CS230

Video Segmentation within image frames for autonomous driving.

Wenfei Du
Department of Statistics
Stanford University
wdu@stanford.edu

Mayank Malhotra
Stanford Graduate School of Business
mayank01@stanford.edu

Hiro Tien (Kai Ping)
Stanford Graduate School of Business
Stanford School of Earth, Energy & Environmental Sciences
hirotien@stanford.edu

Abstract

We build a model to help autonomous vehicles segment movable objects, such as cars and pedestrians, within image frames. We implement two general approaches based on FCN/SegNet and tuned the specifics. For each, we learn the weights on upsampling layers added on top of convolutional layers taken from VGG 16 pretrained on imagenet. We face limitations due to our low sample size and achieve an validation IOU of 0.04.

1. Introduction

Autonomous vehicles are not able to quickly identify different kinds of moving items on the road (such as a person or a car) as effortlessly as a human is able to. To solve this problem, we investigate the possibility of building a model which helps autonomous vehicles segment movable objects, such as cars and pedestrians, within image frames.

Specifically, we predict segmentations of different movable objects appearing in the view of a car camera. Our approach is to use the VGG 16 convolutional neural net and then add a multilayer upsampling network. The network consists of a series of convolution, relu, and max pooling layers followed by a series of upsampling, convolution, and relu layers to construct a dense pixel-wise class prediction map. The unpooling is done via deconvolution for FCN and unpooling for SegNet. The input to our algorithm is an image frame from video recordings and the output is a predicted class of object for each pixel.

2. Related work

In our project, we drew heavily from Segnet and FCN for semantic regression. Here, we will briefly describe the inspiration behind these related works. Segnet was inspired by the unsupervised feature learning architecture proposed by Ranzato et al. [1]. The key learning module is an encoder-decoder network. An encoder consists of convolution with a filter bank, element-wise tanh non-linearity, max-pooling and sub-sampling to obtain the feature maps. For each sample, the indices of the max locations computed during pooling are stored and passed to the decoder. The decoder upsamples the feature maps by using the stored pooled indices. It convolves this upsampled map using a trainable decoder filter bank to reconstruct the input image. This architecture was used for unsupervised pre-training for classification. A somewhat similar decoding technique is used for visualizing trained convolutional networks [2] for classification. The architecture of Ranzato et al. mainly focused on layer-wise feature learning using small input patches. This was extended by Kavukcuoglu et. al. [3] to accept full image sizes as input to learn hierarchical encoders. Both these approaches however did not attempt to use deep encoder-decoder networks for unsupervised feature training as they discarded the decoders after each encoder training.

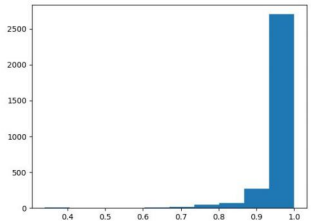
Segnet approach differs from these architectures as the deep encoder-decoder network is trained jointly for a supervised learning task and hence the decoders are an integral part of the network in test time. Other applications where pixel wise predictions are made using deep networks are image super-resolution [4] and depth map prediction from a single image [5]. The authors in [5] discuss the need for learning to upsample from low resolution feature maps which is the central topic of this paper.

Our FCN approach draws on recent successes of deep nets for image classification [6, 7, 8] and transfer learning [9,10]. Transfer was first demonstrated on various visual recognition tasks [9,10], then on detection, and on both instance and semantic segmentation in hybrid proposal classifier models [11, 12, 13]. FCN then re-architects and finetunes classification nets to direct, dense prediction of semantic segmentation.

3. Dataset and Features

The dataset is provided by Baidu through CVPR 2018 WAD Video Segmentation Challenge on Kaggle. It contains a large number of segmented and original driving images with multiple labels such as [car, 33 ; motorbicycle, 34 ; bicycle, 35 ; person, 36 etc.]. This comprise of over 95.2GB of labelled image data for our segmentation. These are taken from 44 videos of cars on different roads using right or left facing cameras in the training set and 12 videos in the test set. The competition evaluates seven different instance-level annotations, which are car, motorcycle, bicycle, pedestrian, truck, bus, and tricycle. Although more labels are provided, we consider only the seven evaluation classes and combine the rest into a background class.

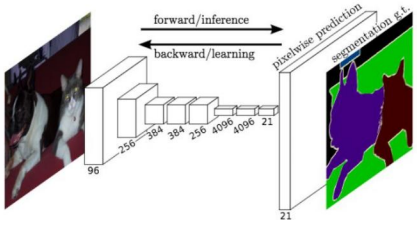
For our purpose of building and testing different models, we downsample the images for ease of data transfer and model building and testing. The original image size is 3384x2710px, which was larger than our memory constraint. We downsample by a factor of 6 to 564x224px. The training label format is given the in the same size as the original images, with pixel values encoding the label and instance (int(PixelValue/1000) = label and PixelValue % 1000 = instance). To downsample these, we use the corresponding locations for each pixel and took the mode as the value. This is not exact, but since it would only matter for details at the boundary of each instance, we ignored these errors. We also only train on every tenth images out of around 500 to 1000 total from each video. The reasoning is that many of the images were similar for the same video, since they seem to be frames taken every second. For the same reason, we divide the training set by video into a 80/20 split to get a validation set, so that frames taken from the same video would not bias our validation error. From this procedure, we obtain 3142 images from 36 videos in our training set and 380 images from 4 different videos in our validation set.



The histogram above shows the distribution of percentage of pixels with object labels across our training set. Since most of the images were more than 90% background, we also build a training set out of the 224 images with less than 90% background as positive examples and a sample of 224 images with more than 90% background as negative examples. This dataset performs slightly better than the full dataset. Since each minibatch is more likely to contain positive examples, we avoid converging early to a solution where background is predicted for each pixel. We thus focus on the results using this dataset below, but we note that the small sample size, as well as the remaining skew towards images with more background than object pixels, is a major limitation.

4. Methods

In this section, we broadly describe Segnet and FCN which form the basis of our video segmentation work.

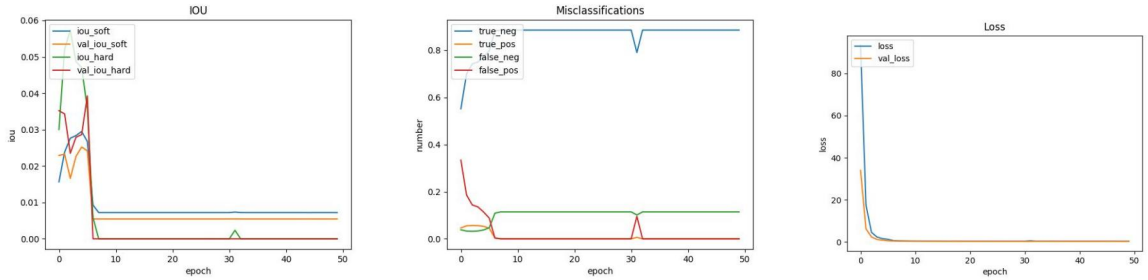


The image above shows our general encoder-decoder approach. We use transfer learning by taking weights from VGG 16 pretrained on imagenet data. On top of these layers we add upsampling, convolution, and relu layers in the same pattern as VGG 16 in order to learn a classification for each pixel. The last layer computes a weighted categorical cross entropy loss. The FCN inspired version uses a deconvolution layer for upsampling and convolution, while the Segnet inspired version uses unpooling by transferring the index of the maxpool for the corresponding convolutional layer in VGG 16. In addition, we test versions of both architectures that include time information from the videos. Instead of using the VGG 16 output for the frame itself, we concatenate the outputs for the frame, the previous frame, and the next frame. This allows information sharing across a few frames in time.

5. Experiments/Results/Discussion

We train using the Adam algorithm with 0.001 learning rate. We use a minibatch size of 25 images, the largest that can fit in memory. The choice of learning rate and minibatch size is influenced by our data constraints. Since each minibatch was unlikely to contain many samples with objects, testing with a smaller minibatch size and larger learning rate led to early convergence to a pattern where every pixel was classified as background.

We introduce weights in the categorical cross entropy loss and tune these. With equal weights on each category, the classifier learns to predict background, since most of the samples still contain majority background pixels. We instead use weights of 0.01 for background and .99/7 for each other class. Use of much larger weights than this led to instability in training. We then do fine tuning in two stages. First, we run our individual frame and three frame versions of FCN and Segnet. Then, we consider the best version, add regularization to prevent overfitting due to our small sample size, and tune our regularization parameters. The following results are from our best performance, which is the three frame version of FCN with regularization weight 0.1 on the coefficients.



The above plot on the far right shows the training and validation loss and shows our learning path. We note that both losses decay to close to zero within the first five epochs. The IOU measure is the main metric of interest. The plot on the left shows this measure, which sums for each object class the intersection of true and predicted points and divides by the union. The hard IOU uses the zero or one prediction, while the soft IOU uses the actual output probability. The plot in the middle shows the average misclassification percentages, which

contribute to the IOU calculation. True negatives refer to background pixels classified as background and true positives refer to object pixels classified as the correct object. We see that at around epoch five, we have a balance between the number of true positives and false positives and between the number of true negatives and false negatives. The validation IOU is also largest at epoch five, with a value around 0.04. After this epoch, the gains in the loss are due to the optimization learning to classify every pixel as background, and we see a decrease in both true and false positives and in IOU.

6. Conclusion/Future Work

We have tested two of the prominent architectures for image segmentation, including adding time dimensions for video frames, and were able to detect objects with an IOU of 0.04. The most limiting factor we faced was the small sample size, and in the future, one important step would be to collect more positive example images from each of the videos. Alternatively, an approach using object detection prior to segmentation may avoid this issue, since the tasks of detecting bounding boxes and segmenting with those boxes may require less data. Also, we have not fully explored ways to use the video information. We have included a basic model allowing information sharing between the previous and next frames, but a more complex model would use a recurrent neural net. Other information we have not yet used are variables provided with each video, including camera side and road. Finally, there is more work to be done in tuning the number and size of layers, which we have not covered in this paper.

7. Contributions

Team Members	**Contributions**
Hiro Tien	- Research on existing literature (Segnet) - Drafting milestone paper - Analyze existing frameworks for transfer learning purposes - Writing final paper
Wen Fei	- Research on existing literature - Finalize chosen frameworks and methodologies for the project - Coding for selected neural networks - Analysing model results
Mayank	- Setting up Amazon AWS - Research on existing literature (FCN) - Drafting milestone paper - Building poster

References

[1] M. Ranzato, F. J. Huang, Y. Boureau, and Y. LeCun, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in CVPR, 2007.

[2] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in CVPR, pp. 2528–2535, IEEE, 2010.

[3] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, “Learning convolutional feature hierarchies for visual recognition,” in NIPS, pp. 1090–1098, 2010.

[4] C. Dong, C. C. Loy, K. He, and X. Tang, “Learning a deep convolutional network for image super-resolution,” in ECCV, pp. 184–199, Springer, 2014.

[5] D. Eigen, C. Puhrsch, and R. Fergus, “Depth map prediction from a single image using a multi-scale deep network,” in NIPS, pp. 2366–2374, 2014.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. 1, 2, 3, 5

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. CoRR, abs/1409.4842, 2014. 1, 2, 3, 5

[9] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. DeCAF: A deep convolutional activation feature for generic visual recognition.

[10] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation.

[12] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Simul- ´ taneous detection and segmentation