10707 Project Report
How Do Neural Networks Learn Visual Counting

Boyu Liu
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213
boyuliu@andrew.cmu.edu

Rui Yan
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
ruiyan@andrew.cmu.edu

Da Yin
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
dayin@andrew.cmu.edu

1 Introduction

Visual counting is a problem in computer vision, which requires accurately counting object instances in scenarios. Compared to other popular visual recognition tasks, like object classification and object detection, visual counting is an area of less interest. This may due to the weaker importance of counting, or the family of methods which view counting as a sub-problem of object detection. Surely, if we have a good object detector, counting is a trivial problem. However, visual counting has a substantial advantage over object detection and even object classification: **Data collection and annotation**.

Compared to object detection, object counting does not required human-labeled accurate tight bounding boxes around the objects. object count is a image-level annotation, which is much cheaper than instance-level annotations, like object detection bounding boxes. Precise localization is quite labor-intensive work, and C-WSL[1] claimed that bounding-box annotation is $38\times$ more time-consuming than counting the number of objects. And the time consumption has only small overhead over image-level classification.

Compared to object classification, although the annotation of object counting requires a little bit more time, the collection of data is much easier. Current image-level object classification largely focus on single object classification, which required a clean image containing single object. However, natural image always contains multiple objects. Manually cropping out the object also requires considerable amount of time. However, object counting instead could take natural multi-object images as valid data.

Thus we come to the main idea of this paper: *Could we use object counting as a **weak supervision** to other visual tasks? Or could we **transfer the intermediate result** (like feature maps or heatmaps) of object counting to other visual tasks?*

With the rapid development of deep learning, recent researchers have been applying deep convolution networks or other architectures to handle this problem. A two stages process is widely used in computer vision. The first stage is to extract image features using different filters which are expected to capture image latent structures. The second stage is to use these features to get the output such as classification label or regression counting number. Concerning about visual counting problem, those labeled locations could be used as supervised label in first stage. Compared to end-to-end training,

10707 Class Project (2019 Spring).

this supervision information is much more informative than one counting result number and is an efficient way to use human prior knowledge to overcome the lack of labeled data.

Our intuition, on the contrary, is to see whether neural networks can learn these extra provided supervision information automatically. In other word, if we only provide the model with the number of object instances, could this model extract useful hidden features? The most straight forward experiment we could do is to go back to the hidden outputs inside neural networks after first stage and see whether it really captures the target object.

We conducted a series of experiments on several different datasets and the results show that with the supervision of object instance count, given enough data, deep convolution neural network could still learn object locations. On top of that, if the training samples contain objects from different categories, the model can even learn the category as well. In addition, the learned heat map could be used for extracting bounding box approximately. While labeling of bounding box is comparatively more expensive than that of counting, our experiment result provides an alternative way to retrieve object locations.

The rest of this paper has the following structure: **Section 2** introduce some related work in visual counting and weakly supervised visual tasks. **Section 3** elaborate the dataset we constructed and used for our task. **Section 4** depicts our methods and networks for visual counting, and how to transfer to other tasks. **Section 5** showed the experiment results. **Section 6** discussed over the results. **Section 7** gives a conclusion and view of future work.

2 Related Work

Visual Counting: There are several previous work relate to visual counting. Unsupervised learning methods including detection based and regression based suffers from low accuracy and are sensitive to variants of structures such as overlapping and non-uniform distributions. [2] proposed supervised learning method to solve visual counting. They use MESA metric as part of loss function and apply quadratic optimization to solve parameters. Their experiments on *Bacterial cells in fluorescence-light microscopy image* dataset [3] and *Pedestrians in surveillance video* dataset proves that supervised learning method can greatly outperform unsupervised learning methods. [4] apply Convolutional Neural Network on counting problems. They treat this counting task's first stage as density estimation and proposed two architecture to solve it: counting CNN and Hydra CNN. Their experiments on TRANCOS dataset and UCSD crowd dataset demonstrate the power of Hydra CNN and perform better than previous methods. [5] try to solve cross-domain object counting problem. Prior researchers mostly focus on single category object counting while in this paper, they use adapter module based on [6]'s work to extend the functionality of neural network into different domains without forgetting previous domain data.

Weakly Supervised Visual Tasks: [7] propose to use counting information to improve weakly supervised localization. With weakly supervised model, they first extract some bounding boxes and use counting information to determine which of them are objects. Their method does not solve visual counting problem but the supervision information of counting can help weakly supervised model find high-quality regions. [8] propose to use image-level classification for localization tasks. In this work, only a classification annotation is used for each layer. Similar to our work discussed later, the heatmap is used to give an approximation of the object inside the image. However, this work mainly focused on few objects in an image. And our work use more informative supervision counting instead of image-level class label, with little extra labor. [9] also proposed to use image-level class supervision for detection. However, this work use region proposal algorithms to generate proposals, which is either complex heuristic(like selective-search) or pre-trained model. Instead, our work could learn from scratch without complex human design.

3 Dataset

The counting task is much less informative than object detection and classification information so our configuration requires more data for model to converge. While current visual counting datasets are relatively small, in our experiment, we firstly build our own dataset which allows us to use any many samples as possible. Then we applied real world dataset. Here is a detailed introduction to datasets we use.

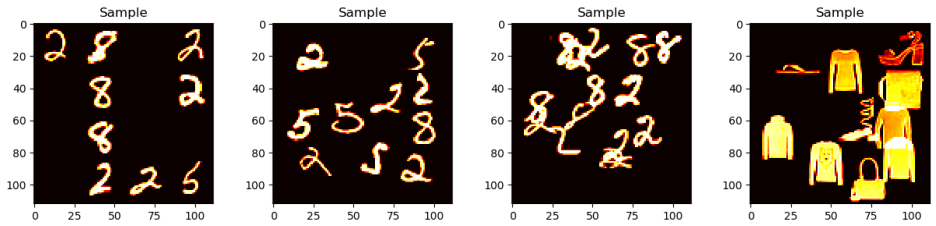

Figure 1: Left: A generated image from MNIST handwritten digits without random placing and overlap. Target object is number 8. Label of this image is 3. Grid size is 4. Mid-left, Mid-right: A generated image from MNIST handwritten digits with random placing and overlap. Overlap rate is 0.3 (mid-left), 0.8 (mid-right). Target object is number 2,5,8. Right: A generated image from fashion MNIST with random placing and overlap. Overlap rate is 0.3. Target object is *Pullover*, *Sandal*, *Bag*.

MNIST based generated dataset Our generated dataset is based on MNIST handwritten digits dataset [10]. The generation method in our first attempt is to fill a $G \times G$ image with some MNIST image. For example, in left of Figure 1, we place MNIST digits in a 4×4 grid image. Each digit is placed in one grid and some grids can be empty. This is the dataset we used in midway report. There are some improvements based on that.

1. In our previous experiment, we only have one regression target. That means each model is only trained to predict the number of one digit. The first improvement is to extend our training object to multiple categories. The does not change too much to our generated samples except for the additional labels of other classes.
2. The second improvement is to make digit placing random and enable overlapping. In Figure 1, the digits are placing randomly which can cross grid boundary. We also make the overlapping possible. Here we use overlap rate 0.3 in default. The right side of Figure 1 is initialized with overlap rate 0.8 which is even hard for human beings to count.
3. The third improvement is to use fashion MNIST [11] to replace digit MNIST. Fashion MNIST has clothes and bags as samples and 10 different classes in total. Figure 1 shows an example of generated fashion MNIST sample.

Our generated datasets can be arbitrary large. Generally, larger categories and denser digits will make training harder. In return, we will use larger model and generate more data in train dataset to make model converge.

TRANCOS dataset TRANCOS dataset [12] is a vehicle counting dataset including 823 train samples and 421 test samples. The original dataset includes specific locations of vehicles and each sample is 640×480 pixels. This real world dataset has much larger input images and sparser object distribution compared to the MNIST generated dataset. A sample of TRANCOS is shown in 10.

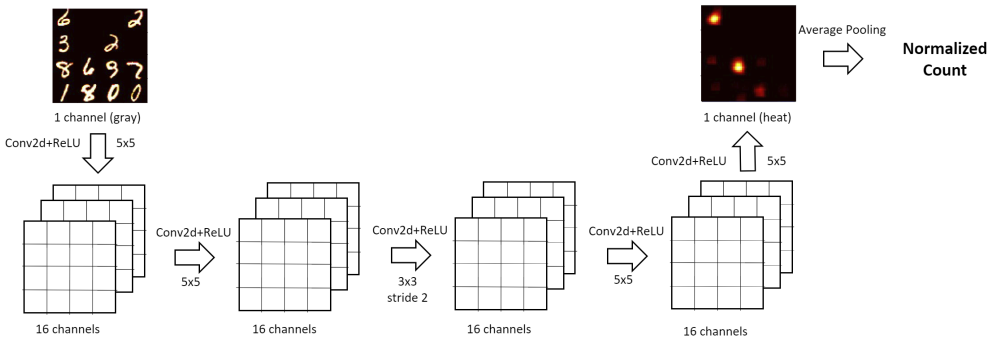

Figure 2: Neural Network Architecture

4 Methods

4.1 Network Structure

In order to see if neural networks really learn latent structure of data during learning counting, we need to construct networks and train on some dataset with counting tasks. However, current available datasets are relatively small compared to the whole input space so it is hard for models to train directly on the final counting number. So in our experiments, we construct some dataset to train models and observe their hidden state.

We expect our hidden output to be some feature maps like heat maps or density estimation maps. The final counting output should be proportion to the heat value of the whole map. So our model's second stage's input is a $W' \times H'$ image and there is only one average pooling layer in second stage to take average of the hidden output from first stage.

For first stage, we use 5 convolution layers with ReLU. The middle layer is using stride 2 and kernel size 3 which is a variant pooling layer. The other convolution layers are using kernel size 5 and padding 2 which keep output shape same with input shape. Those convolution layers have a span of 28 to combine information. We expected the hidden output of this stage can present some structure information of the input image which can be used for counting. This model configuration is applied to all our MNIST based datasets. The architecture of this model is shown in Figure 2.

The model is designed in a standard way. The reason for that is if the most simple convolution network can actually learn the hidden heat map laid behind, more complex model will also have the capability to do so.

4.2 Training Counting Network

We feed N randomly generated samples to train our model and use 1000 other randomly generated samples to validate our model. Since training set and validation set do not have distribution bias so the over-fitting problem is not a critical problem now. The training dataset is enlarged with the increase of task difficulties. With randomized placement and non zero overlap rate, the counting task is more difficult. The random placing and overlapping is used to make our generated dataset more like real world dataset. Also, if we increase the class number of counting targets, the model need to learn more. So in these cases, we increase the filter size in convolution layers and feed more training data for model to learn. For simplicity, our current model do not deal with multi-scale object yet.

Notice that we do not choose too large overlap rate. As shown in Figure 1, a sample with large learning rate is even hard for human beings to recognize. In real world dataset, such as cell dataset or vehicle dataset, objects could be really close or even connected. But overlapping with a large proportion of objects is infrequent. So the overlap rate we use here is defined as the proportion of objects overlapping area and we set it to 0.3 as default.

Choosing L1 Smooth Loss as our training loss function is because we found mean square error will somehow make the training process unstable. We speculate that it is because for large counting error, it will update the model parameters with large weight which is too aggressive. But l1 loss itself for small counting error, it will keep the same update rate with larger cases which will make small regression error equally important as large error. L1 smooth loss can balance these two. The formula is as below

$$\mathcal{L} = \frac{1}{n} \sum_i z_i$$

$$z_i = \begin{cases} 0.5(y_{true} - y_{pred})^2, & |y_{true} - y_{pred}| < 1 \\ |y_{true} - y_{pred}| - 0.5, & otherwise \end{cases}$$

The optimizer we used was Stochastic Gradient Decent, with batch size 64, learning rate 1e-4 and decay of 0.01 every 10 epoch, and the network is trained for 30 epochs.

4.3 Transferring to Other tasks

Here we mainly use the intermediate heatmap (just before last average pooling) for other tasks.

Table 1: The experiment results: CLS represents the number of categories. Rand represents whether the items are randomly placed. Overlap represents the overlap rate. Grid is the grid size. MaxN is the max number of one category instances in one sample. Filters is the size of convolution layer filters. Epoch is the training epoch. Loss is the training loss (We use L1 smooth loss as default.) AE is the absolute error accumulated over different categories which is our evaluation metric.

Dataset[Size]	CLS	Rand	Overlap	Grid	MaxN	Filters	Epoch	Loss	AE
Digit[10000]	1	False	0.0	4	16	16	25	0.204	0.512
Digit[10000]	10	False	0.0	4	16	16	25	0.183	3.467
Digit[30000]	10	False	0.0	4	16	16	30	0.091	3.131
Digit[50000]	10	False	0.0	4	16	64	30	0.050	2.189
Digit[10000]	3	True	0.3	4	9	16	30	0.336	2.070
Digit[30000]	3	True	0.3	4	9	16	30	0.064	0.844
Digit[50000]	3	True	0.3	4	9	64	30	0.032	0.582
Fashion[30000]	10	False	0.0	4	16	32	30	0.246	5.374
Fashion[30000]	3	True	0.3	4	9	32	30	0.064	0.843

4.3.1 To Classification

This problem is actually a subset of object counting. we could just sum over the heatmaps for each class, and judge whether the result is larger than 0.5 or not. (0.5 is a soft threshold, as the count for positive data should be larger or equal to 1).

4.4 To Detection

As shown later in the results part, we could observe that the heatmap provide a quite strong localization information. Here we design a heuristic based method to extract bounding boxes from heatmaps. (1) we set a threshold for the heatmap to clean the noise and generate a binary image. (2) we use connected component algorithm to get the component for each object. (3) we use the heatmap value of each component as weight, and calculate a weighted sum of coordinate of each pixel in the component, to get the center of the object. (4) we use the prior of the object size to draw the bounding box.

5 Experiment Result

Our model's performance is shown in Table 1. We can see that the larger dataset and the larger filter size we use the better results we could get.

Note that the AE (Absolute Error) is accumulated over categories. The reason for not doing average on this metric is because the total number of object instances is not linearly to the number of categories. For example, in 10 categories experiments, we still have a maximum number of 16 objects disregard of their labels and in experiments containing 3 categories with random placement and overlapping, for each category object, we have at most 3 objects so in total there is 9 objects for maximum.

We also use heuristic method to extract object bounding box from retrieved heat map. In Figure 3, we show one sample of extracting bounding box from heat map. We can see that our heat map is pretty useful for extracting bounding box while there is still some problems with the extraction. For example, in Figure 3, the heat of two number 4 is connected, so the extracted bounding box is at the center of these two digit. This problem could be alleviated given the total number of target object instances and there are some paper dealing problems related to that. Due to time limit, we do not implement further.

6 Discussion and Analysis

We will analyze our experiments from several comparisons of model.

Single Category vs. Multiple Categories If we do not increase the filter size of convolution layers, the increase of categories to be counted will make the retrieved heat map focus more on the

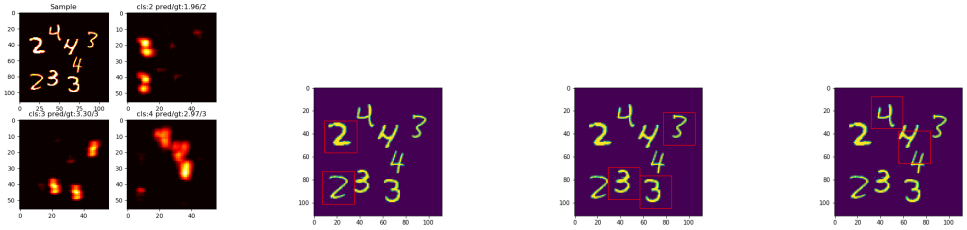

Figure 3: The extracted bounding box from heat map. The location and size of bounding box is mapped back to original image.

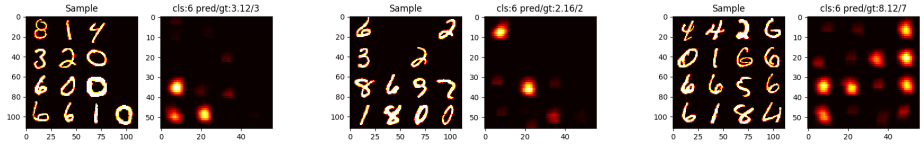

Figure 4: The heat map of our baseline model targeting at counting number 6.

center of the object. This can be shown in comparison between Figure 4 and 5. Training on single class can have a larger activation region while on multiple categories this region tends to be much smaller. The reason is probably because in multiple categories cases, the model need to learn the differences between categories more accurately. For each point in the retrieved heat map, it is actually a convolution result of a 28×28 region in original image. So the center of each digit is directly convoluted from that digit while surrounding points are more or less effected by surrounding digits. The farther the point locates from center, the more it will be effected from other digits. Notice that we do not use a softmax layer since it is not necessary to assign a digit label to each pixel in heat map. So to make clear distinction between digits, the model tends to make decision only depending on the center of digits in heat map which it is rather confident with and leave surroundings make no contribution to any digit. Comparing the left image and right image in Figure 5 we could see that even we triple the size of feeding dataset, the retrieved heat map still presents to have the same feature (small activation region).

Fixed Position vs. Random Placement + Overlapping From Table 1, we see that without enlargement of training dataset, the performance is really bad so we only show the results of model trained on 30000 data in Figure 7. Here we use 3 different categories but with no other category interference. It turns out that the model can still learn the heat map correctly but the learned result is not as sharp as what we have in Figure 4. The heat map is more blurred and there are lots of noise with relatively low heat. This phenomenon could be explained by the kernel size we use. Our designed

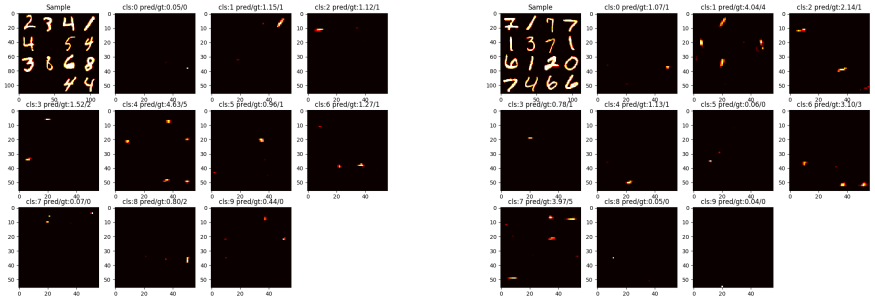

Figure 5: Model trained with 16 filters and 10000 (left) / 30000 (right) data w/o random placement or overlapping targeting at all 10 numbers.

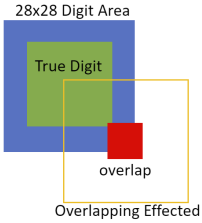

Figure 6: By spreading out activation region, the model could alleviate the effect brought by overlapping area.

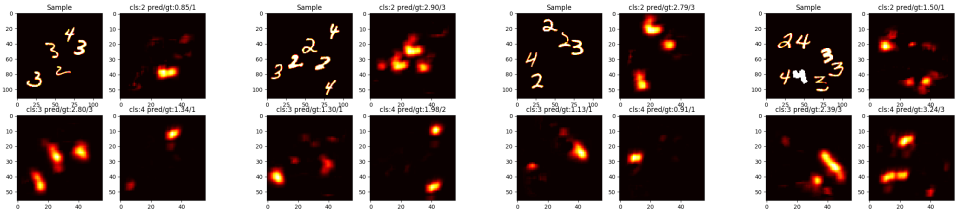

Figure 7: Model trained with 16 filters and 30000 data w/ random placement and overlapping targeting at number 2,3,4.

multi-layer convolution network is able to capture information in 28×28 size region. However, even if the model could learn the kernel of each digit correctly, the overlapping part in the region will also alter the convolution results of original digit. Thus, instead of making activation region smaller which is shown in previous analysis, in this configuration, model will make activation region spread out to alleviate the influence of overlapping area. In Figure 6, the blue box is the 28×28 original digit. It is overlapped by red box, part of another digit. The green box is the core part of original digit. By using our kernel design, each pixel in heat map will involve a 28×28 region in original image, and the orange box is the area of heat value affected by overlapping area. The final target of this model is to make the summation of values in blue box equal to 1 approximately. If the heat map is concentrated in the center of blue box, it will be directly affected by overlapping part. But if the heat map is spread out over green box, we could see that only about $1/4$ part of the heat map is affected by overlapping area so the average pooling will make side-effect much less than the center point.

Handwritten Digit MNIST vs. Fashion MNIST The fashion MNIST is a bit harder than digit MNIST. Even we use 32 filters, the regression results on 10 categories counting is poorer than that of digit MNIST using 16 filters. In Figure 8, we can see that the model could do counting on objects with clear differences such as bags, shoes, trousers. But for shirt (label 2), coat (label 4) and pullover (label 6) which are really similar, the model has some difficulties to distinguish them. In the right sample of Figure 8, the heat map of category 2, 4 and 6 are close.

If we pick out *Dress* (label 3), *Sandal* (label 5), *Bag* (label 8) to train, as shown in Table 1, the performance is pretty close to what we have in MNIST although we doubled the filter size. In Figure 9, we could see that the learned heat map is similar to Figure 7. All heat map we present are normalized with themselves. So in third sample of Figure 9, the heat map for category 3 is actually saying those three black area are even less likely to be *Dress* than empty noise.

We also try to apply this data to TRANCOS dataset. However, the performance is poor and the model cannot learn anything only from the counting number of vehicles. Compared to our previous experiments, this real world dataset is much larger and target object instances only take up a tiny proportion of the whole picture. The size of dataset is also too small. While previous datasets we built usually have over 10000 training data, this real world dataset only have around 800 samples to learn. As a result, the model is easily trapped by local optimum which is the mean of training dataset's counting objective distribution.

However, we do not fully utilize the existing real world dataset. Counting datasets like TRANCOS does have precise location information although most of them lacks object bounding box. We could

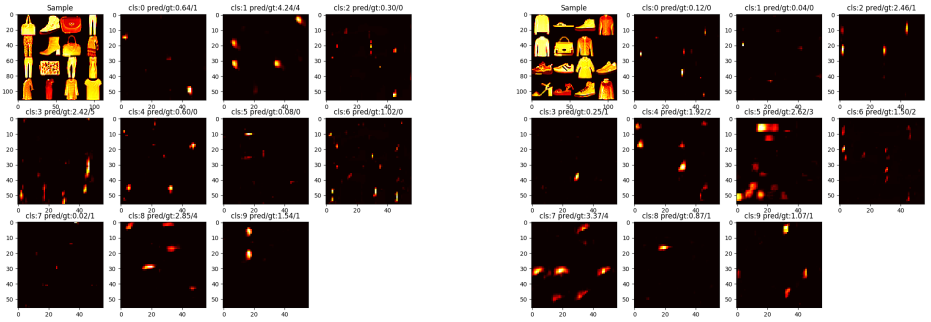

Figure 8: Fashion model trained with 32 filters and 30000 data w/o random replacement and overlapping targeting at 10 categories. Ten categories are *T-shirt*, *Trousers*, *Pullover*, *Dress*, *Coat*, *Sandal*, *Shirt*, *Sneaker*, *Bag*, *Ankle Boot*.

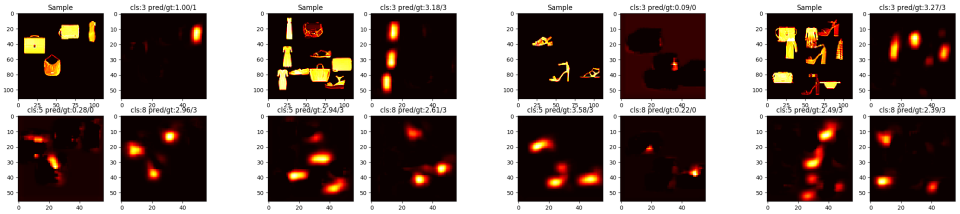

Figure 9: Fashion model trained with 32 filters and 30000 data w/ random replacement and overlapping targeting at 3 categories including *Dress*, *Sandal*, *Bag*

do sub-sampling to boost the number of available training data which can definitely benefits our model since by using average pooling we do not necessarily require network input samples having same size.

7 Conclusion

In this work, we propose to use counting information as supervision information to train traditional multi-layer convolution neural network. We found that by using average pooling layer, we could retrieve object heat map from this network. Our constructed MNIST based datasets show that this method is effective under different scenarios including multi-class, random placement, overlapping and different types of objects. We also implemented a simple heuristic algorithm to extract bounding box from heat map.

There is still more work to do on this problem. Real world counting datasets usually contain much less samples compared to the training dataset size we use in this experiment. Data augmentation could be applied to help solve this problem.

In conclusion, our work reveals that neural network can automatically learn object positions and how they look like in the learning process of counting which implies neural nets are not completely black boxes. Instead, their learning does have something in common with that of human beings.

References

[1] Mingfei Gao, Ang Li, Ruichi Yu, Vlad I Morariu, and Larry S Davis. C-wsl: Count-guided weakly supervised localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 152–168, 2018.

[2] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In *Advances in neural information processing systems*, pages 1324–1332, 2010.

[3] DE Caldwell, DR Korber, and JR Lawrence. Imaging of bacterial cells by fluorescence exclusion using scanning confocal laser microscopy. *Journal of microbiological methods*, 15(4):249–261, 1992.

[4] Daniel Onoro-Rubio and Roberto J López-Sastre. Towards perspective-free object counting with deep learning. In *European Conference on Computer Vision*, pages 615–629. Springer, 2016.

[5] Mark Marsden, Kevin McGuinness, Suzanne Little, Ciara E Keogh, and Noel E O'Connor. People, penguins and petri dishes: Adapting object counting models to new visual domains and object types without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8070–8079, 2018.

[6] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017.

[7] Mingfei Gao, Ang Li, Ruichi Yu, Vlad I Morariu, and Larry S Davis. C-wsl: Count-guided weakly supervised localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 152–168, 2018.

[8] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[9] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2846–2854, 2016.

[10] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[11] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[12] Roberto López-Sastre Saturnino Maldonado Bascón Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez and Daniel Oñoro-Rubio. Extremely overlapping vehicle counting. In *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2015.

Appendix

Figure 10: A sample of TRANCOS dataset.

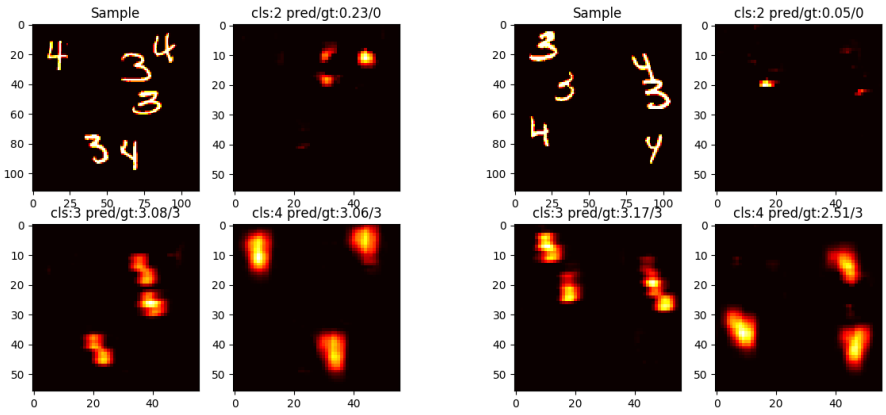

Figure 11: Model trained with 64 filters and 50000 data w/ random placement or overlapping targeting at number 2,3,4.

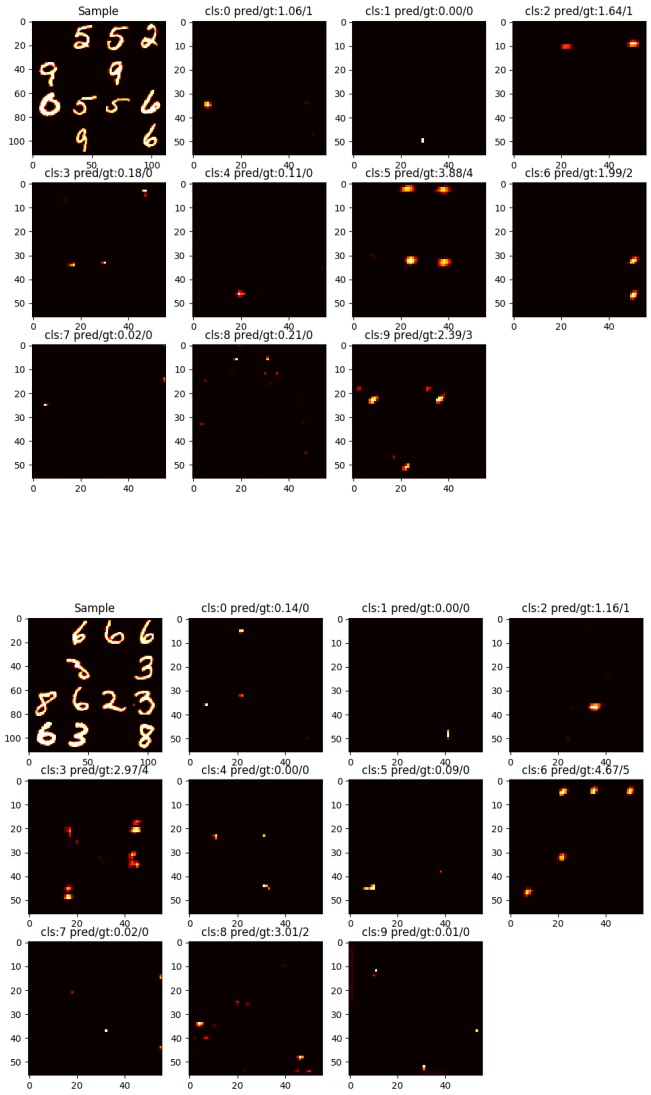

Figure 12: Model trained with 64 filters and 50000 data w/o random placement or overlapping targeting at all 10 numbers