LSUN: Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop

Fisher Yu Yinda Zhang Shuran Song Ari Seff Jianxiong Xiao

Princeton University

Abstract

The state-of-the-art visual recognition algorithms are all data-hungry, requiring a huge amount of labeled image data to optimize millions of parameters. While there has been remarkable progress in algorithm and system design, the labeled datasets used by these models are quickly becoming outdated in terms of size. To overcome the bottleneck of human labeling speed during dataset construction, we propose to amplify human effort using deep learning with humans in the loop. Our procedure comes equipped with precision and recall guarantees to ensure labeling quality, reaching the same level of performance as fully manual annotation. To demonstrate the power of our annotation procedure and enable further progress of visual recognition, we construct a scene-centric database called "LSUN" containing millions of labeled images in each scene category. We experiment with popular deep nets using our dataset and obtain a substantial performance gain with the same model trained using our larger training set. All data and source code will be available online upon acceptance of the paper.

1 Introduction

High capacity supervised learning algorithms, such as deep convolutional neural networks [9, 8], have created a discontinuity in visual recognition over the past three years, and continue to push the state-of-the-art performance to the next level (e.g. [6, 17, 7]). These models usually have millions of parameters which results in two consequences. On the one hand, the many degrees of freedom of the models support the extremely impressive description power; they can learn to represent almost any complex function and transformation automatically from the data. On the other hand, to search for the optimal settings for a large number of parameters, these **data-hungry** algorithms require a massive amount of training data with human-produced labels.

Although there has been remarkable progress in improving the **algorithms** (e.g. [6, 13]) or high performance computing **systems** (e.g. [17]) to train these models, there is little progress on pushing forward the **data** construction efforts to the next level. The ImageNet dataset [3], used by most of these algorithms, is six years old and has been heavily overfitted (e.g. the overfitting scandal on ImageNet [1]). The number of parameters in many deep models has exceeded the number of images on ImageNet, stretching the performance under the risk of the lack of bounded constraints [14, 10, 5]. Another dataset that recently came out, Places [19] for scenes (ImageNet is mostly for objects), is not much larger than ImageNet. While the models are getting deeper (e.g. [12, 13]) and the number of parameters is increasing, the size of the datasets for training and evaluation is not increasing, but rather lagging behind and hindering further progress of large-scale visual recognition.

It is not easy, however, to build a much bigger dataset than ImageNet or Places because of the constraints of human labeling effort. The construction of ImageNet and Places both required more than a year of human effort via Amazon Mechanical Turk (AMT), the largest crowd-sourcing platform available on the Internet. If we are looking for a N-times bigger dataset, it will require more than

N years of human annotation, which will not scale quickly enough to support the progress of deep visual learning. Clearly, we have no choice but to introduce a certain amount of automation in this process, in order to let data annotation catch up with the growth of deep models. However, because this annotation will be used to train and evaluate new models, we need to be extremely cautious to ensure the accuracy of the labels obtained (semi-)automatically. Furthermore, humans make mistakes quite often, especially when performing repetitive and tedious jobs, such as data labeling. Thus, a systematic way to check human labeling accuracy is a necessity.

In this paper, we introduce an integrated framework using deep learning with human in the loop to annotate large-scale image data. We focus on interface design, procedures for checking for human annotation, as well as labeling propagation to amplify human effort automatically. Our procedure has **precision and recall guarantees** to ensure labeling quality, reaching the same level of performance as fully manual annotation.

To demonstrate the power of our annotation procedure and enable further progress of deep visual learning, we have construct a scene-centric database called "LSUN" with millions of labeled images in each scene category. We experiment with a popular deep learning model (AlexNet [8]) using our dataset and obtain a substantial performance gain with the same model trained using our bigger training set. We will continue to annotate more data through our platform, and we expect to obtain a dataset $100\times$ bigger than ImageNet or Places within a year with a comparable budget.

1.1 Related Works

There have been several existing datasets focusing image classification tasks on objects [4, 3] or scenes [18, 19]. They have make possible learning feature representation. Although there are several million images in those datasets, there aren't many images in each category, which leads to low dataset density. To conquer this problem, people have been using various techniques to augment the datasets [17]. However, the learned representation still has all kinds of weird phenomenon [14, 10] when the input is slightly perturbed. Our dataset is aimed to be more than 10 times bigger than PLACES [19] and hundreds times than ImageNet [3] for each category, so it has much higher data density.

To construct such as large dataset, we have to label billions of images, which is not feasible for human computation. So we design a pipeline that can combine machine and human efforts. The pipeline has to be accurate, practical and fast. People have looked at this problem from an active learning perspective [11, 15, 2]. The key idea of active learning is to achieve greater machine learning model accuracy with less training labels. For more thorough review, please refer to [11].

Our pipeline is similar to active learning, but with a few key differences. First, we aim to **collect the data** without algorithm bias instead of **learning a model**, which is different from the goal of the most active learning algorithms that aims to learn the same model with less/least data. Second, our task is also easier in the sense that it is **acceptable to overfit a model** to adapt locally to the current target subset of images for splitting, without considering generability outside this set. And in each iteration, we remove the positive and negative data that are confidently identified and the images to split are shrinking in each iteration, so that the classifier in later iteration **doesn't need to generalize** to these identified data as well, but focusing on splitting the remaining data only. Last but not least, the primary goal of our project is on the construction a massive-scale image data set, and we **tail every step** to this practical goal, polishing every design details, including image crawling, duplicate removal, interface design, human quality control, etc.

2 Large-scale Dataset Construction

To build a large-scale image database, we have to acquire a large number of images for each scene category from the Internet to start with. As with the previous works, we turn to existing image search engine to avoid crawling the Internet by ourselves. Currently, all of our image URLs are from Google Images search. We use a technique based on scene adjective queries and synonyms to overcome image limit of each search query and we are able to obtain nearly 100 million URLs to relevant images for each query.

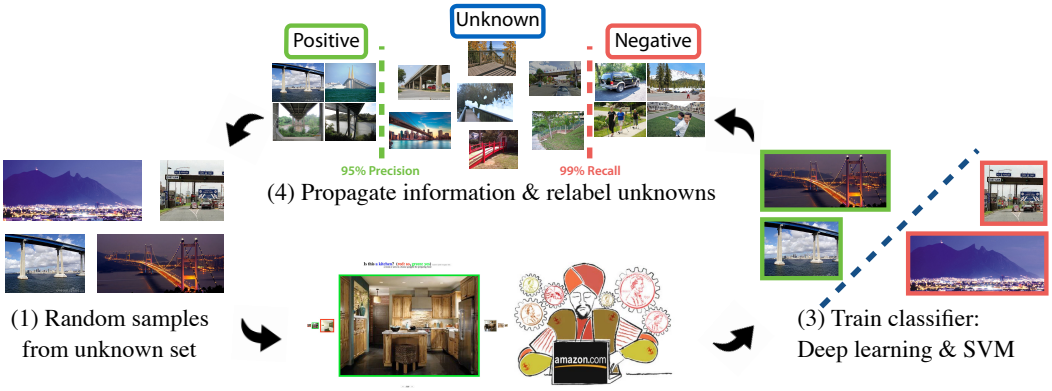


Figure 1: The overview of our pipeline. To annotate a large number of unlabeled images, we first randomly sample a small portion (1), and label them on Amazon Mechanical Turk using our interface (2). These small number of labeled images are utilized to train a binary classifier with deep learning feature (3). Then, we run the binary classifier on the whole unlabeled images. The images with high or low scores are labeled as positive or negative automatically, while the images ambiguous to the classifier are fed into the next iteration as the unlabeled images (4). The pipeline runs iteratively until the number of images in unknown set is affordable for exhaustive labeling.

Since the SUN database has a rich scene taxonomy, LSUN database has inherited the same list of scene categories. To generate the query of image URL, 696 common adjectives (messy, spare, sunny, desolate, etc.), manually selected from a list of popular adjectives in English (obtained by [19]), are combined with each scene category name and are sent to Google. Adding adjectives to the queries allows us to download a larger number of images than what is available in ImageNet and to increase the diversity of visual appearances. We also set the time span for each query three days and query all the three-day time spans since 2009. We find that a shorter time span returns noisier query results. We then remove duplicated URLs and download the raw images. For a common category such as bedroom, kitchen, bathroom, etc., we are able to obtain 50 to 80 million unique URLs. We are expanding this efforts. For each scene category, we also query relevant synonyms. For example, for the category "bathroom", we also query "washroom", "lavatory" and "restroom", which get us about 80 million unique URLs in total. In addition to scene categories, we also use the same method to collect image URLs for object categories. The difference is that we use a different set of adjectives that are more suitable to describe objects. For instance, we have obtained 80 million unique URLs for images relevant to "person".

To date, about 600 million images have been downloaded. A initial quality check is carried out before labeling those image. Only color images of 256×256 pixels or larger are kept. Unlike previous system [18, 19], we don't remove replicates in the initial pool of images, because it is very expensive to remove duplicates while avoiding overkill such tens of millions of images. In addition, we don't label many image, the duplicates in the original image dataset doesn't increase our labeling cost. The duplicate images usually have different compression quality, different sizes and even cropping. They augment our dataset naturally and it will be the decision of the algorithm designers how to utilize the duplication.

In the following two sections, we will first explain our labeling procedure using deep learning with humans in the loop, and then we will go into further details about how to massively crowd-source human labels from AMT.

3 Deep Learning with Humans in the Loop

Although we can collect tens of millions of relevant images for each category, the amount of human effort to label these images are prohibiting and cannot be scaled up. We propose to use a deep learning model with humans in the loop to amplify the human efforts to produce high quality labels semi-automatically. As shown in Figure 1, we continuously train new models based on the labels in the set of images we want to separate. We then use statistical tests to take out of the images that

the models are sure about for certain quality goal. Then new images are labeled from the remaining image set to train better models for it, and the iterations continue. The quality we seek for each split is to keep the ratio of positive images higher than 95% and to lose less than 1% recall among the negative images.

To make best use of the existing models and data, there are three stages in our pipeline depending how hard to separate the unknown set of images. Before we start, we extract features for each image from the model pre-trained on PLACES. First, the images are grouped based on similarity suggested by the pre-trained features. We use k-means clustering to group the similar images and the clusters with only positive or negative images can be taken out directly by ranking clusters and sampling images in the top clusters. Second, we learn multiple boundaries to separate the images in the feature space locally in each cluster and globally for all the images. We use various SVM models to learn the boundaries. Last, we learn the new features with deep learning models to adapt to the image sets that are hard to separate. The model is pre-trained on PLACES dataset and then the model is fine-tuned by the labeled images in the target set. The clustering is only done once, but two or three iterations are carried out in the second stage depending on whether we can make progress by cutting the existing feature space. After the several iterations of the last stage, there are usually hundreds of thousands of images left, that are hard to separate by the existing models, and we will label them all by humans. In the following sections, each step is elaborated.

3.1 Labeling Testing Set

To monitor the general labeling progress and prepare the testing set for data release, we collect a small testing set only based on manual label. This testing set is supposed to evaluate future scene classification algorithm without bias. Therefore, we first randomly sample from all the images in a category and label all of them on AMT twice. This stops when more than 1300 images are labeled as positive by two turkers. We then manually go through the images that are labeled as positives by at least one turker to make sure the labels for the testing set are correct. The labels obtained in testing set have very high quality.

A challenge to our pipeline is to make sure the labels we collect semi-automatically also have good quality. We exploit this testing set to help improve label quality. When we prepare for labeling interface, we can find out the common mistakes in the labels and use them as instruction examples and ground truth. Also, in Section 3.3, we use this set to figure out how many clusters we need to keep.

3.2 Bootstrapping with Deep Features

A good feature representation is essential to start our labeling pipeline. We use the deep learning feature from AlexNet [8] trained on PLACES [19]. In practice, we find the concatenation of fc7 and fc8 layers performs the best in most of the tasks. And also, since the purpose is to extract image information rather than classification, the feature extracted from the whole image is slightly better than the one from a random crop.

3.3 Quick Labeling by Clustering

Because we download tens of millions of relevant images for each category, there are a lot of similar positive and negative images in the initial set. Therefore, image similarity is an important cue to quickly find positive images and remove negative images. To make use of the similarity information, we first cluster images into different groups. We find that a lot of clusters contains high percentage of positive or negative images. To be specific, we do k-means clustering on 200,000 sampled images with correlation metric on the fc7 layer features of AlexNet. 2,000 cluster centers are obtained and then all the images are assigned to the nearest cluster centers. Each image is scored by the initial model and each cluster is assigned the median score of the images in it. All the clusters are ranked by their scores and top clusters are kept for human to label. Given the testing set, we can evaluate the recall of the positive images in the selected clusters. In practice, we seek a 95% recall, which requires no more than 700 clusters for all the categories. We keep at least 600 clusters and normally they can cover about 97% images.

To figure out which cluster to really keep, 100 images are sampled from each cluster and labeled by human. It means we can get about 60,000 labels for our data at this stage. We can then train a better SVM model than the initial one that may bias to PLACES statistics. Given all these labels and the new model, we do two things sequentially to split the whole set into three parts.

First, we use the label as validation set to find the SVM score thresholds for 95% precision and 99% recall. The images with score higher than the first threshold is treated as positive and the images scored lower than the second threshold is treated as negative. The remaining images are kept.

Second, for the remaining images in each cluster, if the rest of the cluster has more than 95% positive labeled images, the remaining images are treated as positive. To decide which clusters to discard, we have to put them all together to decide the proper threshold for positive image ratio. So we rank all the remaining clusters by the ratio of positive images. By the number of total images in each cluster, we can estimate the number of positive images in it. Then we find the cluster such that the positive images in the clusters after it only account to 1% of total positive images in consideration and discard those clusters. Normally, the threshold is around 5%.

After these two steps, we keep the remaining images and split them with the methods in the next two subsections.

3.4 Splitting the Difficult Set

For the remaining images in the remaining clusters, we have two observations: 1) The whole set of images is diverse. 2) Images in each cluster are quite similar. Therefore, we combine a global model to capture the general appearance of the category and a local model to classify the positive image in a fine-grained fashion.

Labels. Given the remaining clusters, we sample 400 images from each cluster (600 images if the clusters are picked by the initial clustering stage). Those images also represent a roughly uniform samples from all the images. They are used to train global and local models discussed below.

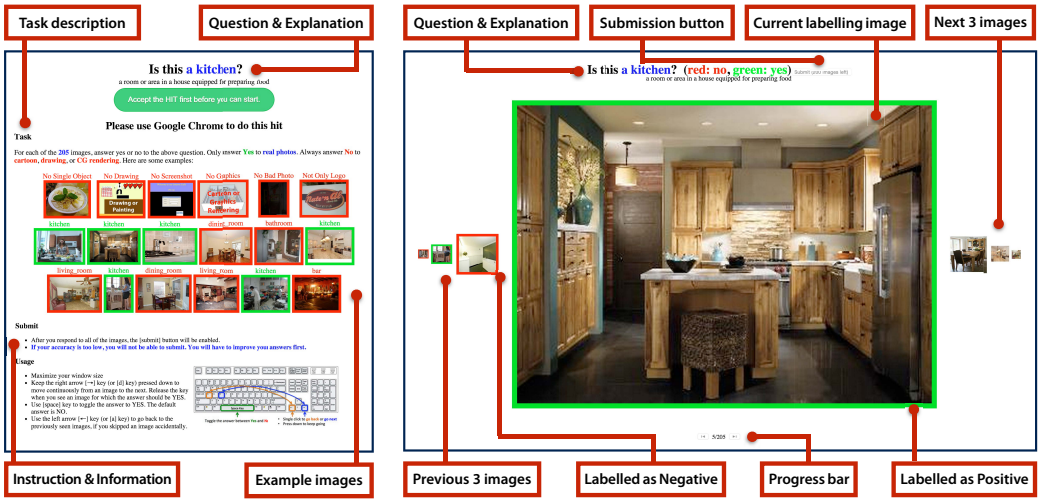
Global Model. We split all the labels into two sets: training and testing. 40% of new labels are put in the testing set. The training set is used to train a linear SVM. Then we figure out what are the score threshold for the positive and negative set.

Local Model. We also train a classification model for each cluster to adopt to the local appearance of each cluster. Because there are much fewer labeled images in each cluster compared to the whole set, we can afford to use more expensive models. To be specific, our model candidates are RBF kernel SVM and exemplar SVM. When the percentage of positive labels or negative labels are low, we use exemplar SVM. In the other cases, we use kernel SVM. We split the labels in each set into three parts: training, validation and testing. Training set is used to train the models. Validation set is used to determine the thresholds for splitting the cluster. In kernel SVM, we also use cross-validation to determine proper parameters. Testing set is used to determine the percentage of positive images in each split.

After training these models, we split the clusters for at most 5 parts. The positive parts and negative parts determined by the global models are treated as positive and negative images. For the parts split by the local models, we use the testing set to determine the positive ratio in each clusters and as in the clustering stage, we take the images in clusters with more than 95% positive images into positive set. A positive ratio threshold is determined based on 99% recall and cluster with lower positive ratio are taken into the negative set. The remaining unknown set is kept and go through this stage again until this method doesn't generate significant positive and negative images. We usually run 2 to 3 iterations at this stage before we move to the next stage, when the existing deep learning features and SVM model can no longer separate the data effectively.

3.5 Fine Tuning for Adaptation

After one to two iterations of splitting based on pertrained deep learning features and SVM models, the difficulty of further splitting the remaining unknown image set increases. Given a unknown set that is hard to separate with the pretrained features on PLACES, we fine tune the deep learning model trained by PLACES to do binary classification using the labels in the unknown set. Before each iteration of fine tuning, we sample 100,000 images and get the labels. However, we don't



(a) Instruction page (b) Labelling interface

Figure 2: Our AMT Labeling Interface.

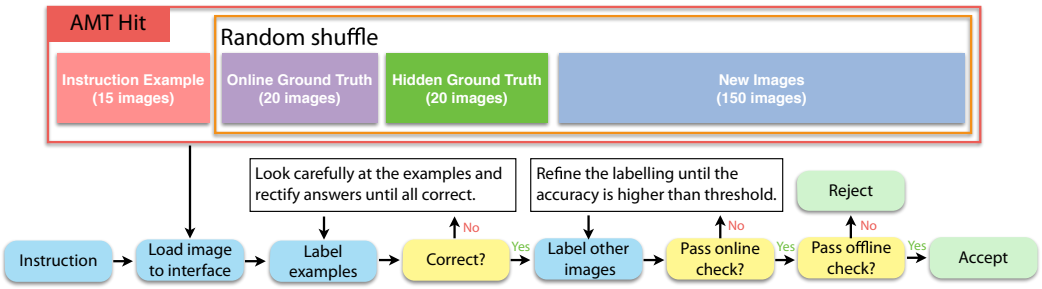


Figure 3: Life time of a HIT on AMT.

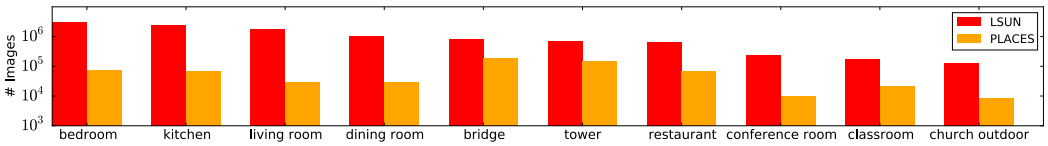
label new images at the first fine tuning, because the existing labels can train a better features than the those pretrained. We use 20,000 of the labels as testing set to determine the threshold for 95% precision and 99% recall. We repeat the same process for a few more iterations, until the number of images in the unknown set is less than 300,000. However, if the fine tuning can still separate the unknown set efficiently, we will keep splitting. In the end, we will just manually label all the unknown images.

4 High Quality Crowd-sourcing

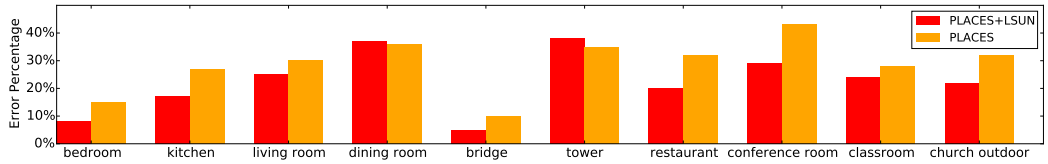
A critical part of our deep learning with humans in the loop annotation pipeline is to harvest high quality annotation from human annotators. We desire to use Amazon Mechanical Turk (AMT), because it is the largest crowd sourcing platform available so that we can collect image labels very fast at a very low cost. However, the labels obtained from AMT are usually quite noisy. Here we present a series of mechanisms and measures we design to ensure the data quality.

4.1 Interface Design

Built on top of the labeling interface from [19], our labeling interface is to let the human annotators to go through the images one by one. As shown in Figure 2, on the top of a screen, with a question like "Is this a kitchen?", a single image at a time is shown at the center, with small thumbnails for previous three images on the left and upcoming three images on the right. The worker can press the left arrow key on their keyboard to go through the images, with default answer of each image set to "No" when the image is viewed. The worker is require to press the space key on the keyboard to toggle the answer if the current answer is wrong, encoded by the color of the boundary box.



(a) Comparison of image number for each category. The higher the better.



(b) Comparison of classification error by training AlexNet. The lower the better.

Figure 4: Comparison of LSUN and PLACES. (a) For most of the categories, we are more than 10 times bigger than PLACES. Please note that the scale of y axis is log. (b) We train AlexNet with PLACES and putting PLACES and LSUN together. With addition of LSUN data, the trained model has better performance evaluated by PLACES test set.

We also observe that the scene image is normally very complicated and a bigger view of the image is very helpful for human to understand the image. So we show one image at a time and zoom the image to fill the whole window. To make sure the window can fully utilize the turker's screen, we provide a simple button and force them to open the labeling window in fullscreen mode.

4.2 Quality Control

It is a well known problem that the work quality of AMT varies a lot, and people have been proposed various ways to overcome it. We take the approach of redundant labeling and enforced instruction. For each image, we ask two people to label it and only use the doubly confirmed labels.

Given a set of images that need to be labeled, we divide them into groups of 150 images. In each AMT HIT, a human annotator is required to label a series of 205 images for \$0.1. 150 of them are the actually labels we are interested in and the other 55 are for quality control purpose. The turkers normally finish the hits within 5 minutes. We have tried more images in a HIT, but this length is the best trade-off between the length and the overhead of quality control.

Figure 3 shows the steps for the life time of a HIT. First, we show an instruction page to explain the task and questions with positive and negative examples. Although it is useful and necessary, we find that some turkers may not actually read through the instruction very carefully, especially the detailed definition and the positive and negative examples. They may also forget about the subtlty for some categories. Therefore, we set the first 15 images in each HIT to test whether the turkers understand the instruction. The images are from a set of examples representing typical scene images and common mistakes. If a turker gives a wrong label to one of the example image, a pop window will show up and block the labeling interface until the turker fixes the label to that image. After the 15 images for teaching purpose, they will go on to label without immediate feedback per image.

We embeded 40 image with known ground truth in each HIT to measure the labeling quality. 20 of them are online, meaning that their quality is checked before the HIT result is submitted to the server. If the turker can't pass the online check, they are not allowed to submit the labeling results and advised to revise their labeling results until they can pass the online check. The other 20 of those images are checked after the HIT is submitted. They are used to check whether the turker hack our Javascript code to past the online check to make sure they exhaustively look for online ground truth to pass the submission test. The labels of these images are not sent to the turker's browser, so they can't hack our interface to submit bad results. The labeling accuracy threshold for ground truth is 90% for the online and 85% for the hidden.

Together with the examples, the ground truth check is to make sure the turkers submit high quality labels. We also use them to prevent the common mistakes. We try to understand the kinds of common mistakes from the wrong labels in the testing set. When we label the images in the training set, we continue adding more images to examples and ground truth when necessary. We maintain hundreds of ground truth images and for each assignment on AMT, we dynamically sample a subset of images.

Iteration	Method	Positive images	Precision	Positive labels	Label Ratio
0	Clustering	941,981	96.9%	9,515	1%
1	ConvNet Feature + SVM	1,918,913	96.4%	41,785	2.1%
2	ConvNet Feature + SVM	2,011,332	96.2%	88,259	4.4%
3	ConvNet Fine Tuning	2,140,763	96.1%	88,259	4.1%
4	ConvNet Fine Tuning	2,215,244	95.9%	147,453	6.6%

Table 1: Algorithm statistics of kitchen. The third column shows the cumulative number of positive images in each iteration. The fifth column shows the cumulative number of positive labels provided by human. The last column shows the ratio of human labeled images compared to all the positive images we obtained. See the main text for more details.

5 Does Bigger Data Really Help?

To test our labeling pipeline and justify the need of bigger dataset for training, we have constructed an initial version of LSUN with millions of labeled images in each scene category. We experiment with popular deep nets using our dataset, and obtain a significant performance gain with the same model but trained using our bigger training set.

5.1 Evaluation and Statistics

We have released 10 categories and some categories has millions of images. Figure 4 shows the dataset statistics compared with PLACES. For most of the categories, our dataset is more than 10 times bigger. In the case of bedroom, ours is 43 times bigger.

We also investigate how well our semi-automatic pipeline can do. We want to know the dataset quality and how much human efforts we can save. The statistics of kitchen is shown in Table 1 as an example. The third column shows the cumulative number of positive images in each iteration. The fifth column shows the cumulative number of positive labels provided by human. The last column shows the ratio of human labeled images compared to all the positive images we obtained. We use the testing set to calculate the sampled positive image ratio, which is shown in the fourth column. We can see that the initial stages can give us the most images with small set of labels, while it becomes harder to get positive images in the later stage. It is because the images in the unknown set become harder to split. It is interesting to see that given the same set of labels, fine tuning can discover more positive images than SVM models. For these 10 categories, we spent around $8,000 on human labor to collect 10 million images in total. Overall, we achieve our targeting dataset quality with small portion of labels.

5.2 Training ConvNet with More Data

Given the data we have obtained, we want to know whether more data can help the current models to achieve better performance. Therefore, we use the standard AlexNet [8] and compare the model trained by PLACES [19] data with the same model trained by both PLACES and LSUN. As a simple measure to balance the dataset, we only take at most 200 thousand images from each LSUN category. Then we test the classification results on PLACES testing set. The error percentage comparison is shown in Figure 4b. The overall detection error percentage is 28.6% with only PLACES and 22.2% with both PLACES and LSUN, which yields a 22.37% improvement on testing error. Although the testing is unfavorable to LSUN due to potential dataset bias issues [16], we can see that the additional of more data can improve the detection accuracy on PLACES testing set.

6 Conclusion

We identify the lack of progress in dataset construction as a major gap in further advancing visual recognition. We propose a working pipeline to amplify the human efforts to enable them to label million-scale image collections, using deep learning with humans in the loop. We have constructed an initial version of "LSUN" database, a scene-centric database with millions of labeled images in each scene category. Simple experiments on this dataset have already demonstrated the great potential of bigger data for visual recognition. We will continue to contruct this large-scale database and hopefully it will be able to serve the community to continue making amazing progress.

Acknowledgment. This work is partially funded by Intel, Google, MERL, Princeton Project X, and a hardware donation from NVIDIA.

References

[1] http://www.image-net.org/challenges/LSVRC/announcement-June-2-2015.

[2] B. Collins, J. Deng, K. Li, and L. Fei-Fei. Towards scalable dataset construction: An active learning approach. In ECCV, pages 86–98. Springer, 2008.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In CVPR, pages 248–255. IEEE, 2009.

[4] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. CVIU, 106(1):59–70, 2007.

[5] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572, 2014.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. arXiv preprint arXiv:1502.01852, 2015.

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[8] 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.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998.

[10] A. Nguyen, J. Yosinski, and J. Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. arXiv preprint arXiv:1412.1897, 2014.

[11] B. Settles. Active learning literature survey. University of Wisconsin, Madison, 52(55-66):11, 2010.

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

[13] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. arXiv preprint arXiv:1409.4842, 2014.

[14] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199, 2013.

[15] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. The Journal of Machine Learning Research, 2:45–66, 2002.

[16] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In CVPR, 2011.

[17] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun. Deep image: Scaling up image recognition. arXiv preprint arXiv:1501.02876, 2015.

[18] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In CVPR, pages 3485–3492. IEEE, 2010.

[19] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In NIPS, 2014.