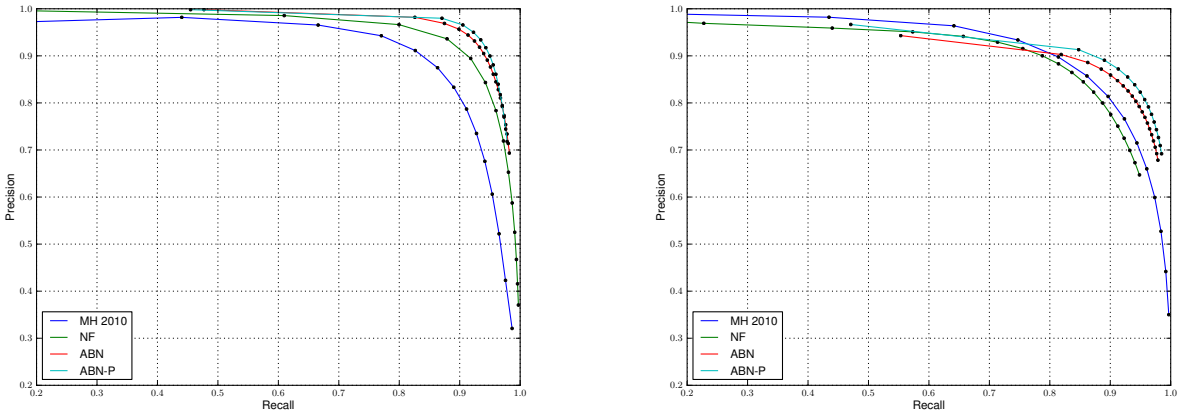
5.2. Model Demonstration

While we delay quantitative results until Section 6, Figure 4 shows some qualitative results that demonstrate the workings of the translational noise model. This example was constructed using a road detector trained on poorly registered road data with the translational noise model. The model uses the target labels produced from the given map (top row) and the model's predictions (middle row) to obtain realigned data (bottom row) through the posterior $p(\mathbf{m}|\tilde{\mathbf{m}}, \mathbf{s})$.

6. Results

6.1. Datasets and Metrics

We evaluate our deep learning framework along with the proposed noise models on the problem of road detection. We use the URBAN1 and URBAN2 datasets that were used in (Mnih & Hinton, 2010) to evaluate a road detection system because these are the largest and arguably most challenging aerial image labeling datasets with published results. The URBAN1 dataset consists of over 500 square kilometers of training data



(a) Precision/recall on the URBAN1 test set.

(b) Precision/recall on the URBAN2 test set.

Figure 5. Comparison of models trained on the URBAN1 dataset.

and 48 square kilometers of test data at a resolution of 1.2m per pixel. This dataset contains both urban and suburban areas of a large city and has relatively few registration problems but does contain omission errors. The URBAN2 dataset consists of a 28 square kilometer subset of a different city than the one covered by URBAN1 and has significant registration problems in addition to containing omission errors. In earlier work, the URBAN2 dataset was only used for testing due to its registration problems. We created an additional training set out of roughly 250 square kilometers of imagery by using the entire city covered by URBAN2 in order to study the effects of registration noise on the training process.

We follow the standard evaluation protocol for road detection problems which involves computing precision/recall plots (Wiedemann et al., 1998). Since the ground truth road locations can suffer from registration problems, it is common practice to use a buffer when computing precision and recall. If a buffer of ρ pixels is used, then a pixel predicted as road is considered to be correctly classified if there is a true road pixel within ρ pixels. Similarly, a true road pixel is considered to be correctly classified for computing recall if there is a predicted road pixel within ρ pixels. We use a buffer of 3 pixels in our evaluations.

6.2. Experimental Setup

To make our results comparable to the best published results on URBAN1 and URBAN2 we used an experimental setup similar to the one used in (Mnih & Hinton, 2010). We trained neural networks to predict 16 by 16 patches of map from 64 by 64 patches of aerial image. In most experiments we used the three hidden layer neural net described in Section 3. The first hidden layer used filter width 12 with stride 4 and 64 filters at each site. The second hidden layer used filter width 4 with stride 2 and 256 filters at each site. The third hidden layer had 4096 hidden units.

The best published results on this data (Mnih & Hinton, 2010) make use of a postprocessing procedure that improves the predictions of a base model by training a new predictor that takes a patch of predictions of the base model as input instead of the aerial image. This procedure tends to fill in short gaps in the predicted road network as well as remove spurious bits of road. The neural network we use for post-processing predicts a 16 by 16 map patch from a 64 by 64 patch of predictions. The network has two locally connected hidden layers with the same connectivity as the first two layers of our three layer network.

We trained all models using mini-batch stochastic gradient descent with a fixed learning rate. We used mini-batches of size 64 and momentum of 0.9. Model parameters were either tuned on the URBAN1 validation set (filter sizes and strides) or set once (number of filters or hidden units) and held fixed for all experiments. All post-processing nets were trained for 8 epochs while base predictor networks were trained for 20 epochs. Our models were trained using consumer GPUs and took about a day to train. We used the CudaMat (Mnih, 2009) and Gnumpy (Tieleman, 2010) Python libraries to implement the algorithms.

6.3. Training on URBAN1

We train three different models on the URBAN1 training set and evaluate them on the URBAN1 and URBAN2 test sets. Figure 5 shows the precision recall curves on the test sets. The model denoted by NF is the three layer network described in the previous section trained using the noise-free model of Equation 2.

To investigate the effectiveness of the ABN loss function we used it to train a model with the same architecture as NF. For this model, shown as ABN, we used the parameter values $\theta_0 = 0.001$ and $\theta_1 = 0.05$. There is a clear improvement in both precision and recall from training with this robust loss function. Neural networks trained with the ABN loss tend to be much more confident in their predictions because they are not penalized as much on the noisy training cases. We also experimented with using the TABN loss function for training on the URBAN1 data but found that this led to nearly identical performance as with the ABN loss. One possible reason for this is that the ABN loss function offers some robustness to translation noise in addition to omission noise. Since the URBAN1 training set has only minor registration problems the added robustness from using the TABN loss does not seem to help.

The best published results for URBAN1 and URBAN2 (Mnih & Hinton, 2010) are included in the plot as MH2010. This model uses a fully connected single hidden layer neural net as a base predictor, followed by a fully connected post processing net. For high recall levels, our ABN model outperforms the MH2010 model by a wide margin on both datasets. When we also train a post-processing neural network (ABN-P) on the outputs of the ABN model we see a further improvement in precision and recall. At low recall levels, MH2010 and NF outperform ABN and ABN-P on the URBAN2 test set, likely because the test set is poorly registered. The noise model allows ABN and ABN-P to make more confident predictions, which lowers precision on parts where the ground truth is misaligned.

6.4. Training on URBAN2

While the URBAN1 dataset contains relatively few registration problems, with most road labels within one or two meters of the true locations, road centerlines in the URBAN2 dataset are often more than 5 meters away from their true locations. We train four models on the URBAN2 training set and evaluate them on the URBAN1 test set. We do not evaluate on the URBAN2 test set because we used the entire city for training due to it being quite small.

Figure 6 shows precision/recall curves on the URBAN1 test set for four models trained on the URBAN2 training set and demonstrates the clear advantage of using robust loss functions on this task. The neural network trained without a noise model, denoted NF, does very poorly, achieving recall of about 0.3 at 0.9 precision. The neural network trained with the ABN model, denoted ABN, achieves double the recall at 0.9 precision

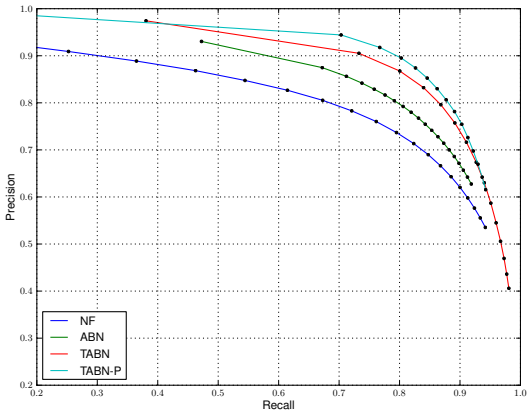


Figure 6. Models trained on the poorly registered URBAN2 dataset evaluated on the URBAN1 test set.

when compared to NF. Unlike for models trained on URBAN1, we get a huge improvement in the precision recall curve from training a network with the TABN model [2]. At 0.9 precision, the TABN model obtains recall of roughly 0.75, which is 2.5 times higher than that of the same network trained without a noise model (NF). Finally, we trained a post-processing network on the outputs of TABN, improving recall to 0.8 for precision 0.9. This last network, denoted TABN-P, was also trained using the translational noise model.

While the performance of these models is worse than for models trained on URBAN1, a gap in performance is to be expected. In addition to its registration problems, the URBAN2 training set is less than half the size of the URBAN1 training set and covers mostly suburban areas which do not resemble the more urban areas in the URBAN1 test set.

7. Related Work

We are not aware of any work related to image labeling that specifically addresses learning from noisy labels. (He & Zemel, 2008) pointed out that the lack of accurately labeled data is a bottleneck in general image labeling and considered the related problem of learning to label images from incomplete observations. The presence of label noise in the aerial image data was addressed in (Kluckner & Bischof, 2009), but it was only used to motivate the use of random forests which can tolerate some mislabeling of the data without having an explicit noise model.

The general problem of learning from noisy labels has been considered in a variety of settings. In particular, the idea of modeling true unobserved labels as latent

[2]The TABN model used the parameter values $\theta_0 = 0.001$, $\theta_1 = 0.05$, and $\theta_t = 0.85$.

variables is widely used. For example, (Pal et al., 2007) also uses an asymmetric Bernoulli noise model, but in the context of information extraction.

The idea of automatically aligning data has been explored in unsupervised learning, where (Frey & Jojic, 1999) incorporated latent variables encoding spatial transformations into a mixture modeling framework, allowing it to simultaneously align and cluster images.

8. Conclusions

Maps provide a very rich source of labels for systems that learn to interpret aerial images and this makes it possible to use systems with a very large number of parameters such as deep neural networks trained on large image patches. However, the performance of these systems is significantly degraded by missing labels and poor registration. We have shown that it is possible to significantly improve performance by using robust loss functions that treat the target labels as noisy observations of true labels. During learning, a version of the EM algorithm is used to infer the true labels and these inferred labels are then used as the targets for training the parameters.

References

Dollar, Piotr, Tu, Zhuowen, and Belongie, Serge. Supervised learning of edges and object boundaries. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1964–1971, 2006.

Frey, Brendan J. and Jojic, Nebojsa. Estimating mixture models of images and inferring spatial transformations using the em algorithm. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 416–422, 1999.

He, Xuming and Zemel, Richard S. Learning hybrid models for image annotation with partially labeled data. In *NIPS*, pp. 625–632, 2008.

Hinton, Geoffrey E., Osindero, Simon, and Teh, Yee-Whye. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18:1527–1554, July 2006.

Kavukcuoglu, Koray, Sermanet, Pierre, Boureau, Y-Lan, Gregor, Karol, Mathieu, Michaël, and LeCun, Yann. Learning convolutional feature hierachies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS 2010)*, 2010.

Kluckner, Stefan and Bischof, Horst. Semantic classification by covariance descriptors within a randomized forest. In *Computer Vision Workshops (ICCV)*, pp. 665–672. IEEE, 2009.

Kluckner, Stefan, Mauthner, Thomas, Roth, Peter M., and Bischof, Horst. Semantic classification in aerial imagery by integrating appearance and height information. In *ACCV*, volume 5995 of *Lecture Notes in Computer Science*, pp. 477–488. Springer, 2009.

Krizhevsky, Alex. Convolutional deep belief networks on cifar-10. Technical report, University of Toronto, 2011.

Lee, Honglak, Grosse, Roger, Ranganath, Rajesh, and Ng, Andrew Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning*, pp. 609–616, 2009.

Mayer, Helmut. Object extraction in photogrammetric computer vision. *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(2):213–222, March 2008.

Mnih, Volodymyr. Cudamat: a CUDA-based matrix class for python. Technical Report UTML TR 2009-004, Department of Computer Science, University of Toronto, November 2009.

Mnih, Volodymyr and Hinton, Geoffrey. Learning to detect roads in high-resolution aerial images. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*, September 2010.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *ICML*, pp. 807–814, 2010.

Pal, Chris, Mann, Gideon, and Minerich, Richard. Putting semantic information extraction on the map. In *Sixth International Workshop on Information Integration on the Web*, 2007.

Tieleman, T. Gnumpy: an easy way to use GPU boards in Python. Technical Report UTML TR 2010-002, University of Toronto, Department of Computer Science, 2010.

Wiedemann, Christian, Heipke, Christian, Mayer, Helmut, and Jamet, Olivier. Empirical evaluation of automatically extracted road axes. In *Empirical Evaluation Techniques in Computer Vision*, pp. 172–187, 1998.