Ryerson University

CP8208

Soft Computing and Machine Intelligence

Naive Road-Detection using CNNS

Authors:
Sarah Asiri - Domenic Curro

April 24 2016

Contents

1 Abstract

Being able to detect road for safe navigation in autonomous cars is an important problem in computer vision. Road scene segmentation is often applied in autonomous driving and pedestrian detection [7]. In this project, we apply deep learning techniques to train a convolutional neural network to segment road from non-road in images. We will be using the LeNet network as our model of choice[6], as defined in Caffe's example MNIST Challenge solution.

2 Introduction

One of the important problems in computer vision is the detection of road segments for autonomous driving. In this project, we apply deep learning techniques to train a convolutional neural network to classify and further detect road segments in images. We are using the LeNet network, defined by Caffe's example MNIST Challenge solution. The main goal of this project is to build a classifier that, given an image, is able to create a pixel wise segmentation of the image in terms of what is road and what isn't (i.e., binary classification). We apply image segmentation using the SLIC Super-Pixel algorithm. We use segmentation to break the scene up into patches and further classify those patches.

3 Motivation

The problem on hand is well known in the field of computer vision. Road detection is a huge part of developing autonomous cars. Such cars are self-governing and should require no human factor interference. It is important to maintain road safety and the safety of pedestrians' crossing when autonomous cars are on the road. From that perspective, detecting road accurately is of extreme importance. We also chose this project because it contains the fundamentals of machine learning that we need to apply. Supervised learning, Classification and neural networks.

4 Related Work

A variety of work on image segmentation and classification of images has been found in the literature. However, very few paper apply super-pixel segmentation to input

Figure 1: KITTI Dataset[1]

images before training them. Rasmussen and Scot demonstrated in [8] the use of the same methodology in our project, using super-pixels for trail detection in images using CNNs. Moreover, semantic scene segmentation for road images was introduced in [3] aims at assigning every pixel in an image one of the labels. Similar work can be found in [7] where their algorithm learn from machine–generated labels in order to label road scenes in an image using CNNs.

5 Dataset

For this project we will be using the KITTI Road Detection dataset, Which is a part of the KITTI Vision Bench-marking Suit[1] the data contains 289 hand-labeled images. The road and lane estimation benchmark provided by KITTI contains three different categories of road scenes, a sample is shown in figure 1:

1. uu - urban unmarked (98/100)
2. um - urban marked (95/96)
3. umm - urban multiple marked lanes (96/94)

6 Approach

We follow a supervised learning approach, using binary classification, to train our model. We chose to use Convolutional Neural Networks (CNNs), as they are fit to work with image data and the top-most competitive papers for solving visual recognition problems prefer CNNs. CNNs are a variation of the Neural Network, designed to accept images as input. We discuss their architecture in details in upcoming sections. As mentioned before, this is a classification problem and our classification pipeline is as follows:

- Input Images: The images are segmented into 32x32-pixel patches and these patches are fed to our model as input. Each patch is labeled 0 for non-road and and 1 for road.
- Learning Model: We use our training set to train our model to learn which patches are road and which aren't.
- Evaluating Results: Finally, we evaluate obtained results by providing new images, different from the training set, to predict its labels and compare the predicted labels to our ground truth labels.

We started by Splitting the data into training, validation and testing sets. The splits are 70% 10% and 20%, respectively.

Once we have obtained our results, we perform data augmentation and run our model again on the augmented images. Data augmentation methods provides improvement for models as it creates additional examples extending original dataset, as well as providing invariance to geometric and photometic distortions. Our augmentations include: rotation, mirroring, adjusting brightness, contrast, gamma, and adding Gaussian noise.

7 Convolutional Neural Networks

ConvNets are Neural Networks that learn a set of filters that produce feature maps, which later helps discrimination in the fully connected layers. The final layer of the network is a Softmax, which is network's loss function. Similar to a neural network, a ConvNet learns by adjusting a set of weights throughout the model, where features per layer are extracted from the previous layer's output. The network uses feed-forward feature extraction and trains convolutional filters by back-propagating classification error using gradient descent. In a standard fully-connected Neural Network of m layers, three kinds of layers exist:

1. Input layer: Accepts fixed input, image patches.
2. Hidden layers: Takes previous layer's output as input.
3. Output layer: Takes output of the last hidden layer as input.

An abstract representation of the layers is provided in Figure 2.

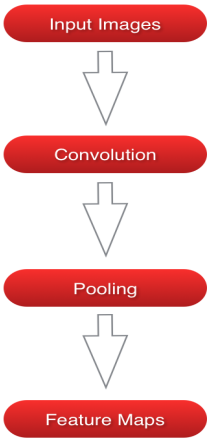

Figure 2: Convolutional Neural Networks

CNNs Vs. NNs

Convolutoinal Neural Networks have several advantages that makes them a perfect training model for images that traditional fully connected neural networks. In a NN, every edge has its own weight (parameter), while in a CNN, weights of different connections are shared, minimizing the number of free parameters, in return, minimizing the model's complexity. Additionally, a minimum variation in the learned input image can cause the NN to recognize it as a new image that it has never seen before, while a CNN is able to recognize it due to its pooling layer. Finally, NNs Do not take into consideration the input topology, while CNNs do, Since images have 2D structure, pixels' order and their correlation is of extreme importance.

7.1 LeNet Architecture

The network used was built in Caffe, specifically LeNet. It was introduced by [6] back in 1998. It made a huge impact in the world of computer vision and image classification and it is still used today. This network consists of an input layer, followed by a convolutional layer, pooling layer, convolutional layer, pooling layer, convolutional layer, relu, fully connected layer, relu, fully connected layer, and finally a softmax output layer. We discuss the details of these layers shortly.

7.1.1 Convolutional Layers

A convolutional layer convolves an input feature map with a filter to produce an output feature map. This approach slides the filters across the image, one pixel at a time, performing dot product to produce a signal response. The filter is a two-dimensional set of weights that the network will learn.

7.1.2 Pooling Layers

A pooling layer takes in a feature map and performs the simplest technique to reduce it's size - the image is broken up into 2x2 sections, and the largest feature is kept (in the case of max pooling), where the other features are discarded. This has the property of making the image half as tall and half as wide. Pooling has the effect of reducing the number of features in the network as well as allowing the network to "zoom in" on the image while focusing on important features.

7.1.3 Fully Connected Layer

A fully connected layer performs a weighted dot product of all previous layer's features.

7.1.4 ReLu Layer

ReLU (The Rectified Linear Unit) activation function,a variation from the original LeNet, which computes the function **f(x)=max(0,x)**. It nullifies all negative features. It returns zero when x ¡ 0 and then linear with slope 1 when x ¿ 0. It is observed that deep CNNs that use ReLUs train several times faster than their equivalents with tanh units[5].

7.1.5 Soft Max Layer

A soft max is a probability distribution across a set of classes. The results of the soft max is used to determine the amount of error that the network generated. This output will be used in back-propagation to determine the at fault weights.

$$P(y = j|x) = \frac{e^{x^{Tw_j}}}{\sum_{k=1}^{K} e^{x^{Tw_k}}}$$

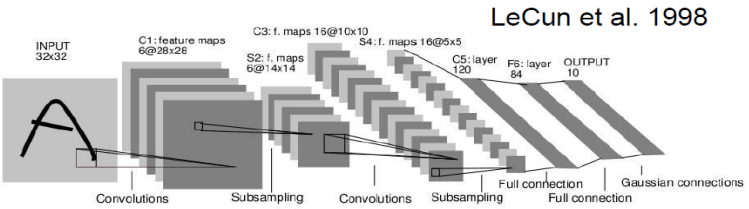

Figure 3: LeNet Architecture [6]

Figure 3 shows the detailed architecture of the LeNet network.

8 Tools and Implementation

To simplify our problem, we segment our input data into super-pixels and generate labeled patches, which we use for network training. The segmentation algorithm used for super-pixel generation is SLIC (Simple Linear Iterative Clustering). It works by clustering pixels in "the combined five-dimensional color and image plane space to efficiently generate compact, nearly uniform superpixels"[2].

We used matlab to generate the data, Caffe Framework [4] for training the segmented pieces for detection of road. For images' super-pixeel segmentation, we apply the SLIC[2] algorithm using VLFeat tool.

8.1 Data Preparation

We started by Splitting the KITTI dataset into training, validation and testing sets. The splits are 70% 10% and 20% respectively. Performing segmentation on the dataset, we segmented every image to 32X32 pixel patches and then labeled them if they contained more that 50% road as specified by the ground truth provided by the KITTI dataset.

8.2 Data Augmentation

We performed data augmentation to increase the size of our dataset and create invariance to slight geometric transformation and photometric distortions. Performed augmentations to patches generated from the original images include: rotation, mirroring, adjusting contrast, gamma effects and adding Gaussian noise.

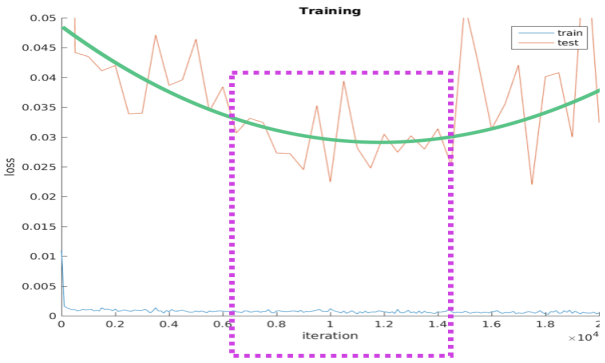

Figure 4: Plotting Loss Vs. Iterations

8.3 Model Training

The defaults of Caffe's MNIST challenge are set to:

- Learning Rate: 0.01
- Weight Decay:0.0005
- Momentum: 0.9

8.4 Monitoring Training

We plotted the loss against iterations for both training and validation data. Shown in figure 4. The training and validation loss were monitored for divergence. Divergence is clear after 15k iterations, so the models generated 6.5k through 15k are checked for performance.

9 Results

Looking at table 1, there exists an increase in performance by adding rotation+flips, to the dataset. The increase continues when we add brightness changes. We see a decrease in performance when contrast is included and an increase when noise is added. We also include the F1 accuracy result found in [3]'s work.

The reason for this decrease in performance is likely due to a misstep in modeling the contrast data. If the contrast data did indeed match reality (within the KITTI dataset), than we should expect an increase. The change of our dataset grows with

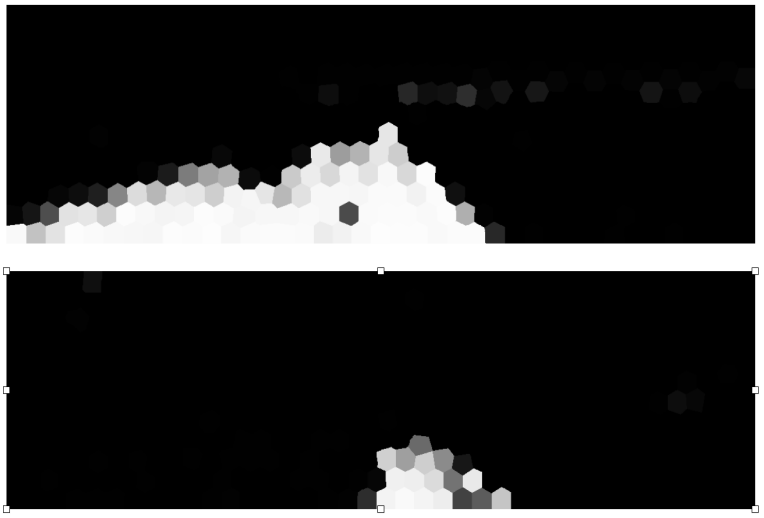

Figure 5: Model's Confidence values

each. Prior to any geometric or photometric augmentations our dataset. Running our model on the original dataset, prior to photometric our photometric dataset, with an iteration number of 7500. After that we trained it again while gradually adding the augmented data, starting by adding flipped and rotated patches, 80964 patches. Then we added brightness modified patches, 485784 patches, on top of the previous data. Figure 5 shows our model's confidence at that stage. After that we added contrasted images to the previous and ended up with 647712 patches. On top of that, we added patches with gamma effect, 809640 patches. Finally, we added noise augmented data. With the performed augmentation on our dataset, we have observed that augmented data does not properly simulate the distortion caused by shadows. Hence, our data was modeled wrong. Figure 7 shows a visualization of our model's results. All results for the previous can be found in table 1. Additionally, the confusion matrix is provided in figure 6. It is important to note, based on obtained results, that the confusion matrix can be improved using the noise invariant model. However, it wasn't included due to time constraints.

In contrast to other published approaches, our F1 score is relatively low, but in light of the fact that our approach is naive in nature, our results show some merit. As stated earlier, we classify each patch on its own devoid of any surrounding image context.

Table 1: Results

Confusion Matrix		Predicted Class	
		Road	Non-Road
Actual Class	Road	4590	2698
	Non-Road	1477	31483

Figure 6: Confusion Matrix showing the number of correctly predicted classes against those falsely predicted.

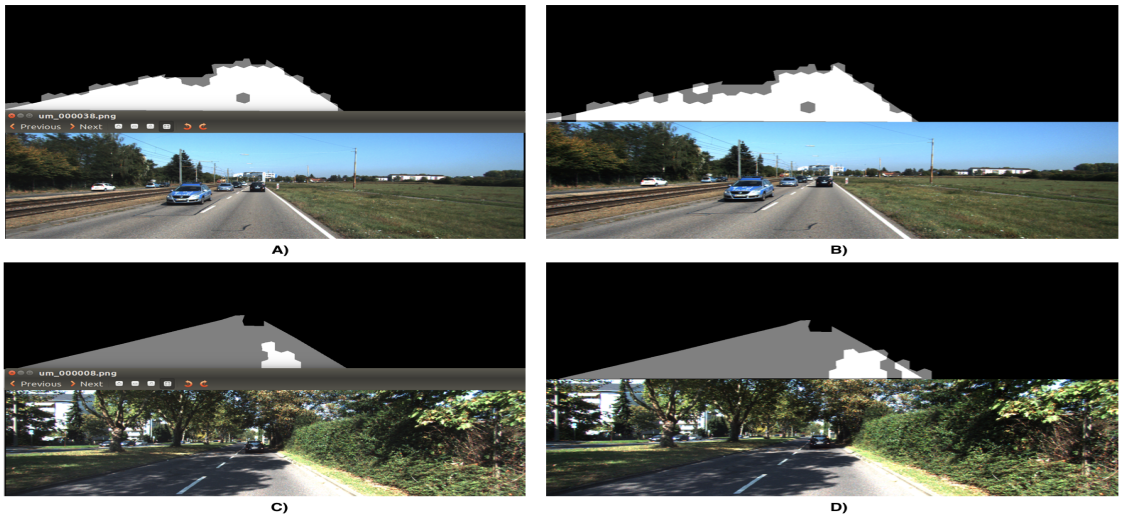

Figure 7: Visualization of Predicted Results: A and C show model's prediction against ground truth image with only geometric Augmentation. B and D show model's prediction against ground truth image with photometric Augmentation

Data	Iteration	F1 Score
Original Data	3500	67.97%
+ Flips & Rotations	7,500	68.12%
+ Brightness	10,000	68.74%
+ Contrast	12,000	68.25%
+ Gamma	12,000	67.70%
+ Noise	5,000	69.87%
Other work[3]	-	86.5%

10 Conclusion

In conclusion, we tackled the problem of road detection in images in the context of autonomous driving. We followed a supervised learning approach to classify road segments with the use of convolutional neural networks. Using the KITTI dataset and super-pixels, we generated our input data and also performed several types of augmentation to increase its size and improve out model's invariance. Compared to other work, obtained results could have been better but we had limited resources in terms of time and computational power. Also, after performing augmentation on our dataset, we have observed that augmented data does not properly simulate the distortion caused by shadows. Hence, our data was modeled wrong.

11 Future Work

For future work, we can improve our augmentation values in order to generate augmented data that represent real life lighting and shadows by determining a proper set of parameters for shadowed and over-lit road patches.

References

[1] Kitti vision benchmark suite.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels. Technical report, 2010.

[3] Jose M Alvarez, Yann LeCun, Theo Gevers, and Antonio M Lopez. Semantic road segmentation via multi-scale ensembles of learned features. In *Computer Vision–ECCV 2012. Workshops and Demonstrations*, pages 586–595. Springer, 2012.

[4] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[7] Yann LeCun and Antonio M. Lopez. *Road Scene Segmentation from a Single Image*, volume 7578, pages 376–389. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[8] Christopher Rasmussen and Donald Scott. Shape-guided superpixel grouping for trail detection and tracking. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 4092–4097. IEEE, 2008.