Weakly Supervised Learning for Road Scene Understanding

Donglu Wang
Supervisor: Dr. Jose M. Alvarez

A subthesis submitted in partial fulfillment of the degree of
Master of Computing (Honours) at
The Department of Computer Science
Australian National University

June 2015

Except where otherwise indicated, this thesis is my own original work.

Donglu Wang
7 June 2015

Acknowledgements

We are grateful for the guidance and support by Dr. Jose M. Alvarez thought out our study. The assistance of Yi Liu in manually annotating the data is sincerely appreciated.

Abstract

In our study, we explored the merits and flaws of various methods for basic road scene understanding, and suggested a weakly supervised learning approach that excels both in theory and practice. Tested over different data sets, our approach gave an accuracy of 89.51% in the best case. It has a desirable prediction stage time complexity, which is linear with regard to the size of input data. A software system as well as an open source library for convolutional auto-encoders were developed accordingly.

Keywords: convolutional auto-encoders, convolutional neural networks, unsupervised learning, feature learning, road detection.

Contents

Chapter 1

Introduction

In this chapter, we began with describing the motivations of our study and related challenges. Then, we discussed the strengths and limitations of related works. Thereafter we defined our objectives and listed our major contributions.

1.1 Motivations

Road scene understanding is an important and challenging research topic in computer vision. Accurate detection of free road surfaces from images and videos is essential to many applications including autonomous driving and driver assistance systems.

Difficulties of vision-based road scene understanding arise from many aspects. For instance, pedestrians and vehicles with unknown movements on the road may increase the uncertainty of our judgment. Varying illumination conditions can cause varying degrees of contrast, brightness, reflection and shades. Disadvantageous weather conditions (e.g., rain, snow and fog) will affect camera lens and distort the image and video obtained. Though numerous methods have been studied to overcome these difficulties, there is still room for improvement.

1.2 Related Work

1.2.1 Handcrafted features

Existing road detection methods highly involve handcrafted features. In addition to low level features (e.g., color, texture, coordinates and SIFT [Lowe 1999]), high level features including shapes, lines, intersection points and road geometry are also shown to be beneficial to producing good results [Álvarez et al. 2014].

Although the effectiveness of such methods has been testified across various data sets, they have noticeable drawbacks constraining their applicability in real world scenarios. High-level features often depend on shapes of artificial objects (e.g., lines and intersection points). Although they work well in situations where the roads are well structured, such as freeways and cities, but they are prone to be ineffective in other situations where the outlines of roads are not obvious (e.g., blocked by pedestrian), or are obscured by other factors (e.g., weather and illumination). Algorithms for computing high-level features are not universally applicable, and are often subject to high

computational cost.

Moreover, some handcrafted features are based on prior assumptions of the input images. For example, the location of the road should be in the bottom area of an image, which is likely to be true in most cases but does not possess generality.

1.2.2 Learned features

To address the issue of handcrafted features, algorithms based on convolutional neural networks are introduced to various computer vision tasks [Alvarez et al. 2012] [Alvarez et al. 2012]. Convolutional neural networks are able to discover local, translation invariant and higher order features automatically from training data, and are shown to be the state-of-the-art in many applications.

The training of a convolutional neural network is the process of optimizing a highly non-convex function. Gradient decent methods with random initialization are often inadequate of reaching good local optimums. Although convolutional neural networks are widely applied on computer vision tasks, not enough emphasis has been put on how to choose an effective optimization method as well as an appropriate network architecture.

Convolutional neural networks based methods mainly work with manually annotated labels or labels generated as the output of other programs of similar purposes. Manual label annotation is time consuming and generated labels are often of a considerable amount of noise. Hence to exploit the utility of unlabeled data is another issue of great concern.

1.3 Objectives and Contributions

In this paper, we presented a weakly supervised approach for road scene understanding with the aim of addressing above mentioned challenges. In order to avoid the limitations of handcrafted features, convolutional neural networks are has been used as the primary framework for supervised learning. On top of this, we focused on unsupervised learning methods to improve the performance of our approach and at the same time preserve its generality. More specifically, we investigated unsupervised feature learning methods to leverage the usability of unlabeled data. We also studied unsupervised image segmentation and graph denoising methods for the purpose of improving the outcome and running time of the prediction stage.

The major contributions of our study can be summarized as follows. First we introduced a weakly supervised approach for road scene understating, and built a software system accordingly. Second, we compared the effectiveness of two different unsupervised feature extraction methods, namely principal component analysis (PCA) and convolutional auto-encoders, in several aspects with regard to our study. Third, we experimented the merits and flaws of two unsupervised image segmentation as well as a graph denoising methods of in the prediction stage. Forth, we implemented an open source library for convolutional auto-encoders, which we believed to be more efficient and flexible in comparison of other libraries available on the same platform.

Chapter 2

Preliminaries

In this chapter, we briefly introduced preliminary backgrounds of the techniques we used through out our study. We started by a classic machine learning model called neural networks. Then several variations of neural networks serving different purposes were compared, which are convolutional neural networks, auto-encoders, convolutional auto-encoders, respectively. After that we looked at another unsupervised feature extraction and dimensionality reduction method called principle component analysis, as well as its application in whitening. Then we discussed a type of probabilistic graphical models called Markov random fields, and how it can be used to express the dependency between variables. Finally, we introduced two of the most concerned topics in the field of computer vision, namely image segmentation and 3D reconstruction.

2.1 Neural Networks

A commonly used model for regression and classification problems takes the form of

$$y(x) = f(W\Phi(x)) \tag{2.1}$$

where $f(\cdot)$ is an activation function, and $\Phi(x)$ are the basis functions, which can also be interpreted as a set of features.

While many models suffer from the limitation that the basis function should be fixed before the training data set is observed, neural networks provide a way of learning features from data implicitly by making making the basis functions $\Phi_j(x)$ depend on parameters which are allowed to be adjusted during the training step.

2.1.1 Network functions

A typical neural network model is composed of a series of layers. Each layer constructs M linear combinations of the input variables from its previous layer, and transforms them using an activation function in the form

$$h = \sigma(Wx + b) \tag{2.2}$$

where parameters W are referred to as the weights of input units, b is the bias term, and $\sigma(\cdot)$ is a differentiable nonlinear function.

For binary classification problems the logistic sigmoid function in the form

$$y = \sigma(a) = \frac{1}{1 + \exp(-a)} \tag{2.3}$$

is used for output units, so that we can interpret the output as the conditional probability with respect to a Bernoulli distribution, and the following properties are satisfied.

$$0 < \sigma(a) < 1, \tag{2.4}$$

$$\sigma(a) = 1 - \sigma(-a). \tag{2.5}$$

Similarly for multiclass problems, a softmax activation function is used, which takes the form

$$y_k(x) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}, \tag{2.6}$$

where a_k is the pre-activation value for the k-th output unit.

For instance, the overall network function of a two-layer neural network can be given as

$$y = \sigma_2(W^2 \sigma_1(W^1 x + b^1) + b^2). \tag{2.7}$$

Its corresponding architecture is illustrated in Figure 2.1.1 [Bishop et al. 2006].

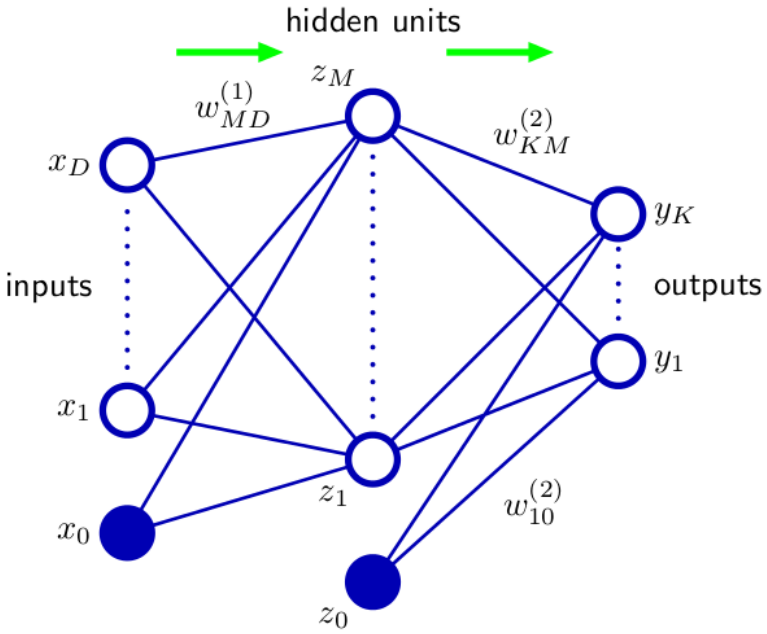

Figure 2.1: Network architecture for the two-layer neural network corresponds to 2.7. Weights associated with node x_0 correspond to term b^1, weights associated with node z_0 correspond to term b^2.

It is shown that multilayer feedforward networks defined this way can approximate any arbitrary function to any desired degree of accuracy, provided sufficiently many

hidden units are available [Hornik et al. 1989].

2.1.2 Network training

Neural networks can be trained by minimizing an error function, which corresponds to the maximum-likelihood estimation (MLE) with regard to certain probabilistic assumptions. In terms of classification problems, the cross-entropy error function of the form

$$E(\theta) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{kn} \ln y(x_n) \tag{2.8}$$

is used, where K is the number of mutually exclusive classes, and t_n is a vector of zeros and ones indicating the target class.

We can apply the stochastic gradient descent algorithm to find a good set of parameters that yields a reasonable amount of errors. The gradient information required with regard to relevant parameters can be evaluated efficiently by error backpropagation [Rumelhart et al. 1985] in the context of neural networks.

2.2 Convolutional Neural Networks

In natural images, pixels that are spatially nearby are highly correlated. The main disadvantage of using neural networks for image classification is it ignores the topology of the input by constructing fully-connected networks. Moreover, fully-connected architectures are associate with a large number of parameters given high-dimensional data, which is prone to overfitting. Therefore convolutional neural networks (CNN) were introduced as a way to extract local, translation invariant features, and limiting the number of parameters substantially at the same time [LeCun and Bengio 1995].

2.2.1 Architectural properties

Convolutional neural networks incorporate three architectural ideas on top of neural networks described previously, which are local receptive fields, shared weights, and pooling and sub-sampling. In following discussions we are going to use the terms "pixel" and "channel" for better illustration in the context of image classification.

2.2.1.1 Local receptive fields

In convolutional neural networks, each pixel of a channel receives inputs from a set of pixels located in a small neighborhood in each channel of previous layer, namely the receptive fields. The output is computed by taking discrete convolution of the receptive fields and kernel matrices (parameters to be tuned), then followed by an activation function.

2.2.1.2 Shared weights

The same set of kernel matrices is shared between receptive fields in different locations of the image. Such approach enables us to extract local features (e.g., oriented edges, end-points, corners) in the scope of the size of kernel matrices. Channels of hidden layers are also referred to as feature maps as each of them represents the outcome obtained using different features (kernel matrices). Separate kernel matrices are used between different input or output channels. For example, a layer with m feature maps and n channels will incorporate a number of $m * n$ kernel matrices.

2.2.1.3 Pooling and sub-sampling

Pooling and sub-sampling refer to the operations that are performed on non-overlapping neighbourhoods. Information of each neighbourhood are usually aggregated in some way to obtain a single value. The benefit of pooling is, the resolution of feature maps are reduced, as well as the sensitivity of the output to shifts and distortions. For instance, the max pooling operation on 2 by 2 non-overlapping neighbourhoods will resulting a feature map of 1/4 of its original size, where each element takes the max value of its corresponding neighbourhood.

2.2.1.4 Network architecture

In practice, a convolutional layer is often followed by a pooling or sub-sampling layer. By stacking such structures repeatedly, we are able to extract effective higher order features. An example architecture of convolutional neural networks are shown in Figure 2.2.1.4 by [LeCun and Bengio 1995].

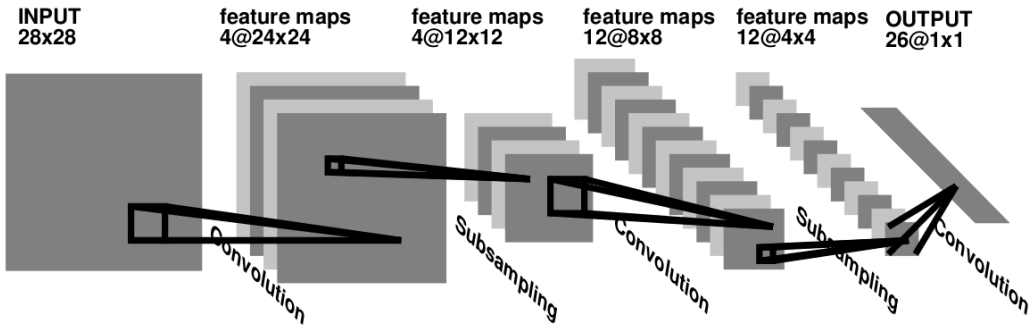

Figure 2.2: The first layer is a convolutional layer with 4 feature maps and 5 by 5 kernels, the second layer is a 2 by 2 sub-sampling layer, the other layers are of similar properties as illustrated.

2.2.2 Network functions

Based on the architecture properties described above, for a N-channel input x the k-th feature map of the next convolutional layer is given by

$$h_k(x) = \sigma(\sum_{n=1}^{N} x_n * W^{nk} + b^k). \tag{2.9}$$

For a pooling or sub-sampling layer, the output depends on the operation specified by that layer. For example, if a p by q max pooling layer is used, the pixel at position (i, j) of a feature map is then given by

$$y_{ij} = \max_{m,n} x_{mn}, \tag{2.10}$$

which subjects to the constraint that

$$m, n \in R_{ij} \tag{2.11}$$

where R_{ij} is a p by q non-overlapping at the (i, j) position. Output activation functions, error functions of convolutional neural networks are similar to neural networks described previously.

2.2.3 Network training

Let E be the error function, for discrete convolution operation $y = x * W$, its partial derivatives with respect to x is given by

$$\frac{\partial E}{\partial x} = \frac{\partial E}{\partial y} \bar{*} w, \tag{2.12}$$

where $\bar{*}$ denotes the opposite way of performing convolutions of $*$ (e.g., if $*$ is the valid convolution, $\hat{*}$ should be the full convolution). Its partial derivatives with respect to W is

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial y} * \tilde{x}, \tag{2.13}$$

where $\tilde{x}$ denotes a matrix given by x with its rows and columns flipped. Having defined the partial derivatives for discrete convolution, we can use error backpropagation and gradient based methods for training convolutional neural networks.

2.3 Auto-encoders

Because the amount of available labels are very limited in many situations, it is important to exploit the usability of unlabeled data. Auto-encoders is an unsupervised learning approach aimed at learning hidden properties of unlabeled data. It is based on neural networks, and can leverage the utility of neural networks in turn.

2.3.1 Network architecture

Auto-encoders are essentially neural networks trained to reproduce its input at the output layer. By doing this it tries to learn a distributed representation that captures the main factors of variation in the data and hence remove input redundancies [Bengio 2009].

Auto-encoders first transform the high-dimensional data into a low-dimensional representation in the form of

$$h(x) = \sigma(Wx + b). \tag{2.14}$$

Then it tries to recover the data from the code via a similar "decoder" network, which is given by

$$\hat{x} = \sigma(W'x + c). \tag{2.15}$$

The two parameter sets W and W' are usually made to share the same weights of the form

$$W' = W^T \tag{2.16}$$

so that the network follows the same manner for encoding and decoding steps.

A typical architecture and its reconstruction result are illustrated in Figure 2.3.1 by [Lemme et al. 2010].

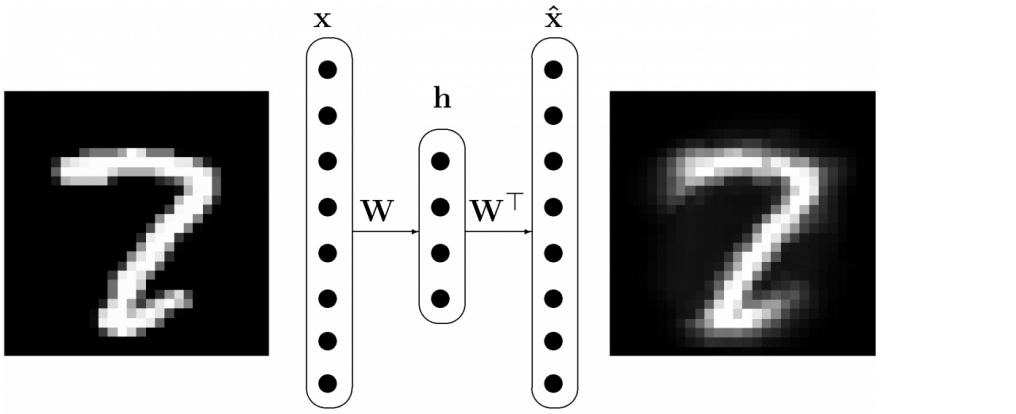

Figure 2.3: A typical auto-encoder network with tied weights W and W^T, input image x is shown in the left, its corresponding reconstructed image $\hat{x}$ is shown in the right.

2.3.2 Network training

Auto-encoders can be trained by minimizing the difference between the original data and its reconstruction. Since image pixels are numerical data, we can use sum of squared errors as our error function, which takes the form

$$E(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(\hat{x_i} - x_i)^2 \tag{2.17}$$

where n is the number of data points.

The required gradients are obtained by using the chain rule to backpropagate error derivatives first through the decoder network and then through the encoder network [Hinton and Salakhutdinov 2006]. In particular, if weights are shared between encoding and decoding steps, W and its transpose appears twice in the network. Thus the gradient with regard of W is a sum of two gradients, which is given by

$$\frac{\partial E}{\partial W} = (\frac{\partial E}{\partial \hat{x}} h)^T + \frac{\partial E}{\partial h} x. \tag{2.18}$$

Knowing the gradient information, the weights of the encoder network can be tuned by performing gradient descent in the same fashion as neural networks.

2.3.3 Denoising Auto-encoders

One problem of this approach is, if the dimension of hidden representation is greater than that of input data, the auto-encoder could potentially just learn the identity function, for the purpose of good reconstruction. The denoising auto-encoder is a variant of auto-encoder that addresses thsi issue by adding stochastical noise to the input data and trying to recover the original data [Bengio 2009].

The error function 2.17 with noiseless input can be applied directly to denoising auto-encoders. It can be proved that it raises a lower bound on the likelihood of several generative models [Vincent et al. 2008].

2.4 Convolutional Auto-encoders

Interesting features can be extracted using denoising auto-encoders. However it assumes that pixels are independent input features, which makes it not suitable enough for processing natural images. Moreover a large number of parameters is required when when applied on high dimensional data. This will increase the non-convexity of error functions, and hence will increase the running time and difficulty in training.

Since convolutional neural networks often excels in image classification problems, convolutional auto-encoders were introduces based on the same architectural ideas. As a result, it is capable of learning local features in an unsupervised way, and scales well to high-dimensional input data since the number of free parameters becomes independent of the input dimensionality [Masci et al. 2011].

2.4.1 Network functions

Similar to general purpose auto-encoders, the aim of convolutional auto-encoders is to obtain a good reconstruction of input data from a set of hidden feature maps. It follows the similar mechanism as general purpose auto-encoders, and applies symmetric convolutional neural network architectures for encoding and decoding.

The network functions of convolutional auto-encoders differ from general purpose auto-encoders as it incorporates discrete convolutions. For a encoding convolutional

layer, the k-th feature map of the latent representation is given by

$$h_k(x) = \sigma(\sum_{n=1}^{N} x_n * W^{nk} + b^k) \tag{2.19}$$

where x is the input data and N is the number of channels of x. In the corresponding deconding stage, the reconstruction is obtained using

$$\hat{x}_n = \sigma(\sum_{k=1}^{K} h_k \bar{*} \tilde{W}^{nk} + c^n) \tag{2.20}$$

where $\tilde{W}^{nk}$ identifies the matrix after flipping both dimensions of W^{nk}, and $\bar{*}$ identifies the opposite way of performing discrete convolution in the encoding step.

In convolutional neural networks, a pooling and sub-sampling layer is often applied to extract translation invariant features. Similarly a max pooling layer that eliminates all non-maximal values in non overlapping neighbourhoods in the form of

$$y_{ij} = \begin{cases} m_{ij}, & \text{if } m_{ij} = x_{ij} \\ 0, & \text{otherwise} \end{cases} \tag{2.21}$$

where m_{ij} is the maximum value of the belonging non-overlapping subregion of x_{ij} defined in 2.10.

By erasing non-maximal values we can acquire a sparse hidden representation which decreases the average number of features contributing to the decoding of each pixel, hence forces features to be more broadly applicable. Consequently, a max pooling layer will reduce the necessity of regularization over weights and stochastic noise over input data [Masci et al. 2011].

2.4.2 Network training

The error function defined in 2.17 is suitable for convolutional auto-encoders as well. Error backpropagation and stochastic gradient decent are also applicable for evaluating gradients and minimizing error functions. Since the weight matrices W affects both the encoding and decoding stage, the corresponding gradient can be given as a summation of two gradient in the from of

$$\frac{\partial E}{\partial W^{nk}} = \widetilde{\frac{\partial E}{\partial \hat{x}_n}} * h_k + \tilde{x_n} * \frac{\partial E}{\partial h_k} \tag{2.22}$$

where $\frac{\partial E}{\partial h_k}$ and $\frac{\partial E}{\partial \hat{x}_n}$ are the partial derivatives of the hidden representation and the reconstruction output.

2.5 Principal Component Analysis

Principal component analysis (PCA) is a commonly used method for dimensionality reduction. It can be defined as the orthogonal projection of the data onto a subspace of lower dimensionality, where the variance of projected data is maximized. Such subspace can represent a set of uncorrelated factors along which the data varies the most, hence it can also be used as a method for unsupervised feature extraction.

2.5.1 Maximum variance

We first begin with the case where the dimensionality of the subspace equals to one. For a given data set x, our goal is to find an orthogonal projection vector u_1 that maximizes the variance v_1 of $u_1^T x$.

The covariance matrix of the data is defined by

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T \tag{2.23}$$

where N is the number of data points, and $\bar{x}$ is the data set mean given by

$$\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_n. \tag{2.24}$$

Therefore the variance of projected data V_1 is given by

$$v_1 = u_1^T S u_1. \tag{2.25}$$

Because we need u_1 to be an orthogonal projection to ensure the bases of the subspace are linearly independent, u_1 needs to satisfy the condition

$$u_1 u_1^T = I \tag{2.26}$$

where I is the identity matrix.

It can be proved that the value of u_1 has to be an eigenvector of the covariance matrix S, and the eigenvector that leads to the maximized variance v_1 is the one with the largest eigenvalue [Bishop et al. 2006]. Eigenvectors and eigenvalues of the covariance matrix can be obtained by diagonalization or singular value decomposition.

2.5.2 Dimensionality reduction

Now we have the required orthogonal projection for one-dimensional space. The orthogonal projection U for k dimensional sub-space can be defined as

$$U = \begin{bmatrix} u_1 & u_2 & ... & u_k \end{bmatrix} \tag{2.27}$$

where u_i eigenvector that corresponds to the i-th largest eigenvalue of S.

Principal components u_1 to u_n then can be interpreted as the top k directions of variation. More precisely, the projected data can be represented by

$$U^T x = \begin{bmatrix} u_1^T x \\ u_2^T x \\ \dots \\ u_k^T x \end{bmatrix}, \tag{2.28}$$

where the rows at the top are likely to take larger values than the rows from the bottom. Thus by selecting the first k components from a m dimensional space, we are approximating the last $m - k$ rows, which have relatively smaller values, by zeros. An example from [Bishop et al. 2006] which projects two-dimensional data onto a one-dimensional subspace using PCA is illustrated in Figure 2.5.2.

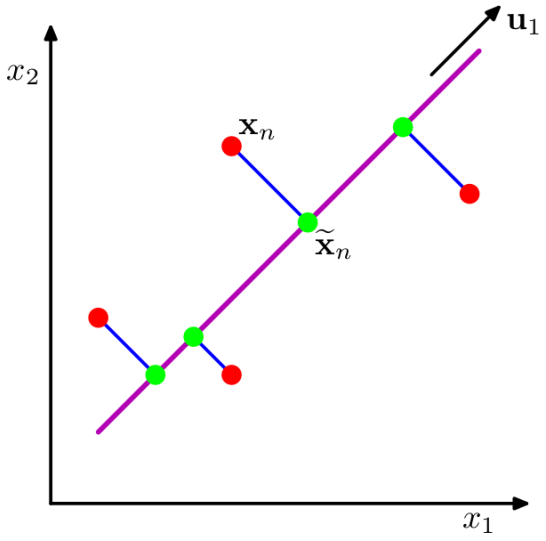

Figure 2.4: Orthogonal projection from two-dimensional space to one-dimensional subspace using PCA, where u_1 is the basis of the subspace, namely the principal component.

2.5.3 PCA approximation

Once the data is projected onto a subspace, we can recover an approximation $\hat{x}$ of the data in the form

$$\hat{x} = \sum_{i=i}^{k} u_i (u_i^T x). \tag{2.29}$$

where k is the dimensionality of the subspace. Thus the value of the last $m - k$ dimensions will be approximated by zero. Since $u_i^T x$ is a scalar, $\hat{x}$ would be a linear combination of the basis vectors.

Recall from 2.3, auto-encoders can generate a hidden representation of data. If the number of hidden units is less than the dimensionality of the input data, this

process can also be interpreted as dimensionality reduction. It is shown that if no nonlinear activation functions are used in auto-encoders, PCA essentially gives the optimal solution of auto-encoders in such cases. In other words, auto-encoders are nonlinear generalizations of PCA [Hinton and Salakhutdinov 2006].

2.5.4 Whitening

Whitening refers to the normalization of data to give it zero mean and same variance, so that different variables become less correlated. One way of achieving unit variance can be done based on PCA by rescaling each dimension of the projected data in the form of

$$w_i = \frac{u_i^T x}{\sqrt{\lambda_i}} \tag{2.30}$$

where λ_i is the corresponding eigenvalue of u_i.

Moreover, if we multiply the whitened data W by an orthogonal matrix, the property of unit variance will not be affected. Therefore, if we multiply W by U and keep all n dimensions of the data, we will get the whitened version of data that is as close as possible to the original data. Such process is called ZCA whitening, which is an important pre-processing step for many algorithms including convolutional neural networks.

2.6 Markov Random Fields

In many cases, a set of features or a set of data points are not independent of each other. Capturing such independencies is essential for building reasonable models. Probabilistic graphical models can be used to make assumptions about conditional independencies among the data set, and Markov random fields (MRF) are a type of probabilistic graphical models that are of particular interests in our study.

2.6.1 Probabilistic graphical models

Probabilities serve as the foundation of statistical machine learning. Numerous models are built based on different probabilistic assumptions. Probabilistic graphical models offer a way of expressing conditional independence properties via a graph, where its underlying mathematical expressions can be derived accordingly.

In a probabilistic graphical model, we used nodes to represents random variables, and edges to express probabilistic relationships between these variables. It constructs a general way of decomposing the joint distribution over all of the random variables into a product of factors, so that each depending only on a subset of the variables.

2.6.2 Markov random fields

Markov random fields are probabilistic graphical models with undirected edges, which is a more natural way of assigning probabilistic relationships than directed graphs in the cases where there is no obvious causal relationships.

If an edge exists between any pair of nodes in a subset of the graph, that set of nodes is called a clique. Every variable from a clique then is dependent of all the other variables in that clique. If no other nodes can be included in a clique, the it is called a maximal clique.

The joint distribution can be written as a production of factors

$$p(x) = \frac{1}{Z} \prod_{C} \psi_C(x_C) \tag{2.31}$$

where x is an assignment of the graph, $\psi_C(x_C)$ is the potential function defined over a maximal clique, and Z is a normalization constant so that

$$\int p(x) \mathrm{d}x = 1. \tag{2.32}$$

2.6.3 Energy function

A convenient way of defining the potential functions is by using energy functions $E(x_C)$ over maximal cliques in the following form, so that the joint probability follows the Boltzmann distribution.

$$\psi_C(x_C) = \exp\{-E(x_C)\} \tag{2.33}$$

Consequently, an assignment of x_C has a high probability if the energy of that clique is low. Moreover the total energy of the joint distribution equals to the summation of the energies of each maximal clique.

2.6.4 Image denoising

Markov random fields can be used for image denoising. Given a noisy image, our goal is to recover the original image based on the probabilistic relationships assumed by Markov random fields.

More specifically, for a m by n noisy image, we first use $m \times n$ nodes to represent all the pixels. Since we already knew the value of each pixels, such set of nodes are called observed nodes. Then we use another set of $m \times n$ latent nodes to represent the noiseless image we want to recover, whose values are unknown. An edge is assigned between each pair of latent and observed nodes, denoting some extent of consistency between the noisy and original image. Edges are also constructed between neighbouring latent nodes, which indicates their values are correlated. An illustration of such Markov random fields given by [Bishop et al. 2006] is shown in Figure 2.6.4.

To ensure the above probabilistic relationships, we need to construct energy functions that returns a smaller value if nodes in a clique agree with each other and returns a greater value if not. Thus the latent nodes with the highest probability will prefer similar values in a neighbourhood and at the same time being consistent with the observed nodes to some extent as well. Therefore such latent nodes are likely to a good estimation of the original noiseless image.

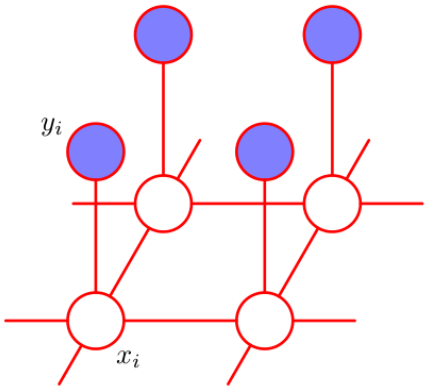

Figure 2.5: An example Markov random fields for image denoising, y_i is a an observed node and x_i is a latent node.

In order to reach the state of the lowest energy, we can use the max-product algorithm on factor graphs. However, the construction of factor graphs is difficult if the structure of MRF is allowed to be mutable. For discrete variables, a local search algorithm called iterated conditional modes can be applied on an arbitrary structure is more suitable to our needs [Kittler and Föglein 1984].

We can further extend this app latent nodes to any given graph with observed assignments as long as they satisfy the properties of Markov random fields.

2.7 Image Segmentation

Image segmentation is the task of splitting images into groups of pixels that share similar attributes (e.g., belong to the same semantic object). Since an image can be segmented in different manners, hundreds of algorithms are studied for various requirements [Szeliski 2010]. In our study, we are particularly interested in segmenting images into super-pixels, so that the pixels within the same group are visually similar but not semantically meaningful.

2.7.1 Graph-based segmentation

Images can be represented by a graph where each pixel corresponds to a node and edges are established between adjacent pixels. While many algorithms use a fixed criterion to decide similarities between pixels or regions, an algorithm that employs a relative measure of evidence for a boundary between two regions is shown to be of great practical importance [Felzenszwalb and Huttenlocher 2004].

If we use graph G to represent an image, then a region R of the image is essentially a connected component of graph G. Let $MST(R)$ be all the edges in a minimum spanning tree of R. The algorithm uses $Int(R)$ to evaluate the internal difference for

any region R, which is given by

$$Int(R) = \max_{e \in MST(R)} w(e) \tag{2.34}$$

where $w(e)$ is the weight of the edge. In terms of image segmentation, it should be the measure of the dissimilarity between the two pixels connected by that edge (e.g., the difference in intensity, color).

The difference between two regions R_1 and R_2 is measured by the weight of the shortest edge connecting two regions, in the form of

$$Dif(R_1, R_2) = \min_{v_1 \in R_1, v_2 \in R_2} w(v_1, v_2). \tag{2.35}$$

The algorithm will merge to disconnected regions if $Dif(R_1, R_2)$ is greater than the minimum value of $Int(R_1)$ and $Int(R_2)$. In other to adjust the evidence of a boundary as we need, a threshold function τ is often added to the minimum values. So that the minimum internal difference of R_1 and R_2 becomes

$$MInt(R_1, R_2) = \min((Int(R_1) + \tau(R_1)), (Int(R_2) + \tau(R_2))). \tag{2.36}$$

Such operations can be performed efficiently using disjoint sets, the running time hence is linear with regard to the number of pixels in practice. Super-pixels generated by the graph-based segmentation can be guaranteed to be neither to coarse or to fine [Felzenszwalb and Huttenlocher 2004]. It has many pragmatic applications, including the 3D reconstruction method we are going to discuss in Section 2.8. An example result using graph-based segmentation on a road scene image is shown in Figure 2.6.

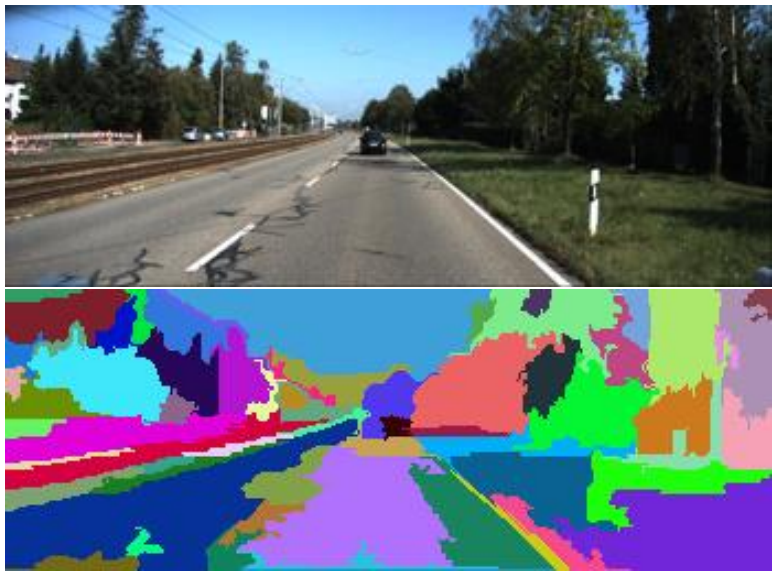

Figure 2.6: Super-pixels (bottom) of a road scene image (top) generated by graph based image segmentation.

2.7.2 Simple linear iterative clustering

One important limitation of the graph-based segmentation algorithm is it does not engage spatial regularization properties, since the weights of edges in a graph only depend on the colour and intensity attributes of connected pixels. Many applications prefer super-pixels that are regular and compact in shapes as they can be considered as a good abstraction of images.

In the simple linear iterative clustering (SLIC) algorithm, pixels are consider as points in a five-dimensional space, which is a combination of a three-dimensional color space and a two-dimensional spatial plane. By utilizing the spatial information of pixels, it offers an adaptable degree of regularization in shapes of super-pixels.

In the SLIC algorithm we often represent images using the Lab colour space, therefore the distance between two pixels in the five-dimensional space is given by

$$d(p_1, p_2) = \sqrt{(\frac{d_c}{n_c})^2, (\frac{d_s}{n_s})^2} \tag{2.37}$$

where n_c and n_s are spatial regularization terms, d_c is the Euclidean distance in colour space in the form of

$$d_c(p_1, p_2) = \sqrt{(l_1 - l_2)^2, (a_1 - a_2)^2, (b_1 - b_2)^2} \tag{2.38}$$

and d_s is the Euclidean distance in spatial plan in the form of

$$d_s(p_1, p_2) = \sqrt{(x_1 - x_2)^2, (y_1 - y_2)^2}. \tag{2.39}$$

The SLIC algorithm then performs a clustering algorithm based on iterative improvement described in [Achanta et al. 2012]. As suggested by its name, the running time of this algorithm is shown to be linear in practice.

2.8 3D Reconstruction

The task of reconstructing 3D models from a set of images is call 3D Reconstruction. It has substantive practical applicability, and many algorithms as been devised for this purpose. However most of the algorithms require two or more images taken from the same scenario, which is not suitable for our purpose.

A different approach to infer 3D structure was introduced as part of the "Photo Pop-up" application by [Hoiem et al. 2005]. Instead of using widely applied multiple view geometry methods, this approach engages statistical machine learning methods to train a classifier. It requires only a single image, can assign each part of the image into one of three basic geometric categories, namely "ground", "vertical" and "sky".

In the training step, a image will be segmented into super-pixels using the graph-based segmentation described in 2.7.1. Then for one super-pixel, a set of features (including colour, texture, location, shapes, lines, intersections etc.) are extracted. A statistical model for classification is trained on a set of manually labeled images.

In the prediction step, an image will be first segmented into super-pixels. Then a label for each super-pixels can be obtained via the trained classifier. Finally they will be grouped into plausible regions in a way such that super-pixels within the same group are of high probability of belonging to the same geometric category. A elaborated description and sample outputs (Figure 2.7) of the algorithm and its theoretical background is provided in [Hoiem et al. 2005].

Figure 2.7: Sample input image (top left), segmentation result (top right), labeling result (bottom left) and 3D view (bottom right) of the "Photo Pop-up" application.

Chapter 3

Methods

In this chapter, we first gave an overview of the methods involved in this study. After that we explained how we collected our data and created labels. Then we introduced the principal model we used for image classification. On top of that, we investigated unsupervised learning methods to improve effectiveness and efficiency in the pre-training and prediction stage separately. Finally, the implementation details of our system were provided.

3.1 Overview

In our study, we focused on a simple task of road scene understanding, which can be defined as to assign every part of a given image a label from one of the three classes, namely road, objects and sky. A widely used method is, for a single pixel, a patch consisting of its adjacent pixels will be used as the input for predicting the class of that pixel (Figure 3.1), then the outputs from different parts of the image are organized together in a certain way to acquire the prediction result of the whole image. By adopting such approach we formalized our task mainly into a typical image classification problem.

Therefore, we took Convolutional Neural Networks as a starting point, and applied various unsupervised learning approaches for the purpose of improving performance without losing generalizability. More specifically, we investigated unsupervised feature extraction methods including Principal Components Analysis and Convolutional Auto-Encoders. To further improve our performance in the prediction stage, several image segmentation algorithms were compared for super-pixel labeling. Finally, a method based on Markov Random Fields were experimented for reducing the noise of prediction output on a super-pixel level.

3.2 Data

3.2.1 Data Source

The KITTI data set [Geiger et al. 2013] is a comprehensive data set specially made for computer vision tasks about autonomous driving. We used the data provided in the

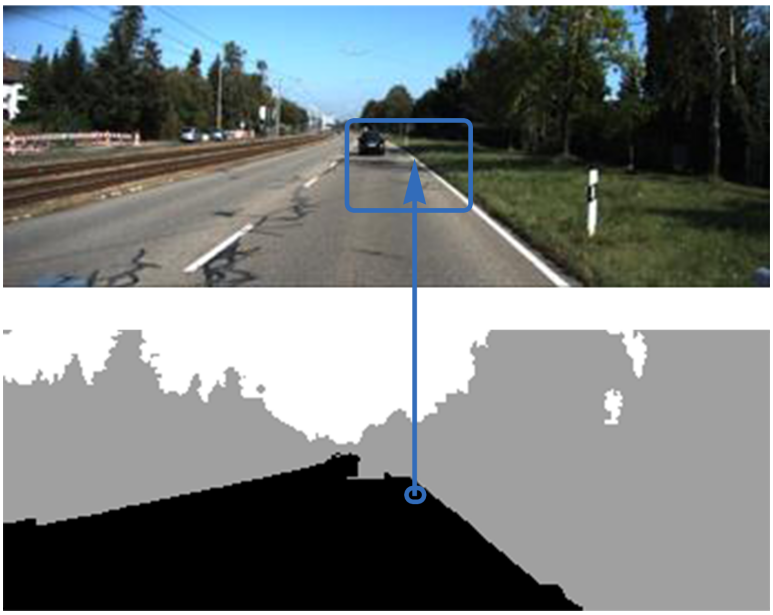

Figure 3.1: Only the area within the blue box in the original image (top) is extracted to be the input of predicting the label of the centering pixel (bottom).

KITTI "road" category, which consists of images taken by high-resolution color and grayscale video cameras equipped on a standard station wagon driving on main roads.

Because consecutive images taken within a short time slot are visually very similar, they are considered to be redundant in our study since we only take patches from a single image individually at a time. Therefore we selected only one out of five consecutive images from then "road" category, and partitioned the selected images into training and test sets.

3.2.2 Label generation

Apart from road scene images, we need their corresponding labels for training the classifier. Labels that suit our needs are not largely available. Since manually annotating labels is labour intensive and time consuming, we mostly used 3D reconstruction methods to generate labels for our data.

Multiple view geometry based 3D reconstruction methods require more than one image from different perspectives of the same scenario, which is not applicable in our case. The "photoPopup" application is based on a trained classifier using handcrafted features, and is capable of roughly assigning each part of a single given images a label out of three categories namely, ground, sky, and vertical [Hoiem et al. 2005].

However this methods is inaccurate itself, and the "ground" category is not exactly what we defined as "road". For instance grasslands, railways and side walks are also expected to belong to the the "ground" category in the "photoPopup" application.

Therefore we manually discarded some labels of intolerable quality, and retain only about 50% of the generated labels. After that, we had 264 labeled images in the training set, and 205 labeled images in the test set.

3.2.3 Manual labels

In order to compare with and quantify the quality of generated labels. A subset of the images that contains 60 most visually different images (40 from the training set, 20 from the test set) were manually annotated by our collaborator Yi Liu. By comparing 60 manual labeled images against generated labels, we know the noise ratio of generated labels is approximately 12.74%. A comparison of generated and manually annotated labels is shown in Figure 3.3.

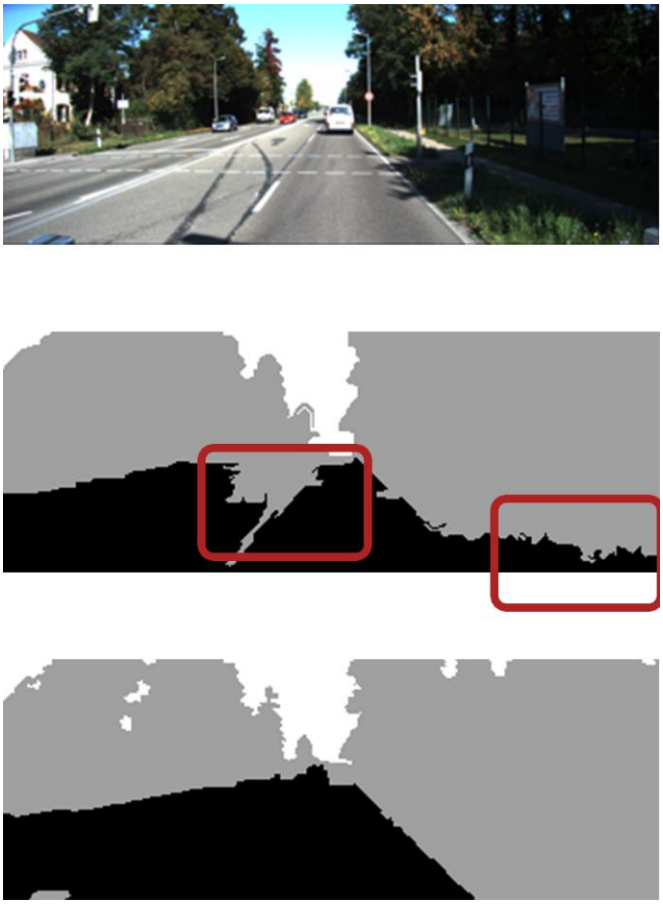

Figure 3.2: Original image (top), generated labels (middle), manual labels (bottom). The areas within the red boxes of generated labels are considered as noise.

3.3 Image Classification

Available machine learning methods are shown to have their own traits on different classification problems. For natural images and videos, adjacent pixels are believed to be highly correlated, which is associated with high redundancy if every pixel is treated as an independent dimension. Thus it is essential to take apply appropriate feature extraction methods in order to acquire good results and reduce algorithmic complexity.

For this reason many feature extraction methods are deliberately designed for image data, especially for images taken from man made objects. As discussed in Section 1.2.1, these handcrafted feature extraction methods can excel in specific problems, but do not possess good generalizability.

From Section 2.1 we know that neural networks are able to learn features implicitly given a redundant input. But since it employs a fully connected architecture, a large number of parameters may be involved, which is prone to overfitting and will increase the difficulty in training.

As described in Section 2.2, convolutional neural networks can capture the correlation of adjacent values by taking discrete convolutions of receptive fields and kernel matrices, and is proven to be an effective approach for many image classification problems. Since weights are shared between receptive fields in different locations, it is able to extract local features and the number of parameters can be reduced substantially. By construct a deep architecture with pooling layers it is able to learn transition invariant and higher order features. Therefore convolutional neural networks are used as the main framework for learning and predicting the label of a given image patch.

3.4 Pre-training

3.4.1 Whitening

From Section 2.5 we know that ZCA whitening can normalize the data to have zero mean and the same variance, while making the features less correlated. Such desirable properties has made it an important pre-processing step for many learning algorithms including convolutional neural networks. In particular, we did not perform dimensionality reduction and rotated the data to be as close as possible to the original data. By doing so we were able to preserve the intrinsic correlations between adjacent pixels, so that the pre-processed data would still be suitable as the input for convolutional neural networks.

3.4.2 Unsupervised Feature Learning

Since generated or manually annotated labels are insufficient either in quality or quantity, to leverage the usability of unlabeled data is of great importance. Although PCA and auto-encoders are shown to be successful for general purpose feature extraction, they fail to capture the existing topology of the data by considering each input dimension independently. Convolutional auto-encoders offer a more sophisticated method for learning meaningful features from images in an unsupervised way.

As was discussed in Section 2.4, convolutional auto-encoders can learn local and transition invariant features using similar architectural ideas as convolutional neural networks. But unlike convolutional neural networks, by using the input data as target output, convolutional auto-encoders can learn intrinsic features of the data itself instead of for some specific purpose. Therefore, features extracted by convolutional auto-encoders are believed to be more robust especially when training labels are noisy or insufficient. Moreover, higher order features can be obtained by training deep architectures in a layer-wise fashion [Masci et al. 2011].

The reconstruction results also provide insights into the issue of how suitable the architecture is for our data. In general, as the complexity of the network architecture increases, the reconstruction error tends to go down. It normally will slowly approach to an asymptotic lower bound after the architectural complexity exceeds some certain point. Thus, the architectural complexity around that point is believed to be an suitable architecture for the data, which is neither too simple to cover important features nor too complex to manipulate.

Moreover, because the first layer kernel matrices are directly applied on input images, by visualizing the kernel matrices per feature map, we are able to interpret the conceptual meaning of some features, such as points, edges and colours. By visualizing the first layer kernels, we can gain a better understanding of indeed how convolutional neural networks and convolutional auto-encoders work.

Another significant benefit of convolutional auto-encoders is, the train results can be used to initialize a convolutional neural network of the same architecture by directly copying corresponding parameters. Optimizing highly non-convex objective functions arises in virtually all deep learning problems, such initialization can avoid numerous distinct local minima and yield a better result in many cases [Masci et al. 2011]. Furthermore, it also helps to reduce the number of iterations required in the training process than random initialization.

3.5 Prediction

3.5.1 Super-pixel labeling

So far we have investigated methods for predicting the label of a single pixel. The simplest approach for inferring all the labels of a given image would be to predict its labels pixel by pixel. Since an image often contains millions of pixels, such approach will be extremely inefficient. For instance, if a 32 by 32 patch is used for predicting one pixel, the amount of data we need to process for a 1024 by 1024 color image will then become $32 \times 32 \times 1024 \times 1024 \times 3$ pixels, which could be greater than one gigabyte.

A more efficient approach for inferring a whole image is by making use of super-pixels. More specifically, we first segment an image into super-pixels such that pixels within the same super-pixel are visually similar. After that, one representative are selected for each super-pixel. Then the inference result of the representative is shared among the pixels from the entire region covered by the super-pixel. The processing of arranging visually similar pixels into super-pixels is essentially another application of

unsupervised learning called clustering, which is used to discover the hidden structure of the data.

As we discussed in Section 2.7, the super-pixels generated using the graph-based image segmentation algorithm can be of arbitrary shapes (Figure 2.6). Such irregular shaped super-pixels raise the difficulty of selecting sensible representatives. For instance, for a ring-shaped super-pixel, the mean or median of the coordinates of the points in that region in fact does not belong to the ring itself. Although there are techniques to select a medoid point, which is the member of a data set whose average dissimilarity to all the other points is minimal. Patches taken from the surrounding areas of such medoid points would still not be representative enough for such ring-shaped super-pixels.

The SLIC algorithm can generate super-pixels with an adjustable degree of spatial regularization. If a proper regularization parameter is chosen, we can obtain super-pixels whose shapes are regular enough to select good representatives from and are at the same time flexible enough to reflect the outline of an image (Figure **??**). For such regular shaped super-pixels, we can easily use their centroids as representatives. Moreover, the number of super-pixels generated by the SLIC algorithm can also be controlled approximately by another input parameter. This allows us to obtain a roughly fixed number of super-pixels regardless of the size of input image. In terms of time complexity, these two algorithms are both linear with regard to the size of the image.

Figure 3.3: Segmentation result of SLIC algorithm. Parameters are adjusted so that the shape of super-pixels are neither too fine or too coarse.

The benefits of this approach are, first, it can reduce the running time required for the inference steps substantially, if a fixed number of super-pixels is used, we can maintain the time complexity of preforming convolutional neural networks at a constant level. Second, since only centroids of super-pixels are used for inference, we managed to avoid the boundary points between super-pixels, where misclassifications are likely to happen. Third, since the same label in shared within a super-pixel that consists of visually similar images, the strong collation of adjacent pixels are captured in some sense.

3.5.2 Super-pixel denoising

It can be expected from similar cases, our inference result can not reach a perfect accuracy. If it satisfies the case that a relatively small amount of misclassifications exist in our output, those misclassified super-pixels can be regarded as noise if a strong correlation of adjacent super-pixels is assumed. Then we can apply noise reduction methods on a super-pixel level in order to achieve a higher accuracy.

We focused on a graph denoising method using probabilistic graphical models. As described in Section 2.6, the MRF based graph denoising method tries to find a most likely state of the original graph according to a probabilistic distribution specified by some energy function (Equation 2.33). Thus, the state of the highest probability will have the lowest energy.

We devised the energy function as follows. For the maximal cliques consisting of only an observed node and its connected latent node, the energy function is given by

$$E(C_{ol}) = E(x_o, x_l) = \begin{cases} 0, & \text{if } x_o = x_l \\ 1, & \text{otherwise} \end{cases} \tag{3.1}$$

where x_o is the observed node and x_l is the latent node. Therefore the clique will have zero energy if the observed node agree with the latent node, and vice versa. For the maximal cliques consisting of a set of mutually connected latent nodes, the energy function is given by

$$E(C_l) = \frac{1}{Z} \sum_{x_1 \in C, x_2 \in C} e(x_1, x_2) \tag{3.2}$$

where Z is a normalization factor depends on the size of the clique, and $e(x_1, x_2)$ is the energy between any pair of nodes in the form of

$$e(x_1, x_2) = \begin{cases} 0, & \text{if } x_1 = x_2 \\ 1, & \text{if } x_1 \neq x_2 \\ & \text{and one of them is of the “objects” state} \\ 2, & \text{if } x_1 \neq x_2 \\ & \text{and they are of the “ground” and “sky” state respectively} \end{cases} . \tag{3.3}$$

According to the above energy function, if the “ground” and “sky” states exist in the same maximal clique, the clique will have a higher energy since these two states are not likely to appear closely together according to our prior knowledge. Apparently this is an empirical assignment, which should be tuned by more sophisticated methods. But this is beyond the scope of our study.

The total energy of the graph is given as

$$E(G) = \alpha \sum E(C_{ol}) + \beta \sum E(C_l) \tag{3.4}$$

where α and β are the parameters for controlling the extent of how much we want the latent node to be consistent with the observed nodes. Given the energy functions, we

can use the iterated conditional modes algorithm to construct a less noisy graph.

3.6 Implementation

A software system was developed to explore the above described methods. We used Matlab as the main platform and integrated some C++ programs. The major components of the system include, label generation, convolutional neural networks, data pre-processing, convolutional auto-encoders, super-pixels and MRF denoising. The running time of the inference stage of our implementation is linear with regard to the size of the image, and only grows by a constant factor as the complexity of network architecture increases.

3.6.1 Label generation

For generating labels automatically, the "photoPopup" application (`http://dhoiem.cs.illinois.edu/projects/popup/`) was integrated to our system. Given an image, this application should be performed on the result of graph based image segmentation. An efficient implementation of graph based image segmentation using the disjoint sets data structure is available at `http://cs.brown.edu/~pff/segment/`.

3.6.2 Classification

Our implementation of convolutional neural networks was based on the DeepLearnToolbox library (`https://github.com/rasmusbergpalm/DeepLearnToolbox`) for Matlab, which applied stochastic gradient descent for network training [Palm 2012]. By the time our study was conducted, it only supported single channel input images. We did proper changes to make it compatible with multiple channel input data. We implemented ZCA whitening followed the guidelines on Stanford's UFLDL (`http://ufldl.stanford.edu/wiki/index.php/Whitening`).

3.6.3 Convolutional auto-encoders library

In terms of convolutional auto-encoders, to the best of our knowledge, there is no suitable libraries for our needs. We developed an open source library for convolutional auto-encoders named cae_toolbox (`https://github.com/Dontloo/cae_toolbox`) according to [Masci et al. 2011]. The toolbox contained functions for training convolutional auto-encoders, visualizing training results and first layer kernels, and initializing convolutional neural networks which is compatible with the DeepLearnToolbox library.

The principal difficulty arose from evaluating the gradient information by error backpropagation. In addition, since discontinuous functions such as max pooling were engaged, it was crucial to ensure the correctness of the partial derivatives given in Equation 2.22.

To this end, we also evaluated the partial derivatives numerically. We first chose ϵ to be a small value (e.g., 10^{-6}). For each parameter θ, we computed the value of

$E(\theta + \epsilon)$ and $E(\theta - \epsilon)$ while other parameters stayed unchanged. Then the partial derivative with regard to θ can be estimated in the form of

$$\frac{\partial E}{\partial \theta} = \frac{E(\theta + \epsilon) - E(\theta - \epsilon)}{2\epsilon}. \tag{3.5}$$

By comparing with partial derivatives evaluated numerically, we have testified the correctness of our error backpropagation implementation.

3.6.4 Prediction

Two available implementations of the SLIC algorithm were compared in our study. The SLIC implementation for the VLFeat library (`http://www.vlfeat.org/overview/slic.html`) had the disadvantage of producing disconnected super-pixels, while another implementation from Image and Visual Representation Lab (IVRL) `http://ivrl.epfl.ch/research/superpixels` could produce more desirable results, hence was integrated into our system for the prediction stage. The optimization of the MRF energy function was implemented according to [Kittler and Föglein 1984].

Chapter 4

Results

In this chapter, we provided our experiment results using different methods and data sets. We started by a brief analysis of our data set. Then we discussed and compared the results of various methods, including convolutional neural networks, ZCA whitening, convolutional auto-encoders, super-pixel labeling and super-pixel denoising, respectively. Thereafter, we evaluated their performance on manually annotated data.

4.1 Data Set Attributes

Our training set contained 264 road scene images and generated labels, which had been rescaled to the resolution of 128 by 348 pixels. We chose a patch size of 32 by 32, which we believed to be an appropriate size that can reflect the local property around a pixel and at same time contain sufficient information for inferring which category it belongs to.

The distribution of pixels with respect to each category was unbalanced. Since we used the softmax error function, different cases of misclassification were equally weighted. Therefore an unbalanced data set was going to bias the classification result to be in favor of the category of the largest population.

4.2 Convolutional Neural Networks

4.2.1 Training and test set

In order to build a balanced training set, for each image, we randomly selected 25 pixels from each category, and extracted their surrounding patches (of size 32 by 32) accordingly. A number of 75 (25×3) patches selected from a 128 by 348 image will roughly cover every part of the image. In total our training set consisted of $3 \times 25 \times 264 = 19,800$ patches evenly taken from each category, which will be referred to as *gen_train* set in following discussions.

We have 205 separate images available for testing, the *patch_test* set was extracted by randomly selecting 100 patches from each image in the test set. Thus *patch_test* set was an unbalanced set consisted of $100 \times 205 = 20,500$ patches.

4.2.2 Outcome

We trained a convolutional neural network directly on the *gen_train* set using stochastic gradient decent for 100 epochs, and then tested it against the *patch_test* set. The training error was 0.1535, the test error was 0.1974. This result was used as a baseline for evaluating the effectiveness of other methods. An example prediction outcome is shown in Figure 4.1.

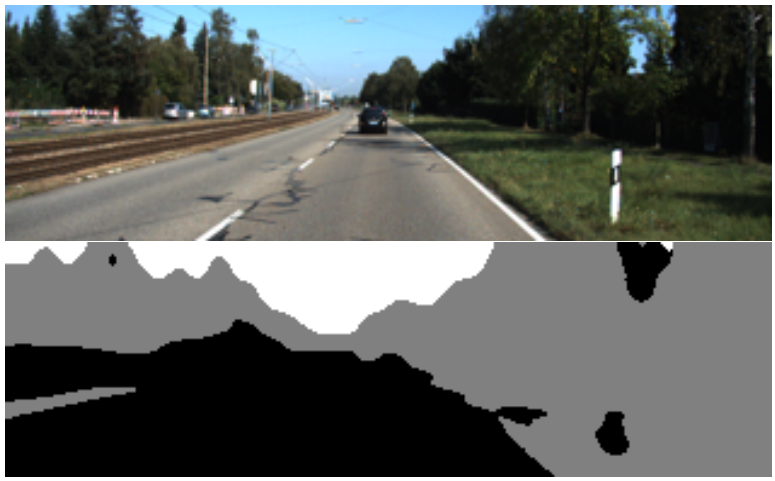

Figure 4.1: The prediction result (bottom) using convolutional neutral network train on *gen_train* set of a input image (top).

The convolutional neural network we used was composed of 4 layers, the first layer was a convolutional layer using 5 by 5 kernels and 15 feature maps, the second layer was a max pooling layer using 2 by 2 pools, the third layer was similar to the first using 24 feature maps instead, the four layer was the same as the second. How we chose thees architectural parameters will be discussed in Section 4.4.

4.3 ZCA Whitening

4.3.1 Pre-training data set

When applying ZCA whitening in the pre-processing step, we need to do the same pre-processing operation for the test data because our classifier would then be trained on whitened images. From Equations 2.29 and 2.30 we can infer that the bases of the projective space U should be independent of how the data were drawn from the ensemble of all the road scene images. Thus for a better generalizability on unknown data, it would be more reasonable if we apply a more general data set to estimate the projective space U.

Therefore out data for pre-processing were drawn from a larger set of 942 images, which is a combination of our training set and other road scene images from the KITTI data set. More specifically, we took 32 patches from each image so that in total $30,144$

patches was contained. This set will be denoted as the *pre_train* set in the following sections.

4.3.2 Outcome

We evaluated the parameters for ZCA whitening on the *pre_train* set, and trained the convolutional neural networks in the same manner as described in Section 4.2. The training error was 0.1245, the test error was 0.1786, which shows an improvement of 1.9% than the baseline. An example outcome using ZCA whitening is shown in Figure 4.1.

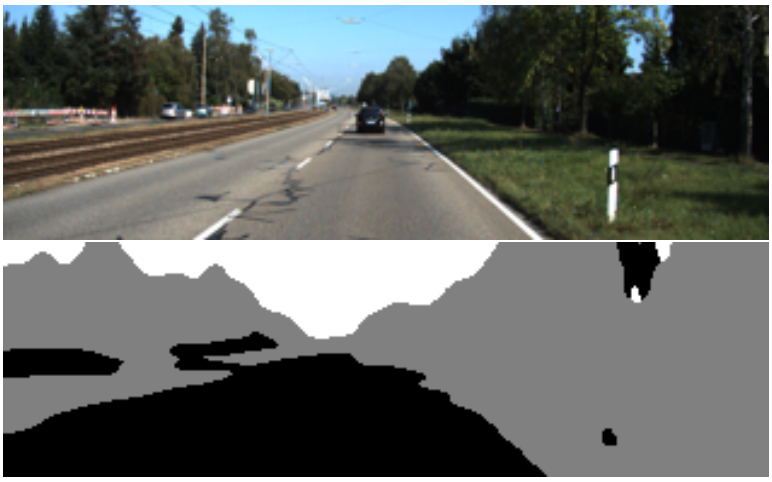

Figure 4.2: The prediction result (bottom) using ZCA whitening of the same input image (top). It shows some improvement compared to Figure 4.1.

4.4 Convolutional Auto-encoders

4.4.1 Unsupervised feature learning

Convolutional auto-encoders is an unsupervised learning approach. For similar reasons as ZCA whitening, we experimented the convolutional auto-encoders over the *pre_train* set for better generalizability. Following the same fashion as described in Section 3.4.2, the fist layer reconstruction error tended to converge at 15 feature maps with 5 by 5 kernel size, the second layer reconstruction error tended to converge at 24 feature maps with 5 by 5 kernel size. The reconstruction error (2.17) of 32 by 32 RGB patches converged at around 13.3 in our experiment. Visualization of selected first layer kernels are shown in Figure 4.3.

An interesting observation was if we apply a more complected architecture, such as increasing the number of feature maps, we are likely to have more lightweight features with no obvious patterns or heavyweight features similar to existing ones. Which again confirmed our belief that the network architecture we adopted was reasonable.

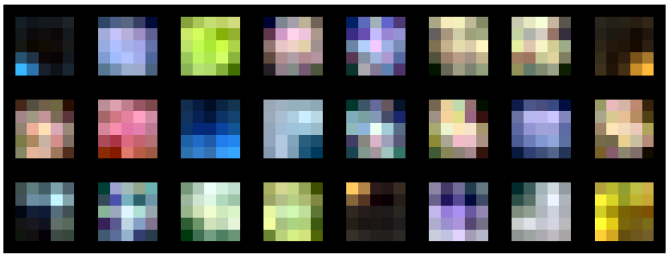

Figure 4.3: A number of 24 first layer kernels trained on the *pre_train set* set. As we can see some of the feature do not have an obvious pattern, in fact they are associated with much lower weights.

	patch_test	*super_test*
Raw	0.1974	0.1918
ZCA	0.1786	0.1762
CAE	0.1726	0.1701

Table 4.1: Test error of different learning and testing methods. Rows indicate learning methods, Raw denotes using raw data as input, ZCA denotes using ZCA whitening to pre-process, CAE stands for using a trained convolutional auto-encoder to initialize. Columns indicate test methods and test sets explained previously.

4.4.2 Outcome

After we applied the parameters of a trained convolutional auto-encoder directly as the initialization of a corresponding convolutional neural network, trained and tested in the same manner in Section 4.2, we obtained a training error of 0.1237, and a test error of 0.1726, which shows an improvement of 0.6% than using ZCA whitening and 2.5% than the baseline. Moreover, the number of epochs required for training error to converge had decreased as the convolutional neural network was well initialized .

4.5 Super-pixel Labeling

For super-pixel labeling (Section 3.5.1), we segmented each image into approximately 300 super-pixels, which turned out to be a relatively small number but sufficient to capture the outline of the image. The 205 images in the test set are used directly, and will be referred to as *super_test* set. As was shown in Table 4.1, when super-pixel labeling was applied, the outcomes of the above three learning methods were improved by different percentages ranging from 0.24% to 0.56%. A sample outcome of super-pixel labeling is shown in Figure 4.4.

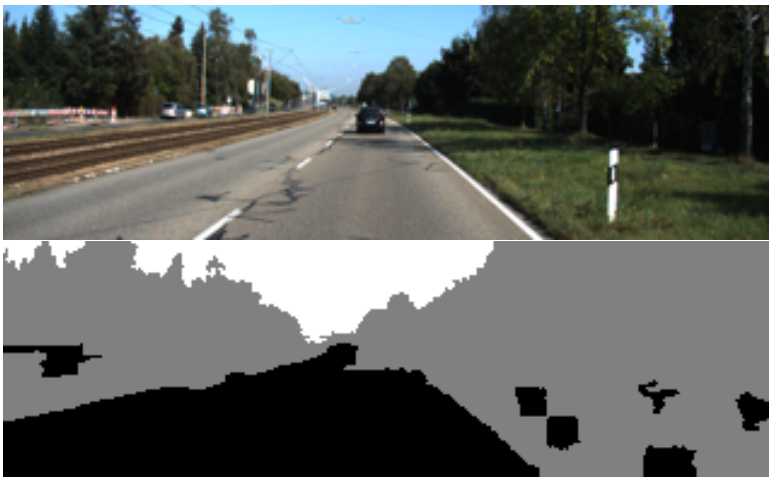

Figure 4.4: The prediction result (bottom) using convolutional auto-encoders and super-pixel denoising. It again shows some improvement compared to Figure 4.1 and 4.2.

4.6 Super-pixel Denoising

As we can see from Figure 4.4, the super-pixel labeling output could be noisy. After MRF denoising, the labels became much smoother (Figure 4.5). However, it also changed some correctly classified but isolated super-pixels in to other categories, in order to achieve a smooth picture. Moreover, because its effectiveness depends on how the misclassifications were distributed, it in fact could decrease the accuracy in many other cases.

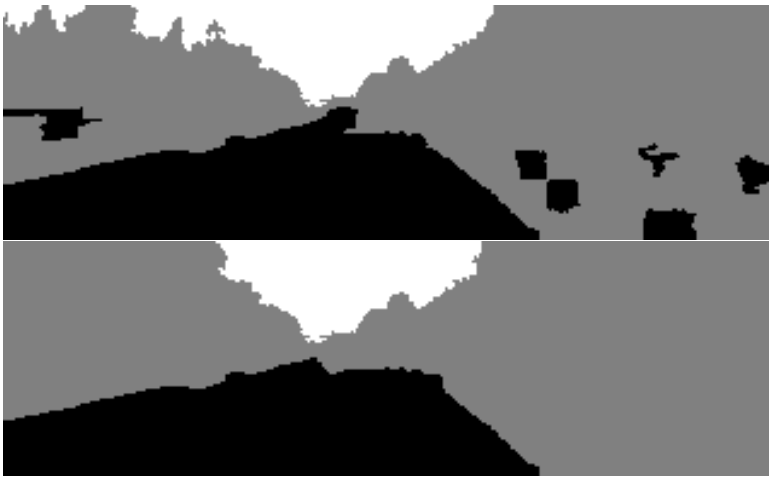

Figure 4.5: The denoised result using Markov random fields. Tough the image has become much smoother, some correctly labeled areas were regarded as noise (e.g., sky in the top left region).

4.7 Manual Labels

4.7.1 Manual training and test sets

The noticeable amount of noise in generated labels brings uncertainty of the validity of our test results. Therefore we used 20 manually labeled images from separate scenarios (referred to as *manual_test* set) to explore the real world applicability of our system.

To further investigate the influence of noisy training data, we repeated out experiments on another set of training data (*manual_train* set), which consisted of 25 patches randomly selected from each category of 40 manually labeled images. Only *manual_test* set was used for test here, since it would be meaningless to test on noisy labels.

4.7.2 Outcome

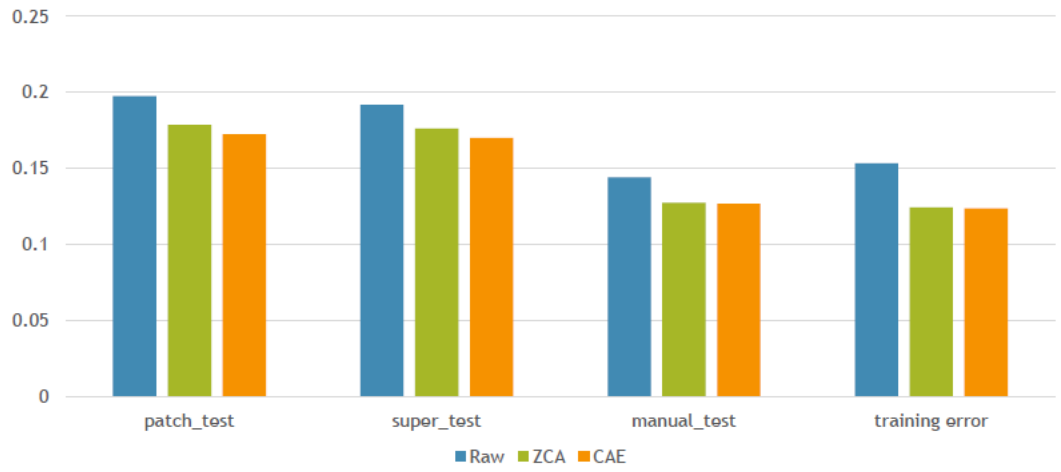

Figure 4.6: Training and test error of different approaches (denoted by columns) on different test sets or test methods (denoted by colours). The smallest error on noiseless test data (orange bar in the third column) was acquired by the combination of convolutional auto-encoders and super-pixel labeling.

As was shown in the figure 4.6, all these methods had substantive increases (around 5%) in accuracy when tested on noise free data. If in the cases where ZCA whitening and convolutional auto-encoders are used, we achieved around 12.7% error rate, which is almost the same as the noise ratio we have in the training data.

	gen_train set	*manual_train* set
Raw	0.1442	0.1320
ZCA	0.1274	0.1173
CAE	0.1267	0.1049

Table 4.2: Test error of different training sets on *super_test* set using super-pixel labeling.

Comparing Table 4.2 with Table 4.1 we can see, although only a small amount of training data was engaged, the test result on manual labels shows an improvement of 1% to 2%.

Chapter 5

Discussion

In this chapter, we first proposed a road scene understanding approach that worked well in our experiments. Then, we compared two unsupervised learning methods (convolutional auto-encoders and PCA), and analyzed the applicability of super-pixel denoising. After that, we discussed the limitations of our study and stressed out potential future works.

5.1 Best Practice

It is shown in Figure 4.6 that, convolutional auto-encoders based methods yielded the best result in our experiment settings. If super-pixel labeling was applied in the inference step, we could further improve the perdition accuracy by approximately 0.5%. Therefore our best choice would be to use convolutional auto-encoders to pre-train on unlabeled data, then use the trained parameters to initialize a convolutional neural network, and use super-pixel labeling for inferring the labels of an input image.

In the best case, this approach gives an accuracy of 89.51% on noiseless test data, which exceeds the baseline by 2.7%. When trained on the data set with a noise ratio of 12.74%, it can reach a same noise level (12.67%) as of the training data, hence shows its robustness to noise. A comparison of prediction results and generated labels are shown is Figure 5.1, as we can see, the prediction result of our approach looks better than the generate labels in this case.

We can know from comparing Table 4.2 and 4.1, when trained on a small amount of data, the advantage of unsupervised feature learning using convolutional auto-coders becomes more noticeable, which confirms the effectiveness of convolutional auto-coders in leveraging the usability of unlabeled data.

Moreover, better results were obtained when trained on noiseless data than noisy data, even if the amount of data set is much smaller (40 images vs. 264 images). Therefore we believe out performance can be further improved given sufficient amount of well labeled data.

An other benefit of this approach is, its running time in the prediction stage becomes linear with respect to the size of input data. If the network architecture is fixed, its time complexity will only be dominated by the time of performing image segmentation algorithms. Moreover, it is traceable in the sense that we can control its running time

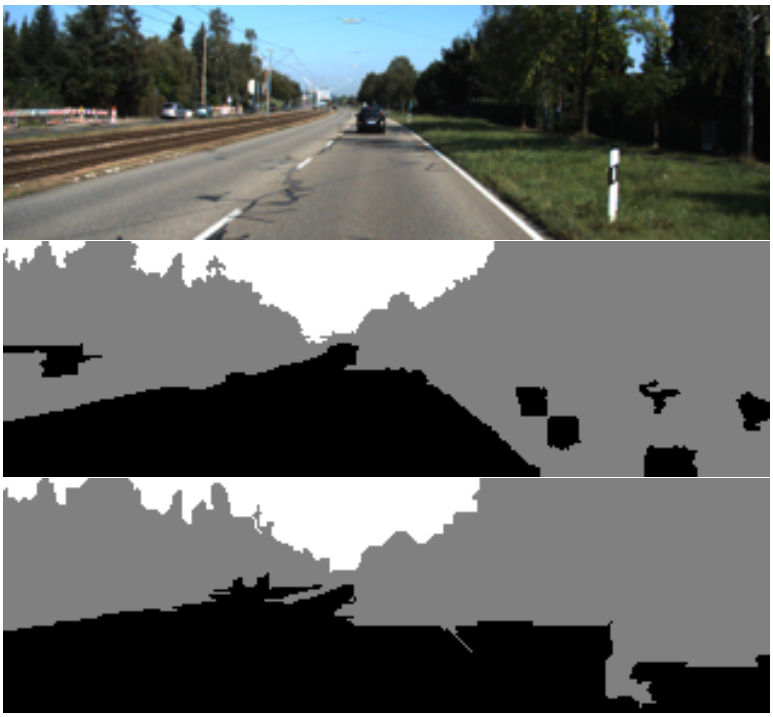

Figure 5.1: Example of an input image (top), prediction result (middle), and generated labels (down).

by adjusting the network architecture and the number of super-pixels, at the expense of accuracy.

Linear and tractable running time is a desirable property in real world applications. For example, in autonomous driving systems, we want to reduce the running time to the minimum since our surrounding environment can change rapidly. Some algorithms for high level handcrafted features are of superlinear time complexity, which constrains their practical utility.

5.2 Convolutional Auto-encoders vs. Principal Component Analysis

As shown in our experiment results, PCA could also produce comparable outputs to convolutional auto-encoders. However convolutional auto-encoders have shown their advantages in many aspects. Convolutional auto-encoders provide us insight into how to adjust the architecture of neural networks. It learns local features instead of global features, which is a more reasonable approach for image data. In the prediction stage, convolutional auto-encoders can be used directly on input data, while PCA require a pre-processing operation.

In terms of computational resources, the memory requirement for training convolutional auto-encoders is independent of the size of input data if online optimization

algorithms are used (e.g., stochastic gradient decent). On the other hand, PCA is often subject to high space complexity when performed on high resolution images. For instance, for a 1024 by 1024 color image, the size of the covariance involved will be $(1024 \times 1024 \times 3)^2$, which will take terabytes of memory. Although algorithms such as partial least squares [Geladi and Kowalski 1986] can be used to approximate PCA, they are of their own limitations.

5.3 Super-pixel Denoising

Although the Markov random fields can be used for noise reduction on a super-pixel level in many cases, it has a serious drawback. The assumption that a strong correlation exists between adjacent super-pixels does not hold in general. For instance, in the case that a pedestrian (represented by a super-pixel) on the road has been predicted successfully into the "object" category, because its surrounding area has been labeled as "road", using MRF denoising will turn the label of that pedestrian super-pixel into the same category as its neighbours. More importantly such misclassifications will cause safety issues if applied on autonomous driving systems, hence MRF denoising is not considered as a beneficial method for our purpose.

5.4 Limitations and Future Work

We have analyzed the pros and cons of various methods, and suggested a weakly supervised method which is of acceptable accuracy, tractable running time and is robust to noise. On the other hand, there are several limitations of our study.

First, because we did not have an adequate amount of well labeled data, we can not give a performance upper bound of this approach. Second. our experiment was conducted using data from the same source (KITTI Road data set). Its generalizability on other data sets has not been tested. Third, because we used fixed size patches as input data, in some cases a patch does not contain enough information for inferring the label of the centering pixel [reflection pic]. Another aspect we chose to neglect in our study is the relationship between consecutive images in a video.

In the next step, we will first focus on addressing the data issues mentioned above. Then we are going to investigate techniques for capturing the global information of an image and the correlation between consecutive images. Furthermore, we may want to explore a more complex road scene understanding task. For instance, we can add more categories denoting pedestrians, vehicles, and side walks.

Chapter 6

Conclusion

In this paper, we suggested a weakly supervised learning approach for basic road scene understanding that excels both in theory and practice. This approach consist of 1) pre-training using convolutional auto-encoders, 2) training using convolutional neural networks initialized by pre-training result, 3) predicting using super-pixel labeling.

We also explored the merits and flaws of various techniques of the same purpose. We compared two image segmentation algorithms and several available implementations, investigated unsupervised learning methods convolutional auto-encoders and principal component analysis from different perspectives, and evaluated the pros and cons of Markov random fields for graph denoising.

Moreover, we analyzed the computational complexity of our approach, showed that the inference running time is linear with respect to the size of input data. A software system was built accordingly for basic road scene understanding tasks, and tested on both generated and manually annotated labels, which gave an accuracy of 89.51% in the best case. In addition, we developed an open source library for convolutional auto-encoders using Matlab and validated its mathematical correctness.

By using a weakly supervised approach based on convolutional neural networks, we avoided the disadvantages of handcrafted features. More importantly, we proposed and compared different methods for improving the efficiency and effectiveness of such approach. Our approach provides a tractable running time and good generalizability, which is desirable for real word applications such as autonomous driving systems.

Bibliography

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Susstrunk, S. 2012. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 34*, 11, 2274–2282. (p. 17)

Alvarez, J. M., Gevers, T., LeCun, Y., and Lopez, A. M. 2012. Road scene segmentation from a single image. In *Computer Vision–ECCV 2012*, pp. 376–389. Springer. (p. 2)

Alvarez, J. M., LeCun, Y., Gevers, T., and Lopez, A. M. 2012. Semantic road segmentation via multi-scale ensembles of learned features. In *Computer Vision–ECCV 2012. Workshops and Demonstrations* (2012), pp. 586–595. Springer. (p. 2)

Álvarez, J. M., López, A. M., Gevers, T., and Lumbreras, F. 2014. Combining priors, appearance, and context for road detection. *IEEE Transactions on Intelligent Transportation Systems 15*, 3, 1168–1178. (p. 1)

Bengio, Y. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning 2*, 1, 1–127. (pp. 8, 9)

Bishop, C. M. et al. 2006. *Pattern recognition and machine learning*, Volume 4. springer New York. (pp. 4, 11, 12, 14)

Felzenszwalb, P. F. and Huttenlocher, D. P. 2004. Efficient graph-based image segmentation. *International Journal of Computer Vision 59*, 2, 167–181. (pp. 15, 16)

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. 2013. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*. (p. 19)

Geladi, P. and Kowalski, B. R. 1986. Partial least-squares regression: a tutorial. *Analytica chimica acta 185*, 1–17. (p. 39)

Hinton, G. E. and Salakhutdinov, R. R. 2006. Reducing the dimensionality of data with neural networks. *Science 313*, 5786, 504–507. (pp. 9, 13)

Hoiem, D., Efros, A. A., and Hebert, M. 2005. Automatic photo pop-up. *ACM Transactions on Graphics (TOG) 24*, 3, 577–584. (pp. 17, 18, 20)

Hornik, K., Stinchcombe, M., and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks 2*, 5, 359–366. (p. 5)

Kittler, J. and Föglein, J. 1984. Contextual classification of multispectral pixel data. *Image and Vision Computing 2*, 1, 13–29. (pp. 15, 27)

LeCun, Y. and Bengio, Y. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks 3361*, 310. (pp. 5, 6)

Lemme, A., Reinhart, R. F., and Steil, J. J. 2010. Efficient online learning of a non-negative sparse autoencoder. In *ESANN* (2010). Citeseer. (p. 8)

Lowe, D. G. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, Volume 2 (1999), pp. 1150–1157. Ieee. (p. 1)

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59. Springer. (pp. 9, 10, 23, 26)

Palm, R. B. 2012. Prediction as a candidate for learning deep hierarchical models of data. Master's thesis. (p. 26)

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1985. Learning internal representations by error propagation. Technical report, DTIC Document. (p. 5)

Szeliski, R. 2010. *Computer vision: algorithms and applications*. Springer Science & Business Media. (p. 15)

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (2008), pp. 1096–1103. ACM. (p. 9)