Resilience to Data Scarcity in the U-Net Architecture

Benjamin Killeen

***Abstract*—Deep convolutional neural networks (DCNNs) underlie myriad recent successes in computer vision tasks, but these successes also rely on very large, labeled datasets for training. In many cases, especially scientific tasks, this volume is untenable due to the cost of labeling. In this paper, we examine the U-Net architecture's resilience to data scarcity as well as the effect of various data augmentation schemes on its performance.**[1]

I. Introduction

Large, labeled datasets like ImageNet [1] have enabled DCNNs to succeed at increasingly advanced vision tasks. In 2012, Krizhevsky et al. [2] coupled the volume of ImageNet with data augmentation to achieve previously unseen accuracies in image classification. When done properly, data augmentation effectively increases the size of the dataset by transforming existing examples. However, it also has the potential to cause overfitting, since no genuinely new examples are added. In cases where the original dataset already exhibits marked scarcity, this risk is amplified.

Although image classification is an impressive feat, many tasks demand higher spatial resolution. Semantic segmentation consists of per-pixel classification, a level of detail that greatly increases the cost of labeling. Although certain applications justify the investment, and datasets like Microsoft's COCO [3] continue to facilitate supervised training, many use-cases which would benefit from DCNNs exhibit marked data scarcity. In particular, Ronneberger et al. [4] considers applications of semantic segmentation in biomedical images, where the specialized nature of each experiment forbids large dataset generation, but the localization of individual structures, *e.g.* neurons, in electron microscope stacks would be invaluable. In general, scientific tasks often exhibit this data scarcity. Ronneberger et al. addresses this problem by introducing the U-Net architecture, which confronts data scarcity through a combination of data augmentation and network design.

In this paper, we evaluate the performance of U-Net on the Oxford-IIIT Pet dataset [5] under increasing levels of data scarcity. We also examine the effect two data augmentation methods on each dataset: horizontal flipping and random thumbnail extraction. We evaluate the classification accuracy of U-Net on a test set consisting of 222 examples, withheld during training.

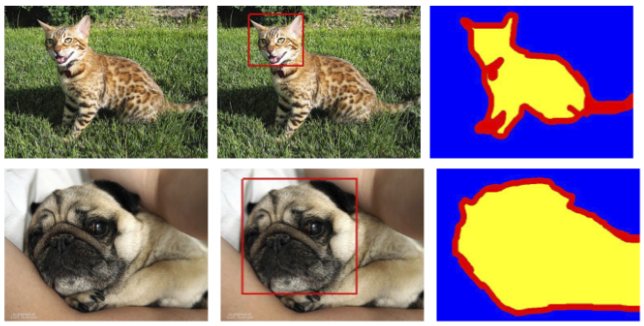

Fig. 1. Examples from the original Oxford-IIIT Pet Dataset [5]. Original images (left) are annotated with bounding boxes (middle) and segmentation trimaps (right).

II. Method

The Oxford-IIIT Pet [5] dataset consists of 7349 images of individual cats and dogs. There are 37 different breeds represented, with approximately 200 examples from each breed. For simplicity, we limited our classification to three classes: cat, dog, and background, but in every dataset, we retained as much uniformity as possible across breeds. The testing set, for example, consists of $n_{\text{test}} = 6$ examples from each breed. We also resized each image to 512×512, for uniformity. As shown in Fig. 1, the Pet dataset includes bounding boxes and semantic segmentation trimaps. To recover labels, we remap the boundary region of each trimap to the animal's classification, and resize the resulting annotation to 512×512, correcting misclassification from interpolation as necessary.

In order to observe the effect of increasing data scarcity S, we generate four datasets with $S = 0, 0.3, 0.6$, and 0.9. S determines the number of examples drawn from each breed $N_{b,S}$ according to

$$N_{b,S} = \lfloor (1 - S) \cdot (N_b - n_{\text{test}}) \rfloor,$$

where N_b is the number of examples from breed b in the original dataset. Table I shows the resulting number of examples in each generated dataset, for different scarcity levels.

In addition to examining increasing scarcity, we examine the effects of two augmentation schemes. The first augmentation, "flip," consists of flipping the image horizontally. Applying the same transformation to the annotation results in a "new" labeled example. Applying just this transformation doubles the

Benjamin Killeen is an undergraduate with the Department of Computer Science, University of Chicago, Chicago, IL 60637, USA (email: killeen@uchicago.edu)

[1]Code available at github.com/bendkill/artifice. See Appendix A for details.

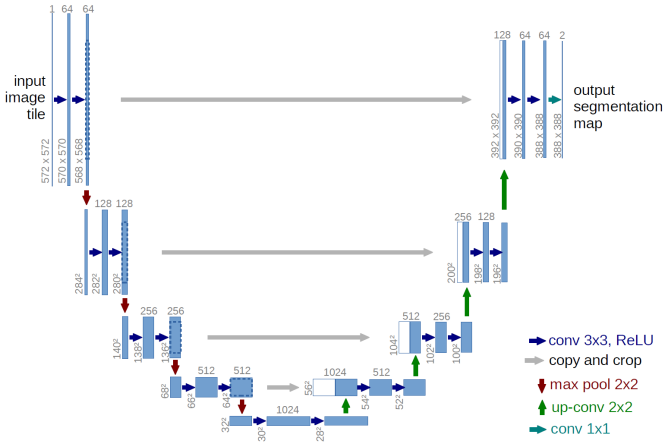

Fig. 2. The U-Net architecture [4]. Blue boxes correspond to muli-channel feature maps. White boxes represent copied feature maps. Arrows represent various operations.

size of the dataset. The second augmentation, "thumbnail," extracts a random 384×384 subsection from the original image and resizes it to 512×512 (extracting the same subsection from the annotation). We apply the "thumbnail" augmentation with the "flip" augmentation additively, tripling the size of the original dataset. Table I shows the number of examples in each augmented dataset, at each scarcity level.

A. Model

Fig. 2 describes the U-Net architecture, which uses a series of convolutions and "de-convolutions" or "up-convolutions" to build up abstract feature maps at multiple levels while maintaining high spatial resolution for the final segmentation. We implement this architecture using Tensorflow [6], and for each experiment limit training to one epoch.[2] With a batch size of four, we utilize Adam optimization [7] over pixel-wise cross-entropy between predicted and ground-truth annotations.

III. Results

We evaluate the performance of U-Net in terms of the average test-accuracy across all pixels. Table I shows the results of training for one epoch on each dataset, measuring the performance on a separate test set withheld during training. Firstly, we note the marked decline in accuracy as scarcity increases (with no augmentations). Training with $S = 0$ yields an accuracy of 73.6% (compare to random classification 33%). We believe that further epochs would significantly improve this value. The smallest dataset, by contrast, yielded an accuracy of only 58.9%.

As can be seen, the effects of augmentation are most significant for datasets with high scarcity. The $S = 0.9$ set saw a 7.3% increase in accuracy after applying both augmentations. The $S = 0$ dataset, by contrast, only experienced a 3.1% increase with the "flip" augmentation, and applying both augmentations actually caused some overfitting, decreasing the accuracy compared to just "flip."

[2]Results from time constraints. In future experiments, we aim to evaluate U-Net's performance over many epochs.

Scarcity	Augmentations	Examples	Test Accuracy (%)
0	None	7128	73.6
0	flip	14256	76.7
0	flip, thumbnail	21384	75.2
0.3	None	4960	73.1
0.3	flip	9916	75.1
0.3	flip, thumbnail	14876	75.7
0.6	None	2824	70.4
0.6	flip	5648	73.8
0.6	flip, thumbnail	8472	74.4
0.9	None	664	58.9
0.9	flip	1324	63.6
0.9	flip, thumbnail	1988	66.2

TABLE I

IV. Discussion

We have shown the effects of increasing data scarcity on the U-Net architecture, as well as the improvements that can be made through simple data augmentation. Although these results are promising, we recognize that much work remains to be done. For instance, we have made the choice (due to time constraints) to limit training to one epoch. While practical, this decision results in a fundamental ambiguity between the effects of data scarcity, where the number unique examples decreases, and limited training. In order to more properly measure the effects of data scarcity, it seems, U-Net should undergo a constant number of training steps, preferably for many epochs, independent of the number of unique examples.

In addition to this shortcoming, we were disappointed to not be able to explore further data augmentation methods, as well as various combinations of each of these augmentations. In particular, we wished to test the effects of object extraction, which would use the known segmentation to extract a cat or dog from the image, translate it within the image, and reinsert it. More than any other data augmentation method, we believe this "extract and shift" scheme has the potential to explore new regions of the data space. That is, it could create the most novel examples for the network to learn from.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," p. 8.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[3] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollr, "Microsoft COCO: Common Objects in Context," *arXiv:1405.0312 [cs]*, May 2014, arXiv: 1405.0312. [Online]. Available: http://arxiv.org/abs/1405.0312

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv:1505.04597 [cs]*, May 2015, arXiv: 1505.04597. [Online]. Available: http://arxiv.org/abs/1505.04597

[5] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and Dogs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker,

V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," p. 19.

[7] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980 [cs]*, Dec. 2014, arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980

APPENDIX A
IMPLEMENTATION INSTRUCTIONS

Because the implementation underlying this report is part of a larger project, we provide instructions here for implementation validation on the University of Chicago `midway` cluster. To set up:

1) `ssh` into `midway2.rcc.uchicago.edu`.
2) Clone the repository at github.com/bendkill/artifice, and change into its root directory.
3) Run
   ```
   export PYTHONPATH=$PYTHONPATH:`pwd`
   ```
4) Download the Oxford-IIIT Pet dataset from here, and untar both annotations and images in `data/pets`.
5) Load the compute node by running
   ```
   sinteractive --partition=gpu2 \
     --gres=gpu:1 --mem=32000
   ```
6) Load the environment by running
   ```
   module load cuda/9.0
   module load Anaconda3/5.0.1
   source activate DL_GPU_cuda_9.0
   ```
7) Run
   ```
   python test_utils/pets.py -h
   ```
   to see test installation and see options. It is normal for this to take a while the first time, as tensorflow sets up its environment. Some warnings are also normal.

You can also run premade batch scripts for training in the `batch` directory. These may have to be modified with the proper account.