On the Importance of Label Quality for Semantic Segmentation

Aleksandar Zlateski[*] **Ronnachai Jaroensri**[*] **Prafull Sharma** **Fredo Durand**
Massachusetts Institute of Technology
[*]**Equally contributed**

Abstract

Convolutional networks (ConvNets) have become the dominant approach to semantic image segmentation. Producing accurate, pixel–level labels required for this task is a tedious and time consuming process; however, producing approximate, coarse labels could take only a fraction of the time and effort. We investigate the relationship between the quality of labels and the performance of ConvNets for semantic segmentation. We create a very large synthetic dataset with perfectly labeled street view scenes. From these perfect labels, we synthetically coarsen labels with different qualities and estimate human–hours required for producing them. We perform a series of experiments by training ConvNets with a varying number of training images and label quality. We found that the performance of ConvNets mostly depends on the time spent creating the training labels. That is, a larger coarsely–annotated dataset can yield the same performance as a smaller finely–annotated one. Furthermore, fine–tuning coarsely pre–trained ConvNets with few finely-annotated labels can yield comparable or superior performance to training it with a large amount of finely-annotated labels alone, at a fraction of the labeling cost. We demonstrate that our result is also valid for different network architectures, and various object classes in an urban scene.

1. Introduction

Driven by the increased computational power of modern hardware, researchers have revived the use of convolutional neural networks (ConvNets) for computer vision. [27, 30] While it achieves state–of–the–art performance on many important learning tasks, supervised training of ConvNets is known to be data intensive. For example, training a ConvNet for general image recognition may require millions of labeled images. [17]. In domains for which labeled data is expensive to produce, additional tricks, such as fine–tuning or using lower–quality data, need to be employed for ConvNets to be feasible.

Semantic image segmentation is a task for which labeled training images are very costly to collect. Producing training images involves assigning a semantic class label for each individual pixel of a given image. This is particularly hard, as object boundaries could be complex and difficult to accurately annotate [26].

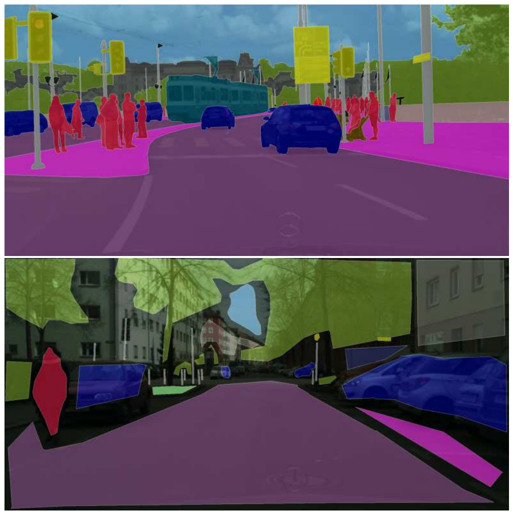


Figure 1: A finely annotated (top) and a corsely annotated (bottom) image from the CityScape's dataset.

Despite their cost, accurate (pixel–perfect) annotations are believed to be essential for high–quality segmentation, as nearly all dataset publishers spend all of their efforts on generating such annotations [10, 4, 5, 11, 9, 3]. Recently, the CityScapes [9] dataset provided a large set of coarsely annotated images alongside a smaller set of finely annotated ones. Utilizing these coarse annotations during the training procedure have been shown to improve the performance of ConvNets. For example, nearly all top performers on CityScapes benchmark [9] utilize coarsely labeled images [32, 34, 8, 33].

Driven by this observation, the main goal of this paper is to quantify the impact of label quality, and thus labeling cost, to performances of semantic segmentation ConvNets. Given a limited budget for annotating data, what is the optimal strategy that yields best performance?

Because real–image datasets contain only up to a few thousand of annotated images [9, 10, 5], we decide to use computer graphics to generate a synthetic dataset for our study. The benefit is two–fold. Firstly, we are able to generate a very large number of images with little human effort. Secondly, because the complete information about the scene is known, we can produce perfect annotation for every pixel in every image in the dataset. For this particular work, we use CityEngine software to procedurally generate a city, and render street–view images using an open-source rendering software called Mitsuba [14].

In order to quantify the coarseness of labels and correlate the human effort required to produce them, we introduce a metric for label quality which was inspired by CityScape's coarsely labeled images[9], and propose an algorithm for producing labels with various qualities. We then measure the amount of human–hours required for producing labels at different quality levels, and use the results of the measurements to estimate the amount of human–hours required to produce labels in our dataset.

Finally, we trained state–of–the–art ConvNets on various combinations of label qualities and numbers of training images, and analyze their respective performances. We repeated these experiments on an additional network architecture and analyzed results on different object classes to ensure the validity of result. Our main findings are as follows:

- ConvNets' performance highly correlates more with the human–hours spent producing the labels and less on the quality of each individual annotation. This means that a larger number of coarsely annotated image can yield the same performance as a smaller number of finely annotated ones.
- It is not optimal to invest time in producing a large number of fine labels. Competitive and/or superior results can be obtained by using a large number of coarsely labeled images combined with a small number of finely annotated ones, which require fewer human–hours to produce.
- While we cannot suggest a general rule for the optimal ratio of fine and coarse labels, our results suggest that, in order to achieve maximal performance, while minimizing human effort, equal or larger amount of human–hours should be spent on producing coarse labels.
- The trade–off for using coarser labels is training time. The training time is highly correlated with the number of images in the training set.

1.1. Terminology and details

Throughout the paper we will be using the terms *fine* labels to refer to images annotated with pixel–level precision. We use the term *true* labels to refer to the *fine* labels generated using computer graphics. With *coarse* labels we refer to the labels that are not *fine*. *Coarse* labels can have different *quality* based on the error metric introduced in the next section.

To simplify the analysis and better isolate the effect of label quality, we primarily focus on the specific problem of two–class semantic segmentation (vehicle vs non-vehicle) of street images, but repeat our experiments for multi–class segmentation.

2. Datasets

Having large dataset is crucial for our experiments because we are compensating coarseness with quantity. Our experiments shows appreciable trends only after using more than 10k images per class for training. While MS COCO [18] boasts $> 200k$ labeled images, its annotations are coarse. Other publicly available real–image datasets have very limited sizes (at most 10k images across all classes [6]). Therefore, we decided to generate and use synthetic data for our study. In this section, we will discuss existing datasets and contrast them with ours. Then, we give details on how our dataset was generated, and how we synthetically coarsened the labels. Finally, we describe the experiment we performed to estimate the human effort required to generate labels at each quality level.

2.1. Existing Synthetic Datasets

When compared to real datasets, synthetic datasets typically contain orders of magnitude more images. The SYNTHIA dataset [24] contains 13,407 images of urban scene with pixel-accurate annotation of several classes such as vehicle, pedestrian, road, and sidewalk. "Driving in the Matrix" dataset [15] contains up to 200,000 images captured from the video game Grand Theft Auto V.

While these datasets boast impressive number of images, they have some drawbacks. First, several consecutive frames within both datasets are often correlated, significantly reducing their effective sizes. Second, the scenes represented in each dataset are limited. The SYNTHIA dataset appears to contain only a few city blocks [24], while Driving in the Matrix dataset contains a single city. Many of the images in Driving in the Matrix dataset are desert highway, and it is unclear how diverse its urban scenes are. Lastly, because nearly no additional information is available, it is very difficult to sample the dataset while avoiding overlapping training and test sets.

Despite these limitations, we utilized SYNTHIA in some of our experiments, as their annotations comprehensively covered street object categories (e.g. cars, pedestrian, road and sidewalk, etc.).

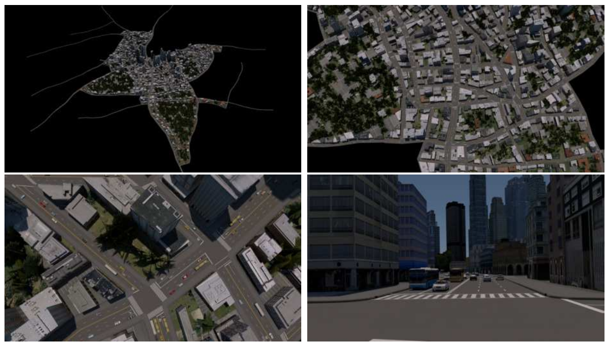


Figure 2: Auto City overview (top left), low rise district (top right and bottom left) and a street scene (bottom right)

2.2. The Auto City Dataset

Because existing datasets' limitations, we chose to generate our own dataset of urban, street view scenes. This allowed us to have a full control over the diversity of the dataset. We used the CityEngine software to procedurally generate the models of our city, and we used Mitsuba [14] to render these models.

Procedural city CityEngine provides a grammar for specifying and parameterizing urban objects, which can be used to generate buildings, sidewalks, cars, etc. with variations. It provides rule–based zoning for easy generations of many cities at once with residential and/or commercial zones.

We followed CityScape's "International city" example to generate cities that included a small financial center, low–rise buildings area, as well as residential areas with vegetation. Figure 2 shows the bird's–eye view of a generated city, as well as a sample of a street–level shot we used for training the ConvNets. The city covered a total area of 15 km^2, around 40 kms of roads, and contained $3,782$ cars.

Rendering For high quality, realistic rendering we used Mitsuba [14], an open source physically based renderer. To mimic images taken from a driving vehicle, we used camera locations at the height of 1.5 meters in the middle of each lane. Images that did not contain a vehicle were discarded. This resulted in a little over $150,000$ images.

For realistic lighting, Preetham [21] sun/sky model implemented in Mitsuba was used. We used a path tracer integrator with the maximal depth of 5 to render images at 768×576 pixels. We chose 16 samples per pixel as a good trade-off between rendering speed and the amount of noise.

Training, test, validation splits The images were divided into 3 non–overlapping **image pools**, sorted east to west. This way each of the three pools covered all of the three city zones (residential, financial, and low rise), while being completely separated from each other. The three pools served the purpose of training, validation, and test image sets.

2.3. Simulating Coarse Label Quality

Inspired by the CityScapes dataset (Fig. 1), we produced coarse labels such that they fit within the true labels' boundaries. The coarse annotations were represented by polygons approximating the true boundary of the objects. More detailed coarse labels were represented with higher degree polygons whose vertices are placed closer to the true boundary of the object.

To formalize the quality of a semantic labeling, we used the maximal distance between the polygonal label and the true object boundary as the measure of labeling error (Δ). Given Δ, we separated the true boundary and the coarsened one by eroding the true label by $\Delta/2$. We then simplified the resulting polygon using the Douglas–Peucker algorithm [12], a commonly used algorithm for polygon simplification. The algorithm attempts to simplify the polygon while allowing for an error of at most $\Delta/2$. This method allowed for a fine control over $\Delta_{\max}$ and produced coarse labels very similar to the ones of CityScape's dataset, which are produced by humans (Fig. 1). An illustration of the proposed algorithm steps are given in Fig. 3 (top row), as well as results obtained by applying the algorithm (bottom three rows). Note that the resulting annotations contain unlabeled pixels; these pixels are ignored during the ConvNet training procedure.

2.4. Estimating Human Effort

In order to understand the relationship between the quality of coarse labels and human effort required for producing such, we designed an experiment to precisely measure differences in human–hours needed for producing labels of different qualities, by measuring the time it took for humans to generate such labels.

We asked four individuals to label a set of 50 diverse images with 1) fine labels and 2) labels similar to the ones produced by our algorithm with Δ of 4,8,16 and 32. For coarse labels, the subjects were asked to trace polygons inside and outside the vehicles labeling both the vehicles and the background. The subjects were also explained the boundary constraint and the notion of maximal allowed error (Δ). Furthermore, the subjects were first given 10 already labeled images, as the ones in Fig 3 (bottom) to serve as warm–up examples to ensure the subjects knew how accurate they needed to be.

For producing fine labels, the subjects were instructed to trace the outlines of vehicles with pixel accuracy.

After the warm–up, the subject were asked to label 40 unlabeled images at each quality level. As the same images were used for each quality levels, the subjects were instructed to take long breaks between labeling images with different

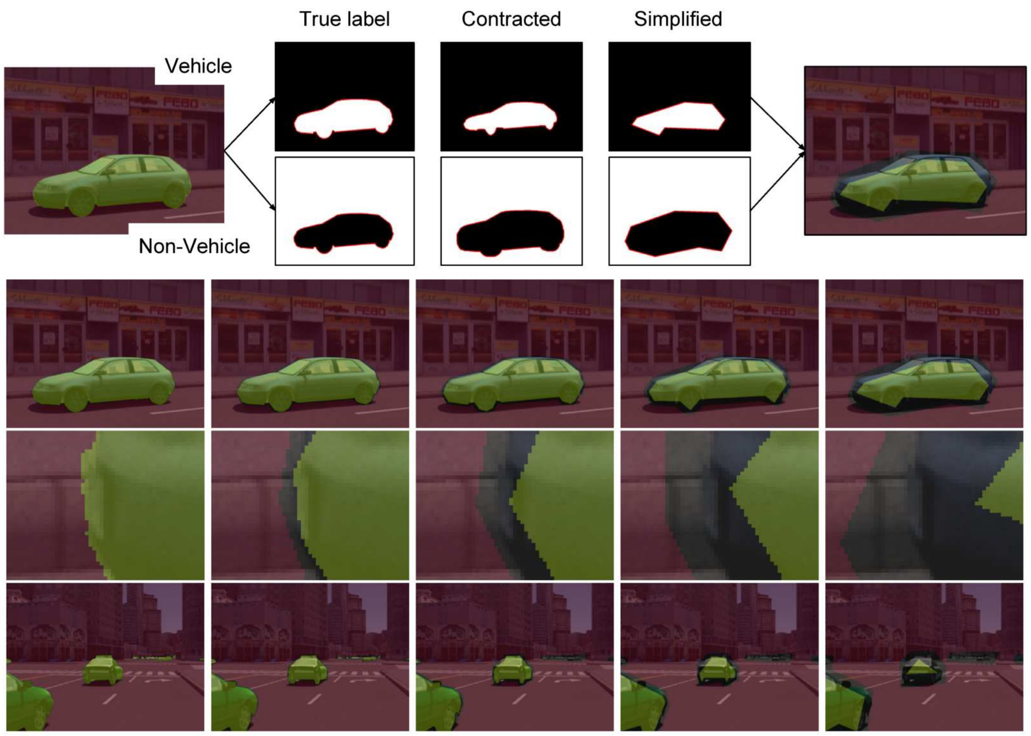


Figure 3: The three steps of our method for producing coarse labels (top). True labels and labels obtained with our simplification algorithm using $\Delta_{\max} = 4, 8, 16, 32$ (middle and bottom)

quality.

For each subject and label quality we computed the average time spent per image. In Fig. 4, we show the mean and standard deviation of the average times spent labeling. The results confirm that creating fine-label was extremely difficult, taking 4 times as much the labeling time for 4px margin. The coarsest 32px margin was the fastest, 15× faster than the fine labels.

We note the large standard deviation for producing fine labels, and relatively small deviations when labeling coarser images. This is reasonable, as perfectly following the object boundary is much harder than drawing approximate labels. It also depends on ones vision, dexterity, etc., and will thus vary more among different individuals. The data suggests that there is a large time penalty for producing pixel–perfect labels, as compared to coarse labels with relatively small error.

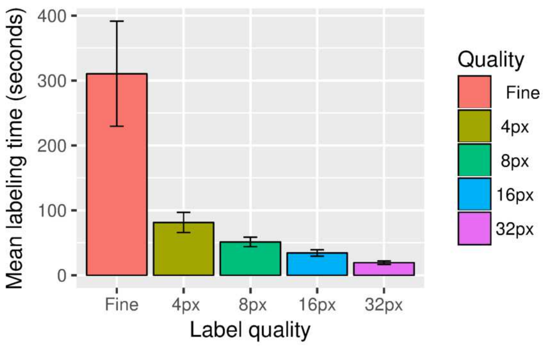


Figure 4: Mean time spent on labeling images with different quality labels.

3. Experimental Setup

3.1. Reference ConvNets

Most state-of-the-art semantic segmentation ConvNets typically consist of an encoder network and a corresponding decoder network that is followed by a pixel-wise classification layer [19, 28, 13, 23, 2, 20, 33, 32, 34]. Often, the encoder network is borrowed from architectures used for image classification, such as VGG16 [29] or AlexNet [17], and is pre–trained on classification datasets such as the ImageNet [25]. The decoder network is responsible for producing features for each pixel for classification, and greatly varies among different approaches.

Our primary ConvNet was based on FusionNet [22], as it is a relatively simple encoder–decoder network, closely resembling many popular networks in recent literature (e.g. DeconvNet [20], SegNet [2, 1] and UNet [23]).

We modified the default FusionNet by reducing the number of channels in each layer, and removing residual blocks in order to simplify the model, and allow for faster training. This resulted in a ConvNet with approximately 1M parameters, which allowed for relatively fast training while producing highly accurate results. It also allowed us to quickly perform many experiments in order to optimize the training parameters.

In order to ensure that our findings are valid for different network architectures, we also ran our experiment using the FCN16 architecture [19], which we consider to be the most different from FusionNet, while being simple enough to run many experiments with.

Unlike the symmetric architecture of FusionNet, FCN16 architecture only uses a light-weight decoder/bilinear upsampler to generate its final segmentation. The architecture itself can be thought of as a sliding an image recognizer such as VGG16 or AlexNet over the image to produce a dense prediction [29, 17]. The decoder layers function mainly to recover lost detail from overlapping receptive fields.

We also ran a subset of experiments with even more sophisticated architecture, DeepLab-ResNet, which is one of the top-performing entries in the CityScapes benchmark [7, 9], and is very costly to train. However, we will not be reporting full result with this architecture.

3.2. Training and Evaluation

We trained each network using the Adam [16] gradient descent optimization algorithm. We were able to carefully tune the modified FusionNet training parameters due to its fast training. For FCN16, We used default parameters, except for the learning rate for which we used a constant value of 10^{-5} instead of decaying learning rate as suggested in the original papers [19]. We found this configuration to yield superior performance on our datasets.

Not all segmentation networks are initialized by pretrained weights, especially in biomedical domain. To show that our results are consistent regardless of whether the network weights were pre–traned, we initialized the weights of FCN16 using VGG16 [29], while we trained the modified FusionNet from scratch. We monitored validation loss, and ended training early when it over–fitted or the validation loss converged.

We used intersection-over-union, a common metric for semantic segmentation task, to evaluate our ConvNets [10]. The validation set and the test set are obtained by drawing $3,000$ images from the validation and test pools respectively. The quality of the annotations used for validation always matched the quality of the annotations used for training. However, we used the true labels for the test set to ensure that all ConvNet were evaluated on the same standards.

3.3. Experimental Conditions

We trained our ConvNets using training sets with various label qualities. Additionally, we also varied the size of the dataset in order to study the impact of dataset diversity. We used 5 training sets starting at $3,000$ images and doubled the size until we reached $48,000$ images. For each training set, we trained our ConvNet with 5 label qualities from fine labels to coarse labels with maximal errors ($\Delta_{\max}$) of 4, 8, 16 and 32 pixels.

The images within a given training set were sampled from a spatial neighborhood within the training pool. The neighborhood size varied in longitude proportionately to the number of images within the training set, while covering the whole city in latitude. This way, the scene diversity would also vary with the number of images, yielding a total of 25 experimental conditions.

Fine–tuning Experiments Another typical technique to deal with limited dataset is finetuning. The network is pre–trained on a larger dataset for a related, or simplified, task, and then fine–tuned on the limited dataset. In our case, this would be equivalent to training on a large amount of coarse labels, and fine–tuning on a smaller number of fine labels. We study the effect of label quality in this setting by finetuning the networks pre–trained on training sets of $12,000$ or more of coarse labels (with maximal error of 4 up to 32 pixels).

4. Results and Analysis

We present and analyze the results of training two different network architectures on our synthetic dataset, as well as the results of training the FCN16 architecture on the Synthia dataset [24]. Through analysis, we aim to answer the following questions:

1. What is the impact of the label quality and the training set size to the overall ConvNet performances?

2. How can we optimally invest labeling time? Is it better to invest time in producing fine or coarse labels, or a combination of both?

3. Are these results valid for different ConvNet architectures and/or differet datasets and labels?

We first focus on the results of our modified FusionNet architecture. We then show that the results of the other architectures and data–set confirm our findings.

Impact of Quality and Quantity The performance of our modified FusionNet trained on various training sets, using the standard intersection over union (IoU) metric [10], are

Table 1: Mean IoU of our modified FusionNet architecture.

Training	Single quality					Coarse + 1k fine			Coarse + 3k fine		
Num images	3k	6k	12k	24k	48k	12k+1k	24k+1k	48k+1k	12k+3k	24k+3k	48k+3k
fine–labels	85.26	88.75	89.61	91.89	**93.93**	n/a	n/a	n/a	n/a	n/a	n/a
4px error	84.97	86.93	90.06	91.56	93.33	90.58	92.11	**94.02**	90.61	92.49	**94.31**
8px error	84.20	86.20	88.61	90.58	92.01	90.17	91.95	93.65	90.62	91.42	93.64
16px error	82.15	83.97	86.92	88.42	90.65	89.60	90.44	92.38	89.81	90.75	92.96
32px error	78.30	91.96	83.57	86.53	88.42	89.41	91.26	91.41	89.32	90.08	92.03

shown in Table. 1. Unsurprisingly, increasing either the quality of the coarse labels or the size of the training set improves the performances of the trained networks. We observe similar trends in our fine–tuning experiments. Using more finely labeled images for fine–tuning, resulted in a higher performance increase. This result is expected, because, intuitively, both larger training set size or better label quality increase the total amount of annotated pixels, hence the amount of useful data in the training set.

Optimal Time Investment On Fig. 5a, we plot the performance of our trained ConvNets against the estimated time required for producing the training set. Here, the orange markers represent networks trained on coarse labels, while the red labels represent the networks trained on fine labels; the shapes represent different qualities of the coarse labels.

As seen in Fig. 5a, the coarse labels performances (shown in orange) roughly lie on a line. This means, when training on coarse labels, the performance correlates strongly with the overall time spent producing labels, and not only with the quality. That is, a larger dataset of coarse images can yield the same overall performances as a smaller dataset with more accurately annotated labels. This is also illustrated by the points circled in Fig. 5a, which show that the same performance is achieved when using training sets of different label quality and different number of images.

Additionally, the resulting performances of ConvNets trained on fine labels (red markers) also lie on a line that is strictly below the orange one. This means that, for a given labeling time budget, coarse labels will outperform the fine ones. Conversely, for a desired target performance, one would only need to spend a fraction of time labeling coarse labels in order to achieve the goal. While the absolute best performance is achieved by training on maximal number of finely labeled images, the accuracy is just slightly higher than the performances on coarse labels which require nearly an order of magnitude less labeling time (note the logarithmic scale). This highlights the difficulty for human to produce fine labels.

Fine–Tuning Experiments Table. 1 also shows the performance of fine–tuned networks. The best performance (boldfaced) in each category is very similar. Fine-tuning with just 1,000 images yielded nearly the same performance as 48k fine labels, which would be extremely expensive to produce; fine–tuning with 3,000 fine images yielded a superior performance. This suggests that, training on many coarse labels, and fine–tune on few fine labels may be a more optimal way when labeling cost is a concern.

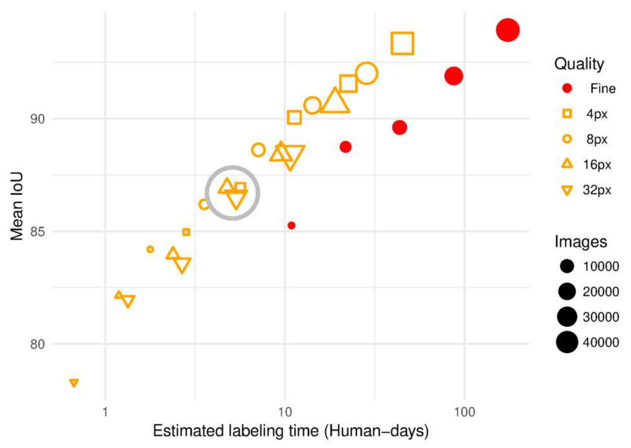
The natural next question is what is the optimal ratio of finely and coarsely labeled images (or the time spent producing fine vs coarse labels). On Fig. 5b, we overlay the performances of the fine–tuned networks. The networks that are fine–tuned with 1,000 fine labeled images are shown in green, while the network fine–tuned with 3,000 fine labeled images are shown in black. Fitted line for each group is shown.

In Fig. 5b, the black line (fine–tuning with 3k images) crosses the orange line (single quality) at around 20 days, which is where the time labeling fine data equals time labeling coarse data (producing $3,000$ finely annotated images would take about 10 days). Additionally, the green line, which starts at 7 days (where coarse-to-fine labeling time ratio is > 2), is always above the orange line. This suggest fine–tuning would be cost effective only when the time spent labeling coarse images equal time spent labeling fine images (the alternative is to spend more time labeling all coarse images). This result could be specific to our dataset, however.

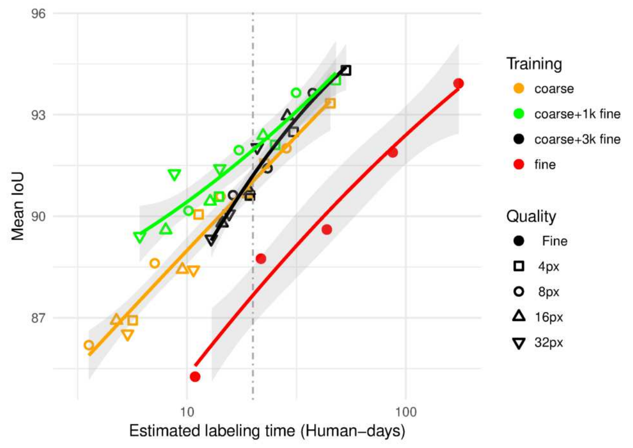
Generalizability of our Findings On Fig. 6 we show the performances of the more sophisticated FCN16 network architecture as a function of the total time spent producing labels for all the training configurations with at least $12,000$ training images. The yellow cluster (coarse labels) forms a line, and the red cluster (fine labels) is, again, below the yellow cluster, confirming the trend we saw with modified FusionNet.

Note that we observe more noise for the results where less human time is spent labeling. This is because we trained FCN16 "out of the box", without extensive tuning of training parameters.

However, the noisy results still confirm our findings. The red line (ConvNets trained only on finely annotated images)



(a)



(b)

Figure 5: (a) Estimated labeling time vs IoU for the vehicle category for single quality experiments (b) Zoomed in of (a) with fine-tuned experiment plotted. The circled points in (a) has nearly the same performance and time spent labeling, but they have different quality, showing that coarse label could be compensated with quantity. In (b), green and black points represent ones trained with coarse and fine–tuned with $1k$ and $3k$ fine labels respectively.

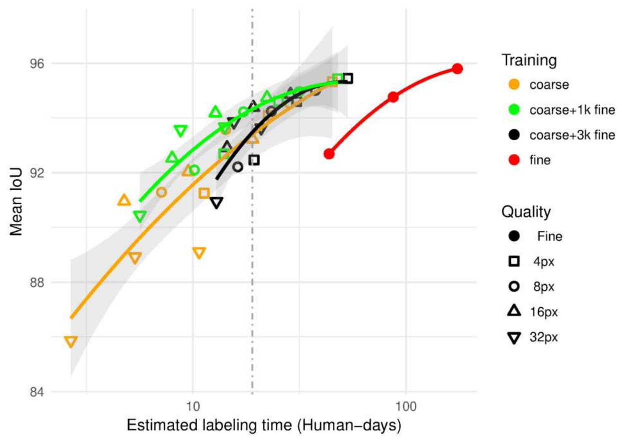


Figure 6: Performances of the FCN-16 network model.

is below all other clusters, confirming that using only finely annotated images is not optimal. The green cluster (ConvNets fine–tuned with $1,000$ images) is "above" the orange one (ConvNets trained solely on coarse labels), confirming that producing $1,000$ finely labeled images is better than producing none. The black line (ConvNets fine–tuned with $3,000$ images) crosses the orange line roughly at the x coordinate – 20 human–days.

Finally, the overall performances of the best performing networks fine–tuned with $1,000$ and $3,000$ images have competitive performances to the network trained on a large set of finely labeled ones, at a fraction of labeling time. Thus, fine–tuning with small set of fine labels can yield comparable performance as training on large set of fine labels.

Additionally, partial experiments were performed using even more sophisticated, DeepLab-ResNet, architecture, which is very costly to train. The subset of performed experiments confirmed our findings.

Generalizability to other semantic classes and datasets. On Fig. 7 we show the result of the FCN16 networks trained on different the Synthia [24] dataset using various configurations.

We trained the networks to distinguish among 5 label classes, and show the performances on labeling humans, vehicles and road. Including humans was of particular importance, as their labels are highly non–convex, which is in contrast to vehicles used in our dataset.

Being much smaller dataset, we could trained the networks on training set sizes of $2,500$, $5,000$ and $10,000$ images. The fine–tuning training configurations were not performed, due to the limited dataset size.

As described above, the nature of the Synthia dataset resulted in high inbalance of class occurrences, which resulted in additional noise. However, the results clearly show that for coarse labels, overall performances depend on the labeling time. Also, given a limited time budged, producing coarse labels is more time–efficient.

Note that, in Fig. 7, we had assume that the time to create labels at different quality levels for each of SYNTHIA semantic class is the same as vehicle class in our dataset. While this may not be true, we expect the trend to hold because we expect fine labels to be costlier than coarse labels for any semantic class.

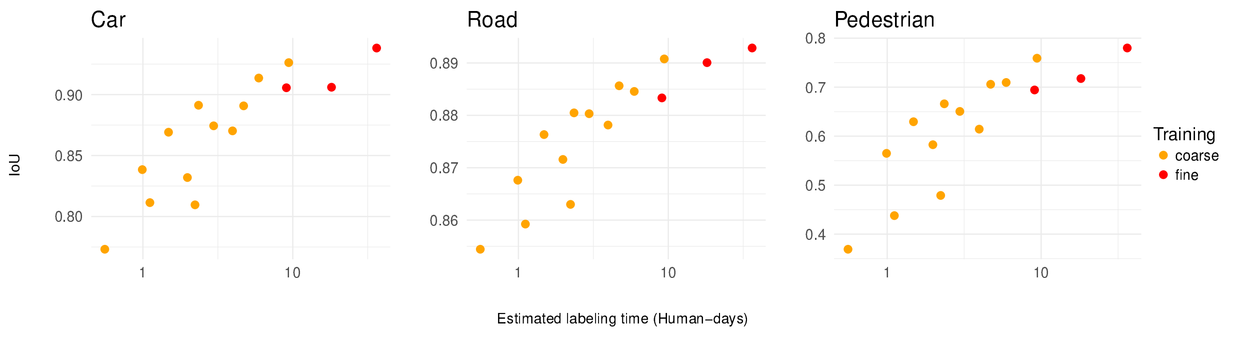


Figure 7: Results of training the FCN16 ConvNet on 15 different datasets. The sizes of the datasets were $2,500$, $5,000$ and $10,000$, and for each size both fine labels as well as labels with $\Delta_{\max} \in \{4, 8, 16, 32\}px$ were used.

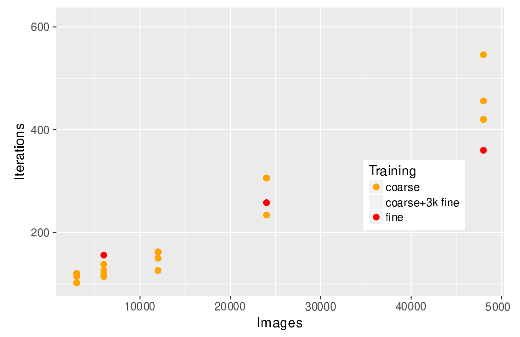


Figure 8: Training time (in iterations) vs number of training samples.

Training time Figure 8 shows the training time (in terms of iterations) it took our ConvNets to converge. The training time increases with training set size, so the trade off for using more coarser-labeled images is to run training longer. This suggests that, if faster training is the goal, human–hours should be spent on labeling images with higher quality coarse labels.

5. Conclusion and closing remarks

Our main take home message of the work presented is that, when creating a training dataset, spending time on producing coarse labels is arguably more important than producing fine ones.

Through experiments we showed that the overall ConvNet performance is strongly correlated with the amount of time spent producing coarse labels, and not the quality of each individual label. This suggests that datasets can greatly benefit by obtaining coarse labels through crowd–sourcing, without strong control of the label quality.

Using coarse labels for pre–training, and fine–tuning with finely annotated images allows for competitive performance to training on a large number of only finely annotated ones, which requires spending much more human–hours on the tedious task of labeling images. The optimal portion of labeling time that should be spent producing coarse and fine labels will vary on the specifics of the dataset, as a rule of thumb we suggest that one should spend at least the same amount of time on coarse labels as on fine ones.

While the overall performances solely depend on the human–hours spend labeling, the training times also depend on the total number of images. If one wishes to minimize the ConvNet training time, one should consider producing coarse labels with higher quality. This could be important for involving complex network architectures, and the training time becomes limiting factor.

Our findings apply for cases when a large number of unlabeled images are readily available, or easily obtainable – such as street–view images used in this paper. When the amount of training images is limited, such as in many medical applications (e.g. ADNI dataset [31]), generating best quality labels would still yield the best performance.

References

[1] V. Badrinarayanan, A. Handa, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015. 4

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 4

[3] S. M. Bileschi. *StreetScenes: Towards scene understanding in still images*. PhD thesis, Citeseer, 2006. 1

[4] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008. 1

[5] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008. 1, 2

[6] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. *arXiv preprint arXiv:1612.03716*, 2016. 2

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 5

[8] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 1

[9] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016. 1, 2, 5

[10] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015. 1, 2, 5

[11] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1

[12] J. Hershberger and J. Snoeyink. An o (n log n) implementation of the douglas-peucker algorithm for line simplification. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 383–384. ACM, 1994. 3

[13] S. Hong, H. Noh, and B. Han. Decoupled deep neural network for semi-supervised semantic segmentation. In *Advances in Neural Information Processing Systems*, pages 1495–1503, 2015. 4

[14] W. Jakob. Mitsuba renderer, 2010. *URL: http://www. mitsuba-renderer. org*, 3:10, 2015. 2, 3

[15] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, and R. Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *arXiv preprint arXiv:1610.01983*, 2016. 2

[16] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 4, 5

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[19] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015. 4, 5

[20] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, 2015. 4

[21] A. J. Preetham, P. Shirley, and B. Smits. A practical analytic model for daylight. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 91–100. ACM Press/Addison-Wesley Publishing Co., 1999. 3

[22] T. M. Quan, D. G. Hilderbrand, and W.-K. Jeong. Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics. *arXiv preprint arXiv:1612.05360*, 2016. 4

[23] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 4

[24] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on mputer Vision and Pattern Recognition*, pages 3234–3243, 2016. 2, 5, 7

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 4

[26] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1):157–173, 2008. 1

[27] D. Scherer, H. Schulz, and S. Behnke. Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors. In *Artificial Neural Networks–ICANN 2010*, pages 82–91. Springer, 2010. 1

[28] E. Shelhamer, J. Long, and T. Darrell. Fully convolutional networks for semantic segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):640–651, 2017. 4

[29] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4, 5

[30] D. Strigl, K. Kofler, and S. Podlipnig. Performance and scalability of gpu-based convolutional neural networks. In *2010 18th Euromicro Confer ce on Parallel, Distributed and Network-based Processing*, pages 317–324. IEEE, 2010. 1

[31] H.-I. Suk and D. Shen. Deep learning-based feature representation for ad/mci classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 583–590. Springer, 2013. 8

[32] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. Cottrell. Understanding convolution for semantic segmentation. *arXiv preprint arXiv:1702.08502*, 2017. 1, 4

[33] Z. Wu, C. Shen, and A. v. d. Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *arXiv preprint arXiv:1611.10080*, 2016. 1, 4

[34] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 4