Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

HistoNet: Image-based prediction of count and size distribution histogram of object instances

Kishan Sharma

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

HistoNet: Image-based prediction of count and size distribution histogram of object instances

Author:	Kishan Sharma
1st examiner:	Prof. Dr. Laura Leal-Taixe, TUM
Primary advisor:	Dr. Jan Dirk Wegner, ETH Zurich
Submission Date:	June 24th, 2019

Acknowledgments

I would like to express my gratitude to all the people who supported me in the successful completion of this Master thesis. I would like to thank, Dr. Jan Dirk Wegner for providing me with the opportunity and support to pursue this Master thesis at ETH Zurich. I am grateful to Prof. Dr. Laura Leal Taixe for allowing me to undertake this thesis under her supervision. Their vast experience and knowledge always steered me in the right direction. Their constant support and guidance have been pivotal in the pursuit of achieving the desired results. I would also like to thank, Dr. Christian Zurbrgg and Moritz Gold for funding this thesis as well as for providing infrastructure for the data acquisition. Last but foremost, I would like to wholeheartedly thank my family and friends for their unwavering support and encouragement.

I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail.

- Abraham Maslow

Abstract

Pixel-wise segmentation and instance segmentation of objects are the core research topics in computer vision, which are essential for scene understanding. Solving this non-trivial problem for less complex tasks such as object counting and size estimation requires training of large models and complex training pipelines. For those tasks, the size of instance segmentation models is disproportionate with respect to the complexity of the problem. In this work, we propose to predict statistical summary of objects in the form of object count and object size histogram in crowded scenes directly without any explicit object instance segmentation. What makes this task challenging is the high density of objects (of the same category), which makes instance identification hard. Instead of explicitly segmenting object instances, we show that directly learning object count and histogram of object sizes improves accuracy while using drastically less parameters. For this, we introduce a novel deep learning architecture *HistoNet*. This is very useful for application scenarios where explicit, pixel-accurate instance segmentation is not needed, but there lies interest in the overall distribution of instance sizes and object count. We show the applications of our method in biology, where we estimate the count and size distribution of soldier fly larvae, and in medicine, where we estimate the count and size distribution of cancer cells as an intermediate step to calculate tumor cellularity score. We also provide a new data set for this task, the *Fly Larvae* dataset, which consists of 11,000 larvae instances labeled pixel-wise. Additionally, we propose an extension of our deep learning architecture called *HistoNet-DSN* in which deep supervision at hidden layer is incorporated for the refinement of our method. We simulate crowded scenarios with large variance in object sizes, by generating synthetic ellipse dataset and thus verifying the robustness of our method. Finally we show that our method results in an overall improvement in the count and size distribution prediction as compared to state-of-the-art instance segmentation method Mask-RCNN [18] and is applicable to different image modality and application domain.

Contents

1. Introduction

1.1. Motivation

To mimic the most powerful human sense, the human vision and to develop algorithms that can be run in such a fast and accurate way is still a vast topic of research in computer vision. A variety of vision-based task such as pixel-wise segmentation of objects (e.g., [46, 17, 42, 53]) and instance segmentation (e.g., [4, 18]) are core research topics in computer vision and has seen a rapid improvement in over a short period of time. While these solutions are directly applicable in various scenarios, for many problems, these are often only an intermediate step which is followed by further post-processing like shape refinement and vectorization [23, 44, 6] that generate a product amenable to a specific application. Learning to predict object count and object sizes for a given image is one such problem.

Detecting object sizes in images is useful for a broad range of applications as it can be associated with physical properties like mass, area etc. The size of an object in an image is directly proportional to the area of the pixel covered by the object assuming constant camera to observation plane distance. In the images depicting crowded scenarios with a large number of objects of the same category as shown in Fig. 1.1a, the size distribution of object instances can be of higher interest than the individual object size information. In many applications, especially in the medical field, one is not interested in segmenting every instance of an object as shown in Fig. 1.1b, but rather finding the distribution of object sizes in the image. The size distribution histogram gives us the statistical summary of the object sizes as shown in Fig. 1.1c. Additionally, counting of objects in a crowded environment is a tiresome and time-consuming task which is susceptible to human error. Counting of objects is useful in various real-world scenarios such as microscopic cell counts, crowd monitoring, wildlife census, and tree population estimation of a forest from aerial images.

Typically, the task of object count and size distribution estimation would be approached via explicit, pixel-accurate instance segmentation with a method like Mask R-CNN [18]. These methods can be used to predict the size of each individual object using the estimated mask, and the object count by estimating the number of detected instances. While these methods are conceptually intuitive and robust, they still suffer from the problem of object overlap, occlusion etc. and need an abundance of data and resources. As the object overlap and the partial occlusion increases, the performance of these methods for size estimation decreases because only visible pixels can be classified and thus are responsible for size

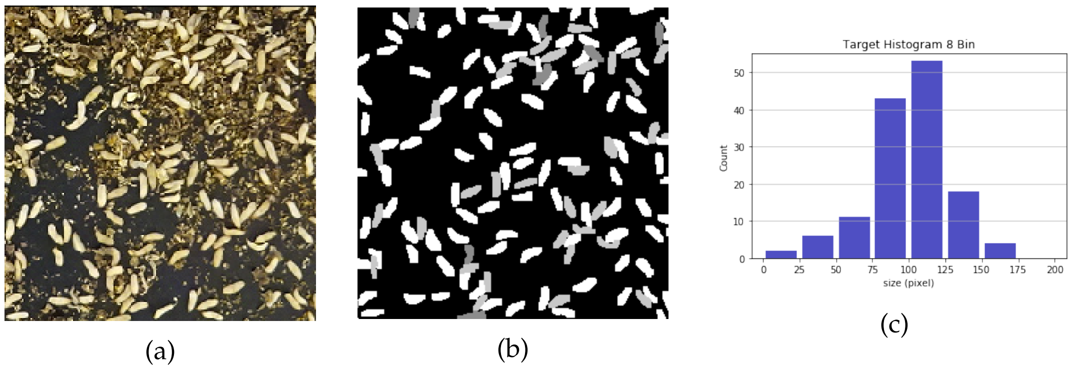


Figure 1.1.: (a) Fly larvae colony (b) pixel-accurate instance segmentation mask (c) size distribution histogram

estimation task as shown in Fig. 1.2. Furthermore, it is well known that instance segmentation methods cannot cope with large object overlap, mainly due to the non-maximum suppression step, missing many objects in the process.

Solving a non-trivial problem of object detection or instance segmentation for slightly less ambitious problems such as object counting and size distribution histogram, is like using a sledgehammer to crack a nut. For those tasks, the size of instance segmentation models is disproportionate with respect to the complexity of the problem. In addition, the number of model parameters are high, which leads to high computational costs. Intuitively, it seems a waste of resources to apply sophisticated instance segmentation methods to a rather simple task, which is estimating the total count and size distribution of object instances. To deal with the problems stated above, we aim to directly predict object count and size distribution histogram without any explicit object instance segmentation. In this way, it reduces the overhead of object detection, per object bounding box and pixel-wise mask prediction.

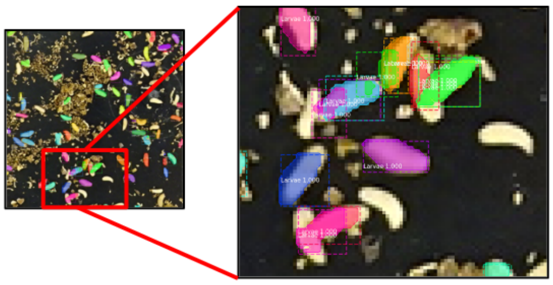


Figure 1.2.: Object overlap problem of instance segmentation methods

1.2. Proposal

The goal of this thesis is to develop a framework for predicting the summarized information of the objects in an input image. In this work, we are proposing to predict size distribution histogram as well as object count and it's justification for the prediction in the form of object spatial localization, in crowded scenes directly without any explicit object instance segmentation. We target highly challenging tasks, for which all object have extremely similar appearance, and thus instance segmentation is difficult.

Our proposals are as follows:

- We propose a novel deep learning architecture (**HistoNet**), which counts and predicts the size distribution of objects directly from an input image, showing superior results with respect to state-of-the-art instance based segmentation methods while having 85% less parameters.
- We present a new data set of 11,000 pixel-wise labeled fly larvae instances, representing crowded scenario and the challenge of predicting size histogram for these small similar looking objects.
- We further evaluate HistoNet's performance on a public cancer cell data set and demonstrate that it achieves good results for this different image modality and application domain.
- We show the robustness of our method on synthetically generated ellipse dataset, having large variance in object size distribution and using deep supervision at hidden layers to further improve our method.

Successful development of such image-based framework for predicting summarized information about objects, will have various real-world applications. One such application is the treatment of municipal organic solid waste by black soldier fly larvae. This is a promising treatment technology for the management of solid waste and it also produces fertilizer and protein for animal feeds [10, 16]. However, for the economic feasibility of this process at an industrial scale, automation is required to calculate the number and size distribution of black soldier fly larvae. Our second application aims at estimating tumor growth directly from medical images. In this, we try to predict cellularity score for malignant cells, by using cell count and size distribution as intermediate results. In this work, we advocate that our approach, while being more accurate, significantly reduces the parameter overhead needed for explicit pixel-accurate instance segmentation and has various applications.

The rest of the thesis is organized as follows. We will first discuss about the background theory needed to understand our work in chapter 2. After this, we will discuss the related work done for counting and size prediction in chapter 3, followed by the developed methodologies for object size distribution and count prediction in chapter 4. We will discuss about datasets used, evaluation strategies and experimental results in chapter 5. Finally, chapter 6 provides a summary of our work and presents future work directions.

2. Background Theory

In this chapter, the theoretical background needed to understand our thesis work is explained. We will discuss about Neural networks, ResNet [19], CountCeption [8] which is the building block of our developed network architecture and Deep Supervised Nets [28].

2.1. Neural Networks

Neural network (NN) is an information processing system, which is primarily inspired by the way biological neural system processes information. Similar to our brain, the basic computational unit in NN is a neuron. Billions of these neurons are interconnected and communicate with each other using synapses. Similarly NN consists of multiple layers of neurons, each layer accepts the input, processes it and hands it over to the next layer. For learning, our biological systems adjust the synaptic connections between the neurons. Similarly, NN adjusts the weights of their neurons to learn from data, extract patterns and detect trends. In this section, we will talk about perceptrons, convolutional neural networks (CNN) and different components of CNN.

2.1.1. Perceptrons

The Perceptron [43] is a single layer neural network, which is used for supervised learning of binary classifiers. Its architecture is shown in Fig. 2.1a. Perceptron unit first linearly combines its inputs using learnable parameters w_i and b. This is further passed through

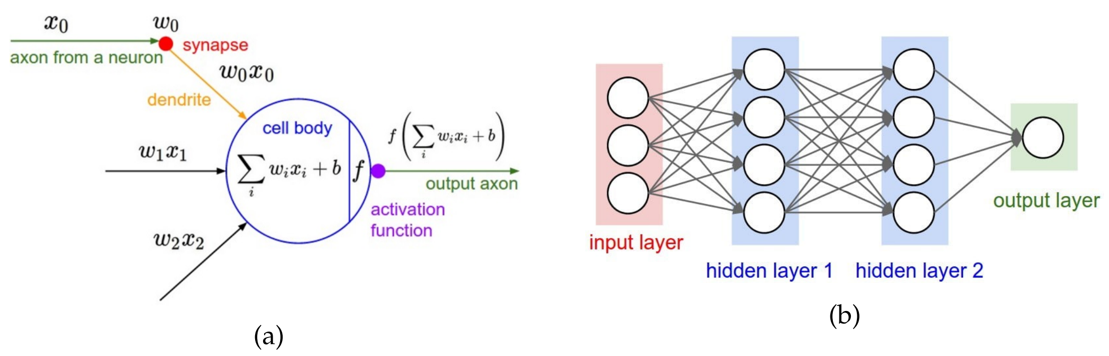


Figure 2.1.: Perceptron (a) mathematical model of a neuron(b) A Multi-layered perceptron with two hidden layers [22]

an activation function which introduces non-linearity in the output. Originally, Perceptron uses step function as non-linearity, which makes it a linear classifier.

$$f(X) = \begin{cases} 1, & \text{if } W.X + b > 0. \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

Where W is a vector of real valued weights, X is input and b is bias. The parameters are learned using a simple iterative algorithm, which converges only if the training data is linearly separable. Therefore, it's impossible for a single layer perceptron to learn the XOR function. Multi-Layered Perceptrons (MLPs) or Feed-Forward Neural Networks overcomes this shortcoming in representational capability as shown in Fig. 2.1b. The layers apart from the input and the output layer are called hidden layers. NNs with at least one hidden layer are universal function approximators, i.e. they can approximate any continuous function [9]. Each layer in an MLP is typically fully-connected i.e. every unit is connected to every unit of the next layer. In each layer, units compute a linear combination of their inputs, followed by a non-linear activation function.

Activation Functions

Sigmoid: This non-linear monotonic function takes an input and squeezes it in the range of 0 to 1, thus sometimes it's used to represent the probability as shown in Fig. 2.2a.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Since the gradient of this function for very large and small values is almost zero, it vanishes the gradient flow during back propagation. This function is not zero centered and always output positive values, which slows down the learning process of parameters.

Hyperbolic tangent function: As shown in Fig. 2.2b $\tanh$ outputs the values in the range -1 to 1. It is zero centered and thus solves the positive output problem of the sigmoid function. Similar to the sigmoid function, it also causes vanishing gradient problem in the network.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

Rectified linear unit: ReLU function solves the vanishing gradient problem. It is computationally inexpensive as compared to sigmoid and tanh which involve exponential function. It is the most commonly used activation function. However, ReLU has zero gradient for negative inputs as shown in Fig. 2.2c and it kills the gradient flow, which is known as ”dying ReLU”.

$$ReLU(x) = \begin{cases} x, & \text{if } x \geq 0. \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

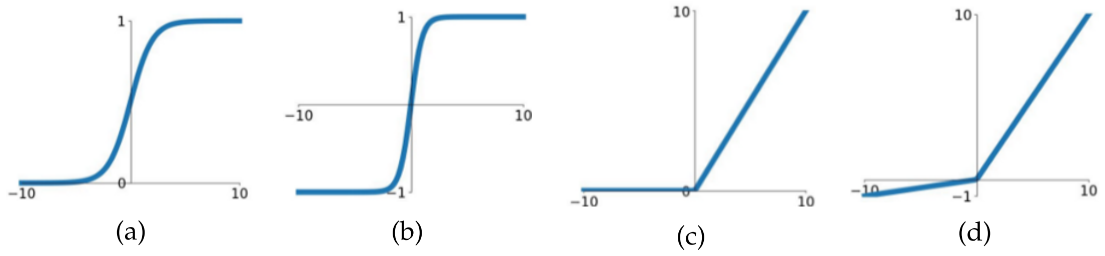


Figure 2.2.: Activation Functions (a) Sigmoid function (b) Hyperbolic tangent function (c) ReLU function (d) Leaky RelU [22]

Leaky rectified linear unit: It addresses the problem of 'dying ReLU' for negative inputs, and provides small negative values for negative inputs and thus enables backpropagation even for negative values as shown in Fig. 2.2d.

$$LeakyReLU(x) = \begin{cases} x, & \text{if } x \geq 0. \\ 0.1x, & \text{otherwise.} \end{cases} \tag{2.5}$$

2.1.2. Convolutional Neural Network

Regular NNs don't scale well with high dimensional data such as images. Owing to the fully-connected layers in MLP, the number of network parameters increase drastically with data dimensionality and network depth. This makes the network susceptible to overfitting the training data. Convolutional Neural Networks (ConvNets or CNNs) are a category of NNs that have proven very effective in areas such as analyzing visual imagery. The main difference between CNN and ordinary Neural Network is that they make explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. The connectivity pattern between neurons in CNNs is inspired by the animal visual cortex. Individual cortical neurons only react to stimuli in a limited area of the visual field known as the receptive field. Different neurons receptive fields partially overlap to cover the entire visual field. Similarly, in CNNs connection between layers are limited to spatially local regions. These convolutional kernels (local parameters) are shared among neurons in the same layer. The shared-weight architecture and translation invariance properties of a CNN allows it to achieve better performance on vision problems. Fig 2.3 shows an example of CNN architecture, which is designed for a classification task. A CNN consists of different types of layers such as Convolutional layer, pooling layer, fully connected layer, batch normalization and dropout layer. We will discuss about these layers in upcoming subsections.

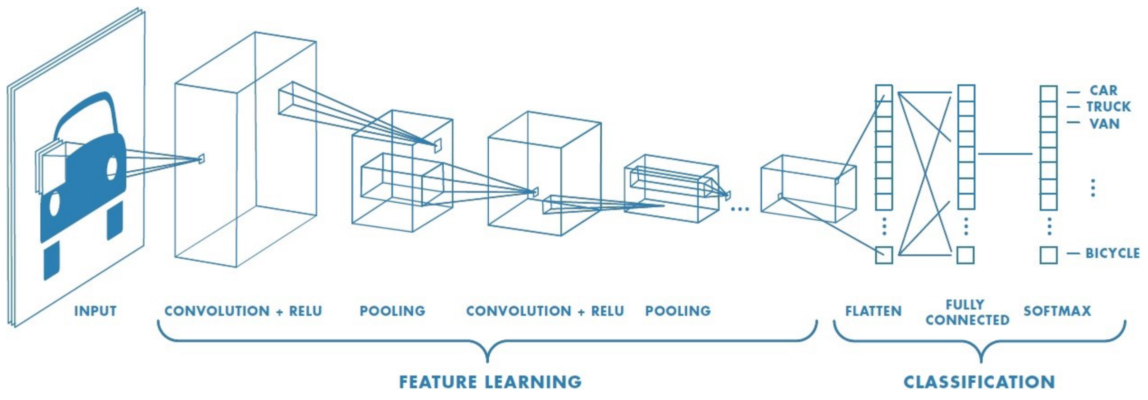


Figure 2.3.: An example of CNN architecture [33]

Convolutional Layer

The core building block of CNN is Convolutional Layer (Conv Layer). Conv Layer's parameters consist of learn able filters (kernels), which have a small receptive field (along width and height) but cover full depth of the input volume. Similar to NNs, Conv Layer in CNNs compute dot products between the entries of the filter and the input. It convolves each filter across the width and height of the input volume and thus produces a 2-dimensional activation map that gives the responses of that filter at every spatial position. Therefore, the network will learn filters that activates when it detects some type of visual feature at different positions in the input volume. Different filters learn different features in the input volume at every position and by stacking these activation maps along the depth dimension, the output volume is constructed. Fig 2.4 shows the convolutional layer.

$$O(C_{out_t}, i, j) = b(C_{out_t}) + \sum_{s=0}^{C_{in}-1} \sum_{q=0}^{m-1} \sum_{r=0}^{m-1} w(s, q, r) * I(s, i+q, j+r) \tag{2.6}$$

Where I is the input, w is the kernel with width and height m, b is the bias, C_{in} and C_{out} are the number of input and output channels and O is the output. The Conv Layer applies kernel to a small region of the input volume and thus enforces sparse local connectivity pattern between neurons of adjacent layers. The size of the kernel controls the extent of this connectivity and is a hyperparameter called the receptive field of the neuron. The Conv layer is used as a feature extractor in CNNs. Following are the hyperparameters which control the size of the output volume:

- **Depth** of the output volume is a hyperparameter, which corresponds to the number of filters in the Conv layer. Each filter learns to detect different features in the input volume.

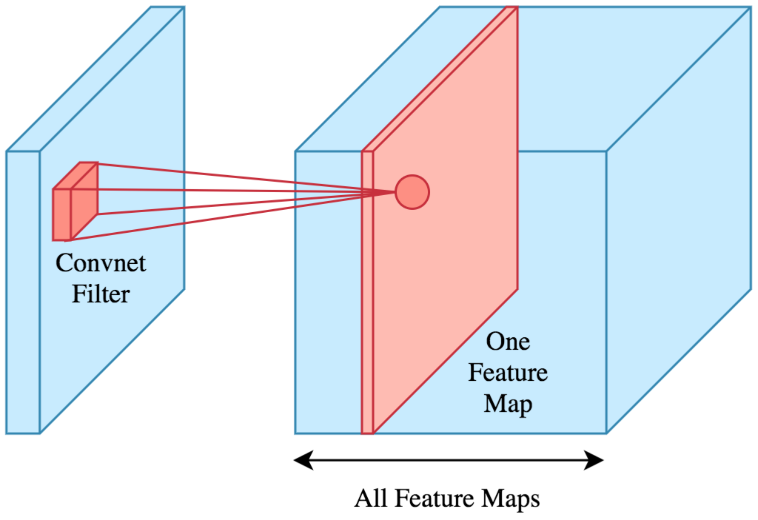


Figure 2.4.: Convolutional Layer [5]

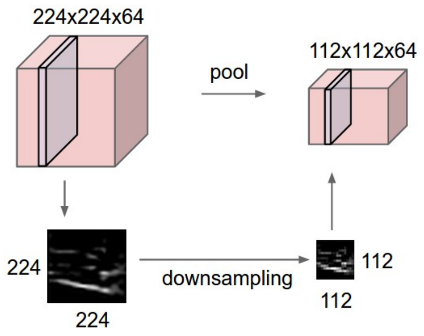
- **Stride** is the number of pixels by which a kernel slides spatially over input volume. For example, if the stride is one, kernel skips 1 pixel and if the stride is 3, kernel skips 3 pixels. As the stride of the kernel increases the size of the output volume decreases spatially.
- **Padding** is used to control the spatial size of the output volume. Convolution operation decreases the size of the output volume as the stride increase. To preserve the input and output spatial size, the input volume can be padded with zeros at the boundaries.

When a Conv Layer with receptive field size m, number of filters (depth) C_{out}, with a stride S and padding P is applied on input volume of spatial size W and depth C_{in}, the spatial size of output volume with depth C_{out} is given as follows:

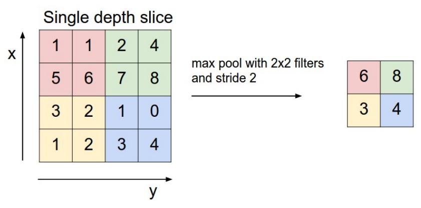
$$\frac{W - m + 2P}{S} + 1 \tag{2.7}$$

Pooling Layer

Pooling Layer is also an essential component of CNN. The pooling layer is used for reducing the size of input volume spatially to reduce the number of parameters and computation in the network, and hence to also control overfitting. Since the pooling filter is applied at all input channels of the input volume independently, the depth of output volume is same as input volume. It is a form of non-linear downsampling. To implement pooling, there are several non-linear functions, among which max pooling is the most common. A max pool layer with filter size $n * n$ and stride n, partitions the input image into non-overlapping



(a)



1	1	2	4
5	6	7	8
3	2	1	0
1	2	3	4

6	8
3	4

(b)

Figure 2.5.: (a) Pooling Layer (b) Max Pooling [22]

rectangles and takes the maximum out of these rectangles. A pooling layer with filter size 2x2 applied and stride of 2, reduces the size of every input channel by 2 spatially and discards 75 percent of the activations. A pooling layer is usually applied in between the successive Conv Layer in a CNN. Apart from Max pooling, average pooling is also commonly used in which the average of the sub region is taken. The pooling layer is used as a feature selector, it selects the strongest activation in the region. Fig 2.5a shows the example of pooling layer, an input volume of size $224 \times 224 \times 64$ pooled with the filter of size 2, stride 2 results into output volume of size $112 \times 112 \times 64$. The volume depth remains the same as input volume. Fig 2.5b shows the max pooling operation in action on 4×4 activation map.

Fully Connected Layer

As seen in MLP, neurons in a fully connected layer have connections to all activation of the previous layer. It is generally applied at the end of the network after several Conv and Max pooling layers. Fully connected layer accumulates global information from lower and mid-level features and provides high-level reasoning in the network.

Batch Normalization Layer

Batch normalization was introduced by Ioffe and Szegedy [20] to improve the speed, performance and stability of deep neural networks. During training, the change at early layers gets amplified at deeper layers as it propagates in the network. This results in covariate shift in the deeper hidden layers. Batch Normalization reduces such shifts and thus accelerate training of the deep networks. This layer forces the output of the previous layer to follow unit gaussian distribution, by subtracting the batch mean and diving by batch standard deviation as shown in Fig. 2.6. It makes the network more robust to different initialization and high learning rates. It is generally applied after fully connected or Conv

Layer and before non-linear activation layer.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure 2.6.: Batch Normalization Layer transform [20]

Dropout Layer

Deep neural networks are prone to overfitting the underlying data. Dropout [48] is a simple and effective regularization technique to overcome overfitting problems in NNs, by preventing co-adaptation between neurons. During each training step, individual neurons are deactivated or dropped out of the network with the probability p as shown in Fig. 2.7. It enforces layers to learn the features based on more neurons and hence improves generalization. It is similar to training a large ensemble of models on a different batch of training data with shared parameters.
We have used above mentioned layers to build a CNN architecture to predict the statistical summary of the objects given an image.

2.2. ResNet

As per Universal approximation theorem [9], a NN of single hidden layer having enough capacity is sufficient to represent any function. But this network is prone to overfitting the data. Deep networks naturally integrate low, mid and high-level features and these levels of features can be enriched by the depth of the network. Many trivial and non-trivial vision tasks have benefited from deep networks. Depth is of crucial importance in NN's performance. After AlexNet [26], CNN architectures are going deeper and deeper, which has become a common trend in the research community.

Learning better network by increasing network depth does not work by simply stacking layers together. In the deep networks, as the gradient is back-propagated to earlier layers, the gradient may vanish or explode, which makes them hard to train. Therefore, as the

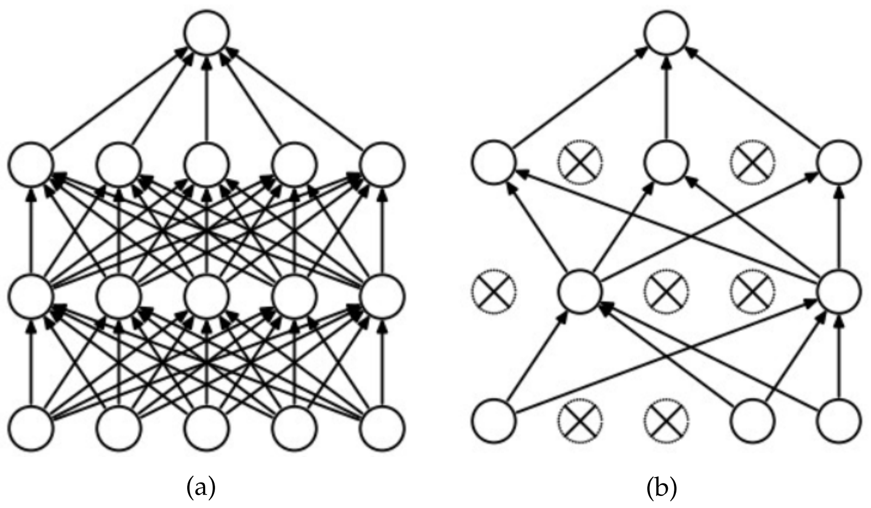


Figure 2.7.: (a) Standard neural network (b) After applying dropout [34]

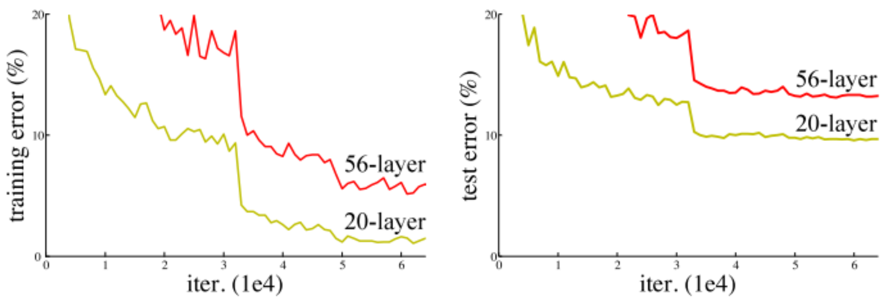


Figure 2.8.: Performance comparison of 56-layer and 20-layer plain network on CIFAR-10 (a) Training error (b) Test error [19]

network depth increases, its performance gets saturated and then degrades rapidly. Unexpectedly, the degradation in performance is not caused by overfitting. Increasing the number of layers to a suitably deep model leads to higher training error as evident from Fig. 2.8. The degradation of training and test accuracy shows that not all networks are easy to optimize. The author of ResNet [19] contends that a shallower architecture and its deeper counterpart with additional layers onto it, should not degrade the performance. If the deeper model is constructed as such, the added layers are identity mapping and the other layers are copied from the shallower model, this should produce no higher training error than its shallower counterpart.

In the paper [19], deep residual learning framework was introduced to tackle degradation problem. They hypothesize that explicitly allowing the stacked layer to fit a residual mapping is an easier task than letting them directly learn a desired underlying mapping. We denote the desired mapping as $H(x)$, we let the stacked nonlinear layers fit $F(x) = H(x) - x$. Thus the original mapping is reformed into $F(x) + x$. It is easier to optimize residual mapping than to optimize original mapping. If needed, the residuals can be pushed to zero to learn the identity mapping, which would be easier than to directly learn identity mapping for a stack of nonlinear layers. As shown in Fig 2.9 a Residual block introduces identity shortcut connections that skips one or more layers. These shortcut connections simply learn identity mapping and add their output with the output of stacked layers. A simple network with identity shortcut connections can be turned into a residual network. This neither increases the parameters of the network nor the computational complexity.

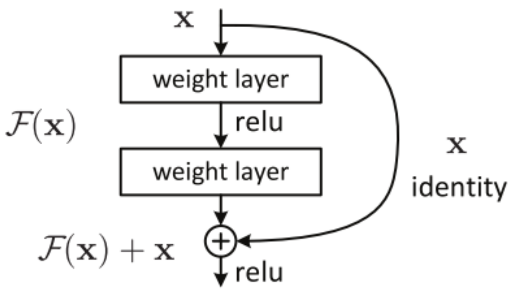

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.9.: Residual Learning: a building block [19]

Sometimes the dimensions of $F(x)$ and x are different, then a projection shortcut is inserted which can be easily implemented using 1×1 convolution. A residual block of 3 layers is

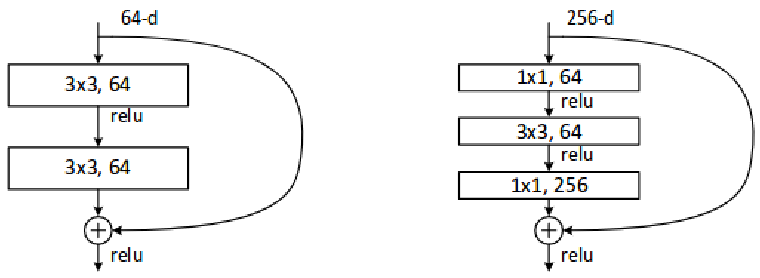
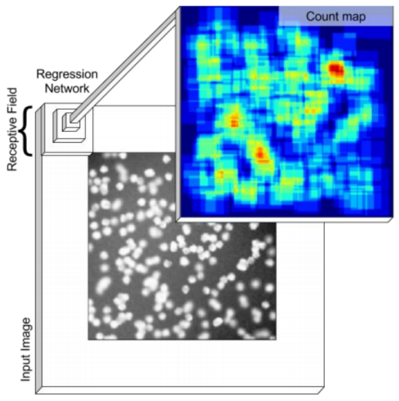
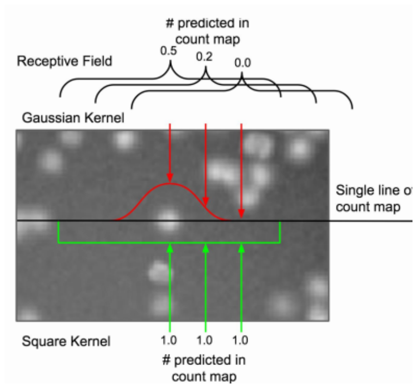


Figure 2.10.: ResNet 2 Layer and 3 Layer block [19]

called bottleneck architecture as shown in Fig. 2.10 right. The two 1×1 convolutions decreases and increases the dimensions, and the middle 3×3 convolution layer is left with smaller input and output dimensions. The 50, 101 and 152 layered residual networks have been proposed in the literature. These networks do not suffer from the degradation prob-



(a)



(b)

Figure 2.11.: (a) CountCeption regression network and redundant count map (b) Gaussian and square kernel [8]

lem and gain accuracy with an increase in depth. Due to the above-mentioned properties of ResNet, we have used it in our network architecture.

2.3. CountCeption

There exist numerous methods for object counting. CountCeption [8] is a novel fully convolutional architecture adapted from inception [49] family of the networks. Given an image as an input, it returns the object count and provides object localization in the form of redundant count map. Instead of predicting density map, Countception predicts count map which consists of redundant object counts on the basis of the receptive field of a small size regression network.

CountCeption processes an image in a fully convolutional way, so each pixel will be accounted multiple times. For example, if the receptive field of CountCeption network is 32, each pixel will be included in 1024 windows ($32X32 = 1024$) i.e. the size of each window. True count of objects can be calculated as an average of redundant count map predictions. Fig. 2.11a shows that a single pixel in the predicted count map is based on the object count in the receptive field of the regression network. Fig. 2.11b shows two different kernels used for predicting redundant count map. The red line indicates the use of gaussian kernel for predicting the count map and the green line indicates the use of square kernel of the same size as the receptive field. The gaussian kernel forces the model to learn how far the object is from the center of the receptive field of the network which makes the task harder than just predicting the presence of an object in the receptive field. Redundant count map using gaussian and square kernels are shown in Fig. 2.12.

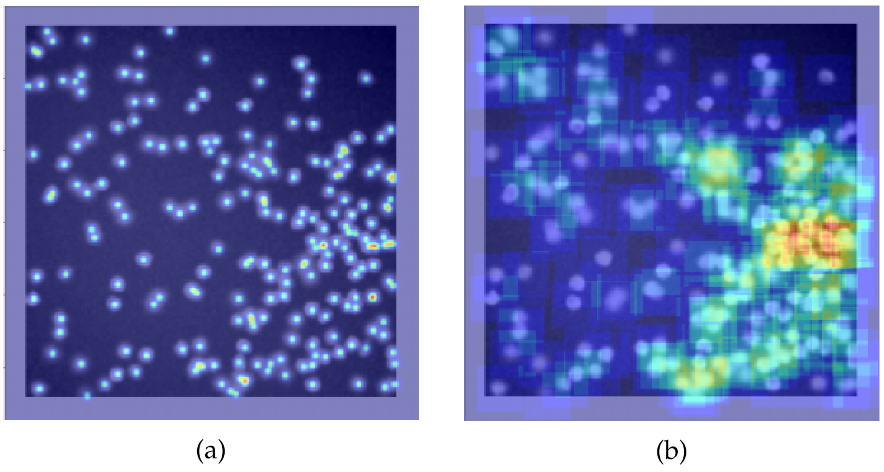


Figure 2.12.: Redundant Count map using (a) Gaussian kernel (b) Square kernel [8]

Although CountCeption network doesn't use any upsampling or deconvolution layer, the size of target redundant count map is bigger than the input image size. Countception network pads the border of the input image and doesn't bottleneck the representation in anyway. Since CountCeption network is small and fully convolutional it has fewer parameters to learn, therefore it reduces overfitting.

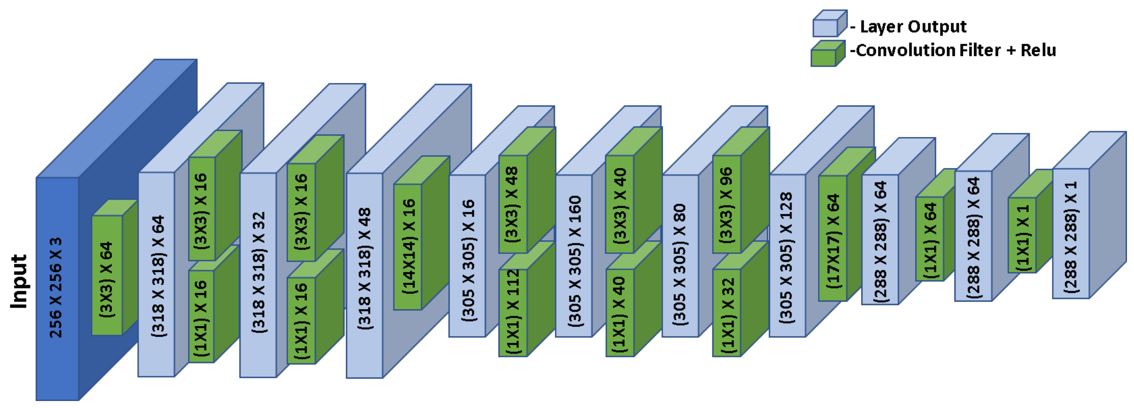


Figure 2.13.: CountCeption Network architecture [8]

Countception network architecture is shown in Fig. 2.13. The first convolution layer zero pads the input image, in order to deal with objects appearing on the border of the image. Following the idea of inception networks [49], CountCeption network uses two types of kernels 1×1 and 3×3 at the same level to handle variations in object size. All other 3×3 convolutions are padded such that the input size is preserved. After each convolution layer batch normalization layers are inserted. The most common way of annotating object for counting is by providing point-annotation, where an object is represented by a single pixel. The target redundant count map can be constructed from the point-annotated map as follows:

$$T(x,y) = \sum_{(x',y')\,\epsilon\,R(x,y)} L(x',y') \tag{2.8}$$

Where $R(x,y)$ is the set of pixel locations in the receptive field corresponding to $T(x,y)$. L is the point annotated map, which is of the same size as input image I. $T(x,y)$ is the total count of objects inside the region of size $r \times r$ receptive field. The network is trained on pixel-wise L1 loss between target and predicted redundant count map. Since the main objective is to count objects, the pixel-wise L1 loss is a proxy objective to the real count which we want. While learning the redundant count map, each object is counted multiple times, which averages over possible errors. The redundant and true count is recovered from redundant count map as follows:

$$\text{\# redundant counts} = \left(\frac{r}{s}\right)^2 \tag{2.9}$$

$$\text{\# true counts} = \frac{\sum_{x,y} F(x,y)}{\text{\# redundant counts}} \tag{2.10}$$

Where r is the width/length of the receptive field, s is the stride length and F is the predicted redundant count map for an input image I. As evident from above equation, with a stride of 1, each object is counted once for every pixel in its receptive field. The number of redundant count decreases as the stride length increases. The true count is recovered by diving the sum of redundant count map with the number of redundant counts. Using redundant count is beneficial in many ways. The CountCeption network can still learn even if the annotated label pixel for an object is not at the center or is at outside the object because the object will appear in the receptive field on an average. Due to the superior performance of CountCeption for object counting, we use it as the backbone of our network architecture.

2.4. Deep Supervised Nets

Deep supervised nets [28] are motivated by the observation that generally the performance of a discriminative classifier will be better if it is trained on highly discriminative features. This signifies that the quality of the hidden layer feature maps of a deep network can be assessed by evaluating the performance of a discriminative classifier trained on them and it also ensures the quality of feature maps at later layers. This feature quality feedback at hidden layers of the network will directly assist in the network parameter update process to facilitate highly discriminative feature maps. The author contends that instead of relying only on the gradients of the output layer, the additional supervision at hidden layers

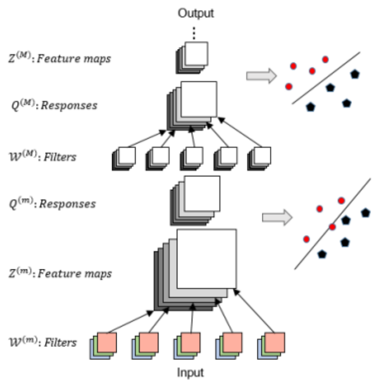


Figure 2.14.: Illustration of deeply supervised nets [28]

will help in rapidly approaching the region of good quality discriminative feature maps.

This method addresses the problem of reduced transparency and discriminativeness of features learned at hidden layers as well as the problem of training difficulty due to exploding and vanishing gradient. It enforces direct and early supervision at hidden layers. It improves the convergence rate over the standard methods, assuming optimization function is strongly convex locally. Additional constraints are applied at the hidden layers as a companion objective to aid the learning process as shown in Fig. 2.14. In this figure, it shows that the supervision is provided at early and later stage of the network, along with the output layer.

The additional output from hidden layers can be thought of as similar to final output that a truncated network might have produced. The backpropagation in the network not only uses the gradient from the final layer, but also from additional local companion outputs. The results from deep supervised nets [28] shows that the additional objectives at hidden layers act as feature regularization and it also helps in achieving faster convergence using small training data. We have used the idea of deep supervised network in our developed methodology to further improve our results.

3. Related Work

In recent years, a number of approaches have been developed to tackle the counting of objects but not so much work has been done for explicit object size distribution estimation. In this chapter, we will discuss about the developed approaches for object counting and object size estimation task.

3.1. Object Counting

Counting objects in images has been a focal point in computer vision research for several years. Earlier some unsupervised approaches were developed for tackling counting problem, based on motion similarities [38] and self-similarities [3]. Due to the limited counting accuracy of these methods, later developed approaches were based on supervised learning. Various approaches of the pre-deep learning era are designed using bottom-up image processing workflows to count objects, segmented with edge detectors [45]. These classical approaches comprise of fine-tuning edge detectors for segmenting objects and count them. For dealing with overlapping objects these methods required watershed transformation. A downside of these approaches is their large number of hyper-parameters that has to be set for each new data set. Approaches for counting of objects can be broadly divided into two categories.

Counting by Regression: These methods [7, 25, 32] avoid directly solving the hard object detection problem but instead directly learn a mapping from the input image to the number of objects. Some method learns mapping to output 1-dimensional statistics i.e. total count, the object location information is discarded [1, 25]. Training these methods require a large number of images labeled with total object count. An elegant method to estimate total object count in images is density estimation. These methods propose to output density map to provide object localization and justification of object count. Integral over the predicted density map gives the total count. Lempitsky et. al [29] computed Scale-Invariant Feature Transform (SIFT) features for the input images and predicted density maps via regression. Fiaschi et. al [11] improved density mapping by using regression forests. Recent works turn to Deep Learning [41] for joint semantic segmentation and density estimation, to identify and count particular tree species of sub-pixel size at very large scale from satellite images. Xie et. al [51] used fully convolutional regression network to output a cell spatial density map across the image to predict total count. Another approach [52] uses structured regression to predict proximity patch, which has higher value for pixel near cell center. A very powerful architecture custom-tailored for object counting was introduced in [8]. It processes the image in a fully convolutional manner and predicts redundant counting instead of density mapping in order to average over errors. Its main insight is that predicting the existence or absence of an object within a receptive field is an easier task than predicting density maps. The latter is harder because in addition to predicting the existence of an object, it has to estimate how far the object is from the center of the receptive field. Due to its very redundant convolutions per image location, this architecture gives good results while being efficient to train.

Counting by detection: These approaches detect the objects and one can count the detected object to retrieve the count. By using visual object detector, individual object instances in the image are localized. Counting becomes trivial after localization of all instances. Some detection methods follow the two-stage propose and verify approach. Methods such as R-CNN [13] and there improved versions Fast R-CNN [12], Faster R-CNN [40], and Mask R-CNN [18] has improved the detection quality. But due to the two stage approach, these methods are slow in inference time. In recent time some one-stage approaches has been developed such as Single Shot detectors (SSD) [31] and You Only Look Once (YOLO) [39] networks which have significantly improved the inference time.

For many applications, object counting is not enough, as the size distribution of the objects in the scene are key to determine, e.g. malignant cell evolution. We propose an architecture that not only counts objects but also predicts their size distribution without explicit instance segmentation.

3.2. Object Size

Although some methods have taken up size as their secondary task, not so much work has been done for estimation of direct size distribution of the objects in an image. We can predict the object size by performing explicit object detection or instance segmentation. The last few years have seen considerable progress in object detectors [14, 40], as well as instance segmentation methods based on Deep Neural Networks [18]. An advantage of these approaches is that they also provide object size as a by-product. This can be approximated by the area of the bounding box enclosing the objects or better estimated if one extracts instance masks for all objects. These masks that stem from an instance segmentation inside each individual bounding box can readily be used to estimate object sizes more accurately than merely relying on bounding boxes. To the best of our knowledge, this strategy is the most accurate and robust today to predict object count and size, thus we use it as the baseline for this work.

A clear downside of the size and counting-by-detection strategy is that we solve a much harder problem, i.e. instance segmentation, in order to predict total counts and object size distributions. This means using large models and complex training schemes to obtain pixel-accurate instance delineation, even though this information is not needed as output. Additionally, overlapping instances and occluded objects often lead to errors. We propose to directly learn to predict size distributions without explicit instance segmentation or object detection. Our method directly predicts global dataset statistics i.e. object size histograms and object count with much higher accuracy and using a leaner architecture with 85% less parameters compared to Mask R-CNN [18].

4. Methodology

Our goal is to develop a method which will give the abstract statistical summary about objects in the given image, by estimating the total count of the objects and providing justification for the count in terms of localization and by estimating size distribution histogram. In this section, we will discuss about our developed approaches, network architectures and loss functions.

4.1. HistoNet

Our novel data-driven method *HistoNet* directly predicts global data statistics of an image, namely, total object count in cluttered scenes and object size distribution histogram. More formally, given an input image I, our aim is to predict a redundant count map P_{map} and size histogram P_{hist}. The CountCeption [8] method works really well for object counting using redundant count map as regression target as discussed in section 2.3. The activation maps after each layer in CountCeption looks as shown in Fig. 4.1 Activation maps of CountCeption network after the first and second layer activates the objects and nearly segments them from the background. Building upon this observation, we construct a network *HistoNet*, Which have two separate heads as follows:

- Lower branch: We use CountCeption network as our backbone network to predict total object count using redundant count map.
- Upper branch: Building upon the observation from Fig. 4.1, we take a separate head from the first layer of CountCeption and pass the activation map with nearly segmented objects, through ResNet-50 [19] network to predict size distribution histogram.

For size distribution histogram prediction head, we experimented with different networks out of which ResNet-50 produced significantly better results. In the next subsection, we will discuss about the network architecture and loss functions in detail.

4.1.1. Network Architecture

As shown in Fig. 4.2, *HistoNet* consists of two branches, one for predicting object count and the other for histogram prediction which estimates the size distribution of object instances. *HistoNet* takes an image I of size 256×256 pixels as an input and predicts a redundant count map P_{map} of size 288×288 pixels in a fully convolutional manner (lower

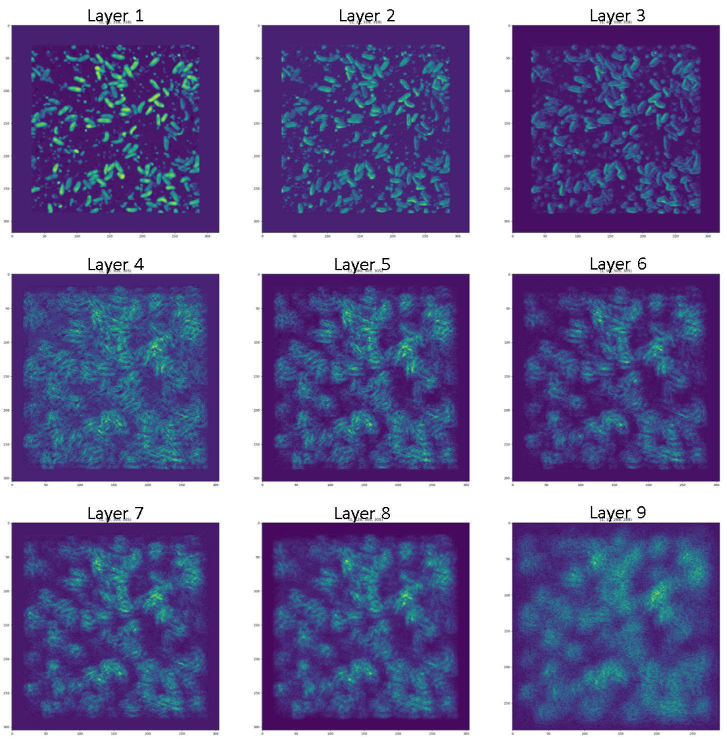


Figure 4.1.: CountCeption - Activation maps after each layer

branch). Note that neither upsampling nor deconvolutions are computed for predicting a map of larger size than the input image. For this, *HistoNet* zero-pads the input image in its first layer having padding width equal to the receptive field of the lower branch of the network. In the Fig. 4.2, two green boxes at the same level in the lower branch of the network represents the application of two kernels on the same input and concatenation of their outputs to handle the variations in object size. For size histogram prediction, the upper branch uses ResNet-50 [19] network on top of the first layer of the lower branch of the network because of its robustness against the vanishing gradients. In the upper branch, instead of using a standard fully connected layer, we add two convolutional layers with kernels of size $3 \times 3 \times 256$ and $1 \times 1 \times 16$ at the end of ResNet-50 network. These convolutional layers are followed by two fully connected layers interspersed with dropout layers. Our final output is a histogram of object sizes P_{hist}. We can capture the size distribution at a finer scale, by increasing the number of bins. In this work, we use 8-bin and 16-bin size distribution histogram as final output.

Symbol	Description
I	Input Image
P_{map}	Predicted Count Map
T_{map}	Target Count Map
P_{hist}	Predicted Histogram
T_{hist}	Target Histogram
$p(H)$	Probability Distribution of Histogram H
L_{count}	Pixel Wise Count Map Loss
L_{wL}	weighted L1 Histogram Loss
W	Weights assigned for L_{wL}
L_{KL}	KL-Div Histogram Loss

Table 4.1.: Notation Summary

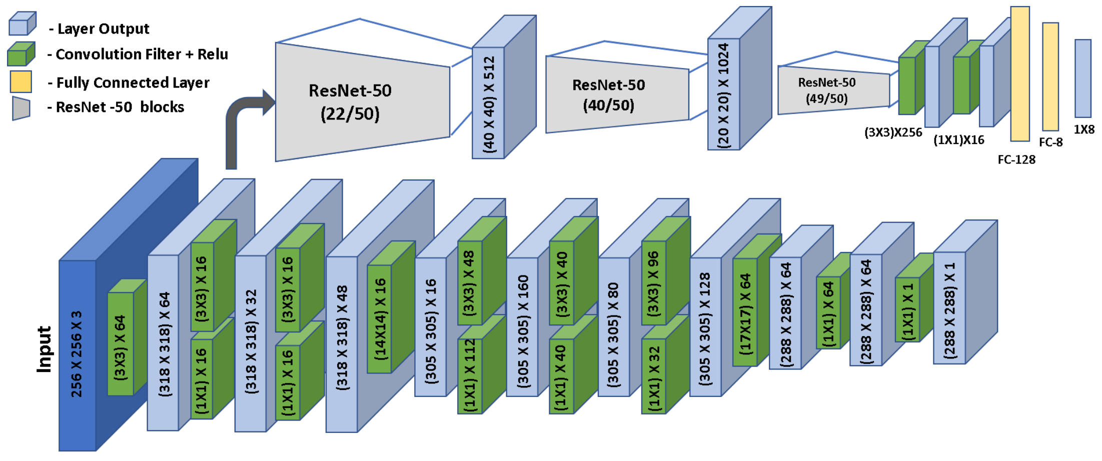


Figure 4.2.: HistoNet Architecture

4.1.2. Loss Functions

To train the network for our multi-objective network, we impose losses on count prediction as well as on histogram distribution prediction. A notation summary for the upcoming equations is given in Table 4.1.

Count Map Loss

For training the network to predict redundant count map, we impose pixel-wise L1 loss between predicted and target redundant count map. Since L2 loss was too harsh for training, L1 loss was selected.

$$L_{count} = \|P_{map}(I) - T_{map}(I)\|_1 \tag{4.1}$$

Where P_{map} is the predicted count map, T_{map} is target count map and $\|.\|_1$ is L1 norm. Since one of our objectives is to predict total count, the pixel-wise L1 loss is helping the network to implicitly learn total object count. If we combine this pixel-wise L1 loss with loss based on total count prediction, it will cause the network to overfit and will not assist in training. The network in order to predict the overall count will try to learn artifacts in each image.

KL-Divergence Loss [27]

For learning to predict size histogram, we impose KL-Divergence loss to capture the underlying probability distribution. The KL-Divergence loss measures the degree of dissimilarity between the predicted and ground truth distributions. By imparting this loss, we will capture the shape of the histogram. For this, we converted both of our histogram size distributions i.e. predicted and target histogram, to their probability distributions.

$$L_{KL} = \sum_{bins} p(T_{hist}) log \left(\frac{p(T_{hist})}{p(P_{hist})} \right) \tag{4.2}$$

Where $p(T_{hist})$ and $p(P_{hist})$ are the probability distributions of target and predicted size histogram.

Weighted L1 Histogram Loss

Having imposed the KL-Divergence loss to capture the underlying probability distribution, we now want to capture the scale of the histogram. For this, we tried various loss functions and found that L1 Loss performed the best. Moreover, a weighted L1-loss, where weights W are assigned according to the normalized center values of the respective bins, further improves the result. Our intuition is that larger objects should incur higher penalty than smaller ones if missed. For calculating weights, we first compute the centers of the histogram bins. After this, we calculate weights by normalizing center bin values i.e. by dividing them with the sum of bin center vector. For example, an 8-bin histogram with object size between 0 and 200 pixels, we get $CenterBin$ = [12.5, 37.5, 67.5, 87.5, 112.5, 137.5, 167.5, 187.5] and compute weights and weighted L1 loss as follows:

$$W = \frac{CenterBin}{\sum CenterBin} \tag{4.3}$$

$$L_{wL} = \sum_{bins} W \left| P_{hist} - T_{hist} \right| \tag{4.4}$$

Where P_{hist} is predicted histogram and T_{hist} is target histogram. The object size-weighted L1-loss in combination with the KL-Divergence loss are mutually reinforcing to capture shape and scale of the histogram.

Total Loss

We jointly train our network on this multi-task loss and minimize it for learning to predict count map and size histogram.

$$L_{total} = L_{count} + 0.5L_{KL} + 0.5L_{wL} \tag{4.5}$$

We empirically found that giving equal weight to KL-loss and weighted L1-loss gives best results.

4.2. Deep Supervised HistoNet

Directly learning fine-grained 8 or 16 bin histograms can be tricky for the network. In order to help the network focus the learning on the hard cases near the bin boundaries, we propose to gradually increase histogram resolution towards the deeper layers. We first learn a 2-bin and 4-bin histogram and later allow the network to increase the resolution to 8 or 16 bins. Fig. 4.3 shows that as we increase the resolution of the histogram from 2-bin to

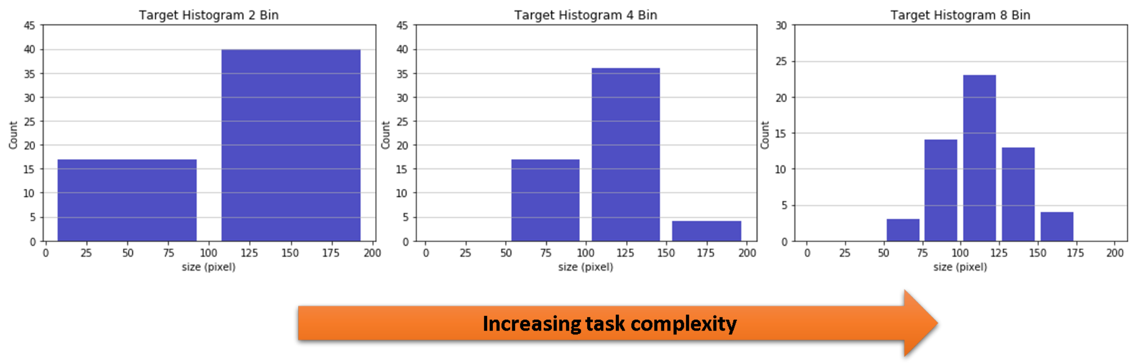


Figure 4.3.: Task complexity for histogram prediction

8-bin the complexity of the model required to predict them also increases. We use the idea behind Deeply Supervised Nets (DSN) [28], which is shown to be helpful in calibrating the model at intermediate stages by enforcing direct and early supervision for both the hidden layers and the output layer.

4.2.1. Network Architecture

We show the deep supervision modules on the upper branch as trapezoids in Fig. 4.4. These have an effect on the early hidden layers and serve as an additional constraint to gradually force the network to split size bins into smaller intervals. Our deep supervision signal at early and middle stage of the histogram branch enforces first a split of sizes into two bins and the following one into four bins. Therefore, in addition, to count map and size

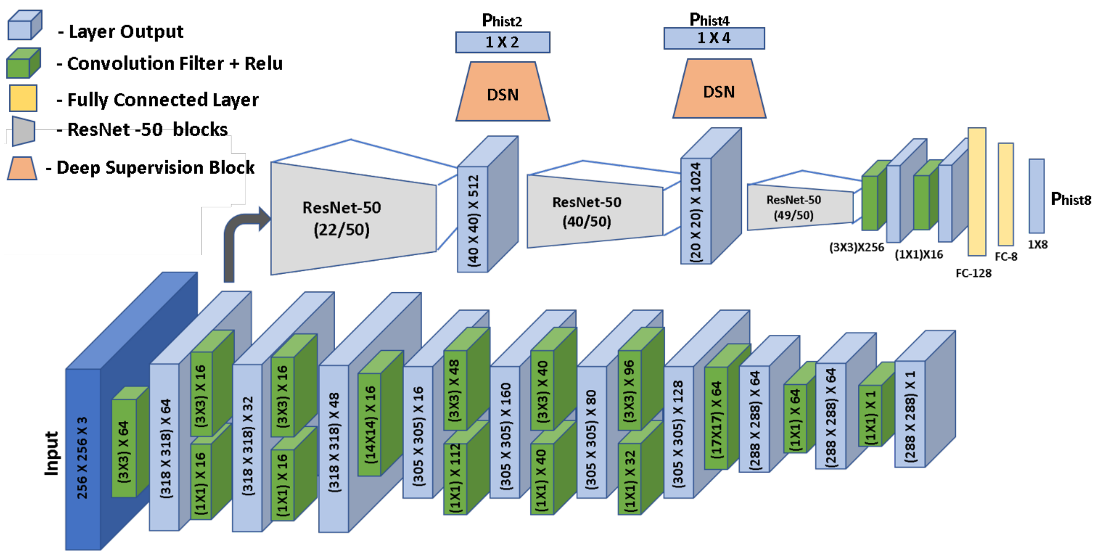


Figure 4.4.: HistoNet-DSN architecture

histogram, we predict 2-bin, 4-bin histograms P_{hist2} and P_{hist4}, respectively. To implement deep supervision, we add a stack of two convolutions followed by two fully connected layers to predict the 2-bin histogram from early layers in the histogram branch as shown in Fig. 4.5. The same block is added at a middle stage for predicting 4-bin histogram. Our full model architecture of *HistoNet-DSN* with deep supervision at intermediate stages of the histogram branch is shown in Fig. 4.4.

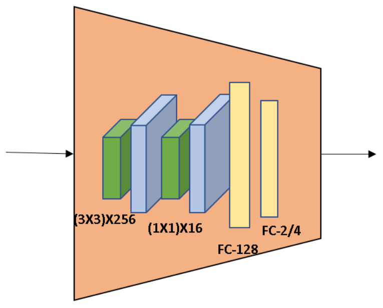


Figure 4.5.: DSN Block for 2/4 bin histogram

4.2.2. Loss Functions

For training *HistoNet-DSN*, we define additional output losses alongside our main objective function. We add KL-divergence and weighted L1 loss for 2-bin and 4-bin histogram predictions.

$$\begin{aligned} L_{total} = L_{count} &+ 0.5(L_{KL} + L_{wL}) \\ &+0.2(L_{KL2} + L_{wL2}) \\ &+0.3(L_{KL4} + L_{wL4}) \end{aligned} \tag{4.6}$$

Where L_{KL}, L_{KL2} and L_{KL4} is kullback-liebler divergence loss for 2-bin, 4-bin and 8-bin histogram respectively. Similarly L_{wL}, L_{wL2} and L_{wL4} is weighted L1 loss for 2-bin, 4-bin and 8-bin histogram respectively. We empirically found that giving more weight to final histogram output loss followed by 4-bin histogram and 2-bin histogram gives better results.

5. Experiments and Results

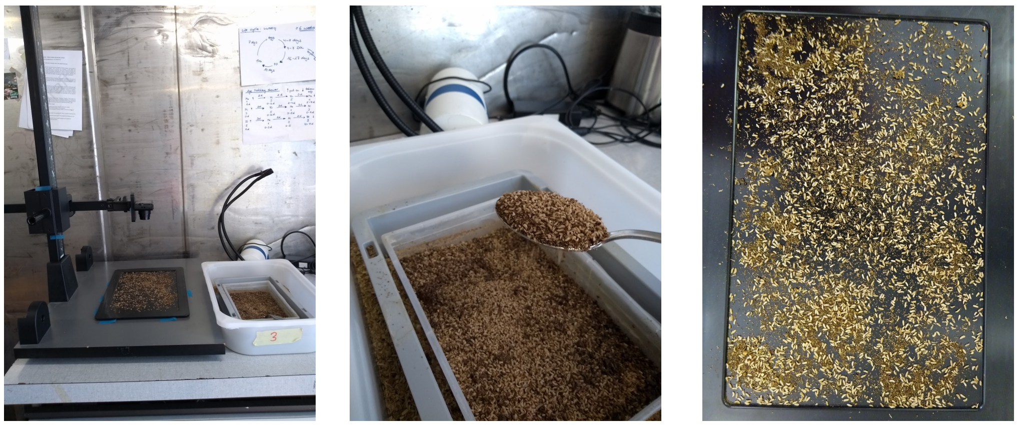
We evaluate the performance of our developed methods on three different datasets. First, we compare the performance of our methods with state-of-the-art instance segmentation method Mask R-CNN [18] on our new *Fly Larvae* dataset, which contains a high density of similar looking objects. In addition, we present experiments on a synthetic ellipse dataset, where we can adjust the density and object size distributions arbitrarily, in order to show the robustness of our method and its ability to predict diverse histogram shapes. Finally, we run experiments with a medical image dataset to verify the applicability of our method to a different image modality as well as for the specific purpose of estimating the tumor cellularity score. In this section, we will discuss about our datasets, evaluation strategies and quantitative results.

5.1. Datasets

5.1.1. Fly Larvae Dataset

We create a new dataset of soldier fly larvae, which are bred in massive quantities for sustainable, environment friendly organic waste decomposition [10, 16]. Fly larvae images were collected using a Sony Cyber shot DSC-WX350 camera with image size 1380×925 pixels as shown in Fig. 5.1c. The camera is installed on a professional fixture to guarantee a fixed distance from camera to observation plane for all image acquisitions as shown in Fig. 5.1a. This is important to avoid any scale variation between the different image acquisitions. Very large numbers of larvae mingled with a lot of larvae feed lead to high object overlap and occlusions. These similar looking brown colored fly larvaes have variations in their sizes and flexible structure. To simplify our tasks, we choose high contrast black color background. The images were collected by using one spoonful (Fig. 5.1b) of larvaes weighing approximately 3-6 grams (1000-2000 Larvaes), uniformly scattered over the image area as shown in Fig. 5.1c. All larvae instances are labeled pixel-wise.

For experiments, we sample patches of size 256×256 pixels from the original images. A sample image of this dataset is shown in Fig. 5.2a which exhibits the crowded scenarios having 137 fly larvaes overlapping and occluding each other. From the pixel-wise labeled mask as shown in Fig. 5.2b, using the centroid information of each larvae instance we create the target redundant count map (Fig. 5.2c) for a receptive field of size equal to the total receptive field of our network i.e. 32×32. We also create 8-bin and 16-bin size histogram (Fig. 5.2d, 5.2e) using the size of masks. It is evident from the Fig. 5.3a, 5.3b, that



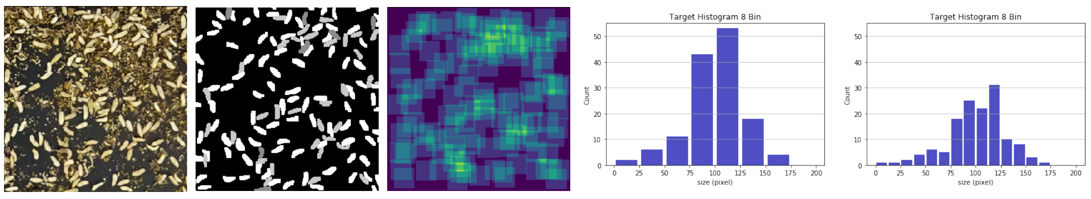
(a) (b) (c)

Figure 5.1.: Experimental Setup: (a) Camera setup (b) Spoonful of fly larvaes for image acquisition (c) Captured sample image

the sizes of larvaes in the dataset approximately follows Gaussian distribution A summary of the *Fly Larvae* dataset is given in Tab. 5.1.

5.1.2. Synthetic Ellipse Dataset

In order to check whether our method really predicts different size distributions or simply learns the Gaussian by heart, we create a synthetic dataset of thin ellipses resembling fly larvaes with strongly varying size distributions. To have a large variance in size and shape of ellipse instances, we randomly vary various parameters such as major, minor axis length, position, full or half arc length of ellipses. These images itself can be used as pixel-wise labeled masks. We created redundant count map and size distribution histogram following the similar process as *Fly Larvae* dataset. Fig. 5.4 shows one image sample and its corresponding ground truths of synthetic ellipse dataset. The size distribution of synthetic ellipse has more variance as apparent from Fig. 5.3c, 5.3d. The standard deviation



(a) (b) (c) (d) (e)

Figure 5.2.: Example of Fly Larvae dataset: (a) input image (b) pixel-accurate mask (c) redundant count map (d) 8-bin size histogram (e) 16-bin size histogram

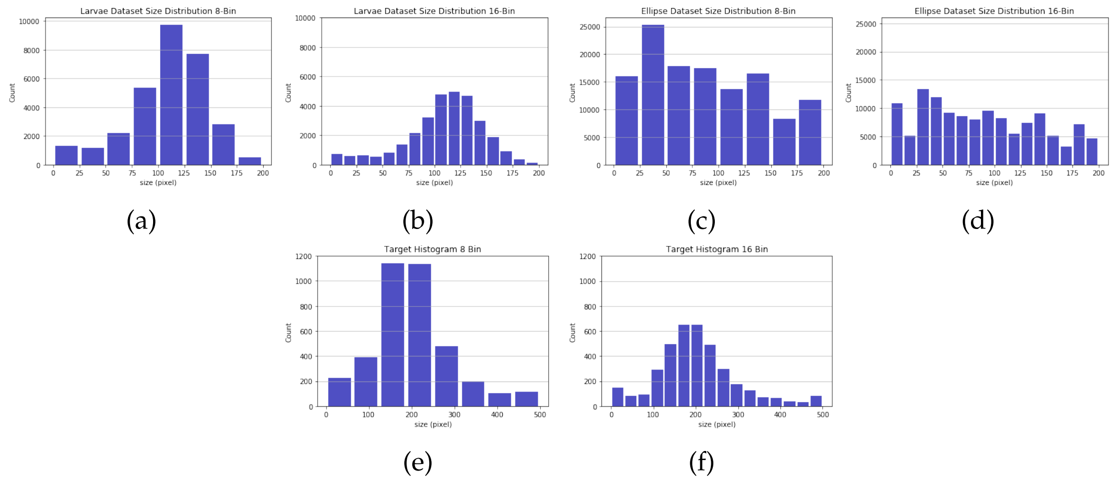


Figure 5.3.: Size distribution histogram of datasets: (a) Fly Larvae 8-Bin (b) Fly Larvae 16-Bin (c) Synthetic ellipse 8-Bin (d) Synthetic ellipse 16-Bin (e) Breast cancer cell 8-bin (f) Breast cancer cell 16-bin

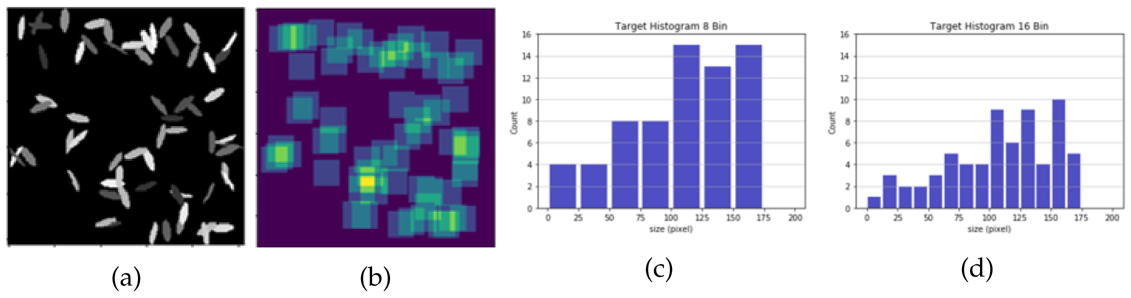


Figure 5.4.: Example of Synthetic ellipse dataset: (a) input image (c) redundant count map (d) 8-bin size histogram (e) 16-bin size histogram

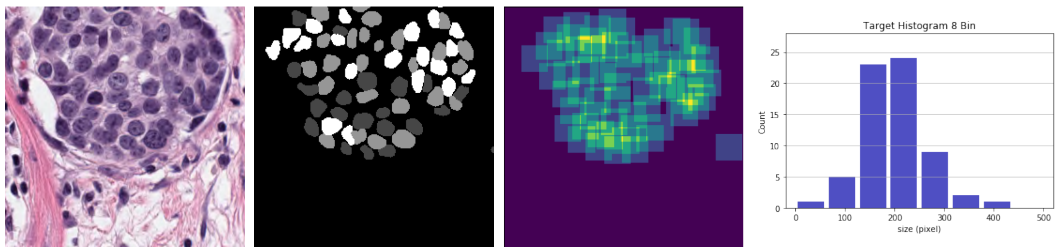
of ellipse instance sizes is nearly double that of the *Fly Larvae* dataset. A summary of this dataset is shown in Tab. 5.1.

5.1.3. Breast Cancer Cell Dataset

In order to validate the applicability of our methods to a different image modality and image content, we are using the breast cancer cell data set [2] that was originally recorded for the BreastPathQ Challenge. Breast Cancer patients with locally advanced disease can be treated with Neoadjuvant treatment (NAT). In addition to tumor size, NAT may change the tumor cellularity, which makes it an important factor to access the response of tumor to the treatment. The cellularity of any given image patch depends on the area covered by malignant cells. Currently, tumor cellularity is estimated manually by the pathologist.

	Larvae data set	Synthetic Ellipse	Breast Cancer Cell
No. Objects	10844	135318	3783
Size (pixels)	120.2 ± 28.1	94.5 ± 63.2	205.9 ± 108.6
Count	80.4 ± 40.7	44.8 ± 20.8	30.5 ± 22.8
Size range (pixels)	0-200	0-200	0-500

Table 5.1.: Dataset Summary



(a) (b) (c) (d)

Figure 5.5.: Example of Breast cancer cell for training HistoNet: (a) input image (b) pixel-wise malignant cell mask (c) redundant count map (d) 8-bin size histogram

This might be subjected to inter-observer variability which affects the quality and reliability of results and assessment in NAT trials. Using the tumor size distribution and cell count as intermediate results, our developed approach *HistoNet* can help in estimating cellularity score.

The public BreastPathQ Challenge dataset consists of 2579 image patches of size 512×512 pixels, and each patch is assigned a tumor cellularity score between 0 to 1 by one expert pathologist. We resize the images to size 256×256 pixels to make it compatible to the model input size. Three example images of this data set having different cellularity scores are shown in Fig 5.6.

The BreastPathQ Challenge also provides an additional dataset which contains 153 images having variable sizes with annotated lymphocytes, malignant epithelial and normal epithelial cell nuclei. We add pixel-accurate labels to 18 of these images for training *HistoNet* model to predict malignant cell size distribution and count. We use the rest of these images to create redundant count map to train the lower branch of *HistoNet*. We sample the image patches of size 256×256 pixels and prepare redundant count map and histogram. The summary of this manually pixel-wise labeled dataset is provided in Table 5.1.

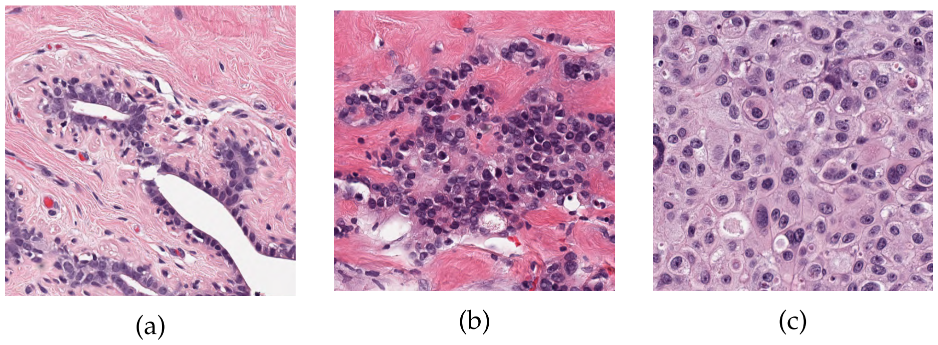


Figure 5.6.: Example of breast cancer cell data set (a) Cellularity Score 0.0 (b) Cellularity Score 0.5 (c) Cellularity Score 1.0

5.2. Evaluation Measures

To evaluate object instance count and size histogram prediction performance, we use several quantitative measures described as follows:

Mean Absolute Count Error (MAE): For counting, we use the Mean Absolute Count Error (*MAE*), which takes the absolute difference between predicted and target count.

$$MAE = \sum_{x,y \epsilon I} \frac{\|P_{map}(x,y) - T_{map}(x,y)\|}{r^2} \tag{5.1}$$

Where r is the receptive field of the lower branch of the *HistoNet* network, I is the image, P_{map} and T_{map} are predicted and target redundant count map respectively. By dividing summation of redundant count map with the area of the receptive field, we extract the total count.

Kullback-Leibler Divergence (*kld*): To measures the degree of dissimilarity between the predicted and ground truth size distribution histograms. We compute the Kullback-Leibler divergence [27] (*kld*) between predicted and target histogram.

$$kld = \sum_{bins} p(T_{hist}) log \left(\frac{p(T_{hist})}{p(P_{hist})} \right) \tag{5.2}$$

Where $p(P_{hist})$ and $p(T_{hist})$ represents the probability distribution of predicted and target histogram.

Bhattacharyya distance (*bhatt*): To measure the difference between predicted and target histogram, we use Bhattacharyya distance [21]. It measures the amount of overlap between two statistical samples or populations.

$$d_{bhatt} = \sqrt{1 - \frac{1}{\sqrt{\overline{P_{hist}}\,\overline{T_{hist}}N^2}} \sum_{bins} \sqrt{P_{hist}T_{hist}}} \tag{5.3}$$

Where N is the total number of bins in the histogram, $\overline{P_{hist}}$ and $\overline{T_{hist}}$ represents the mean of predicted and target histogram.

Chi-Square distance [35] (χ^2): It is used to determine whether there is a significant difference between the expected frequencies and the observed frequencies in one or more histogram bins.

$$\chi^2 = \sum_{bins} \frac{(P_{hist} - T_{hist})^2}{(P_{hist} + T_{hist})} \tag{5.4}$$

Intersection (*isec*): It calculates the similarity of two histograms, with possible values of the intersection ranging between 0-1, where 0 represents dissimilar histograms, and 1 identical histograms.

$$isec = \frac{\sum_{bins} \min(P_{hist}, T_{hist})}{\max(\sum_{bins} P_{hist}, \sum_{bins} T_{hist})} \tag{5.5}$$

Correlation (*coor*): Histogram correlation is another measure to compute the similarity between two histograms.

$$corr = \frac{\sum_{bins}(P_{hist} - \overline{P_{hist}})(T_{hist} - \overline{T_{hist}})}{\sqrt{\sum_{bins}(P_{hist} - \overline{P_{hist}})^2 \sum_{bins}(T_{hist} - \overline{T_{hist}})^2}} \tag{5.6}$$

Prediction Probability Measure (P_K): To assess the performance of methods developed to predict cellularity score on Breast cancer cell dataset, we follow the challenge rules, and use the prediction probability measure [47]. This is calculated for each method for each reference standard (pathologist 1 and pathologist 2), then averaged to determine a final overall prediction probability value. In case, if the P_K values across multiple methods are same, then the P_K with mean reference value lower than 50% is used as a tie breaker. The P_K value is calculated as follows:

$$P_K = \frac{1}{2}\left(\frac{P - Q}{P + Q + T} + 1\right) \tag{5.7}$$

Where P is the number of concordant pairs, Q the number of discordant pairs, and T the number of ties only in the calculated labels.

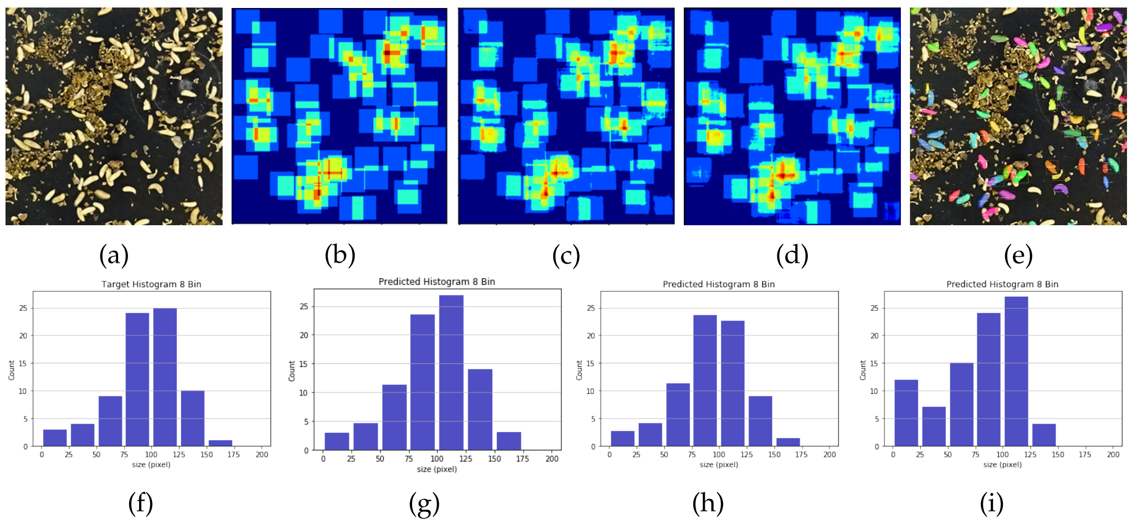


Figure 5.7.: Fly larvae 8-Bin Histogram Prediction: (a) input image (b) Target redundant count map (c) HistoNet countmap prediction (d) HistoNet-DSN countmap prediction (e) Mask R-CNN instance segmentation mask prediction(f) Target Histogram (g) HistoNet histogram Prediction, (h) HistoNet-DSN histogram prediction (i) Mask R-CNN histogram prediction

5.3. Results

5.3.1. Fly Larvae Dataset

We evaluate our developed methods i.e. *HistoNet* and deep supervised version of HistoNet i.e. *HistoNet-DSN* for predicting 8-bin histograms and more fine-grained 16-bin histograms of object sizes. We benchmark our methods against Mask R-CNN [18] as a baseline. Recall that Mask R-CNN predicts pixel-accurate instance labels instead of directly outputting an object size distribution. We thus explicitly do instance segmentation and compute sizes by summing over instance pixels. After this, we compute size distribution histogram from individual object size information.

Fig. 5.7 shows the comparison of object count and 8-bin size histogram prediction for different methods for a test image containing 76 fly larvaes. The redundant count map prediction for both *HistoNet* and *HistoNet-DSN* methods are nearly identical to the ground truth count map as shown in Fig. 5.7b, 5.7c, 5.7d. Our approach predicts histogram which are close to the ground truth size histograms, it captures the shape and scale of the target histograms as evident from Fig. 5.7f, 5.7g, 5.7d. Mask R-CNN over and under-predicts the masks of larvaes as shown in Fig. 5.7e, due to overlaps and occlusion of larvaes, thus it affects the size distribution histogram predictions as depicted in Fig. 5.7i.

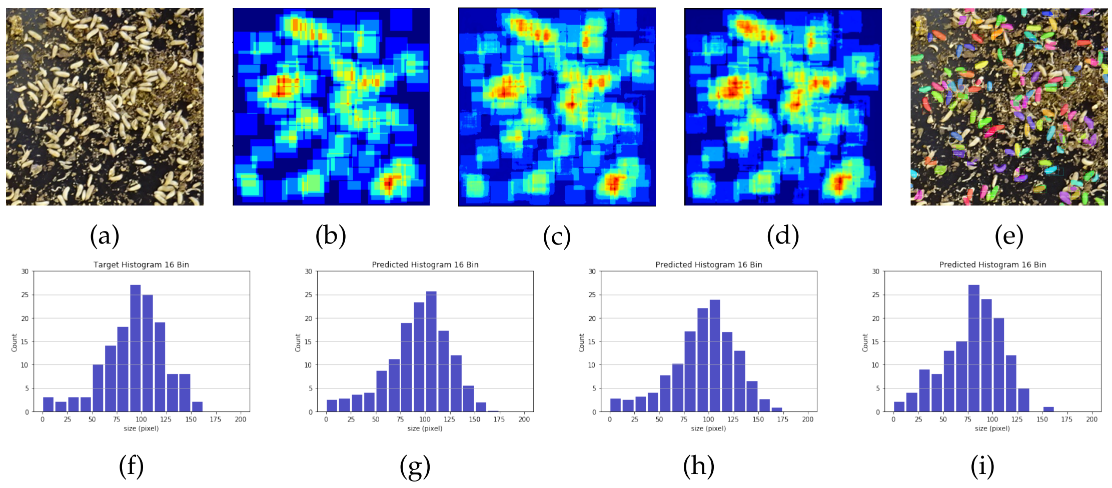


Figure 5.8.: Fly larvae 16-Bin Histogram Prediction: (a) input image (b) Target redundant count map (c) HistoNet countmap prediction (d) HistoNet-DSN countmap prediction (e) Mask R-CNN instance segmentation mask prediction(f) Target Histogram (g) HistoNet histogram Prediction, (h) HistoNet-DSN histogram prediction (i) Mask R-CNN histogram prediction

We also compare the results for a more fine-grained size histogram prediction in Fig. 5.8. The input image Fig. 5.8a consists of 142 larvaes. Similar to above results the object count map are nearly identical to ground truth and the 16-Bin histogram prediction for *HistoNet* and *HistoNet-DSN* is outperforming Mask R-CNN prediction. For this test image, our approach reduces the χ^2-distance between target and predicted histogram by nine to ten order in magnitude from 19.7 to 2.5 as compared to Mask R-CNN. Similarly the coorelation between the histogram also increases from 0.86 to 0.98 for our approach as opposed to the baseline.

Fig. 5.9 shows the deep supervision of HistoNet in action and demonstrates that *HistoNet DSN* further improves over *HistoNet* method. From early and middle layers of *HistoNet-DSN*, we predict 2-bin and 4-bin histogram as shown in Fig. 5.9f, 5.9g, 5.9h, 5.9i. These additional constraints assists our main objective to better predict 8-bin histogram as evident from Fig. 5.9d.

Table 5.2 summarizes performance comparison of Mask R-CNN [18], our proposed *HistoNet* and *HistoNet DSN* methods on *Fly Larvae* dataset for counting and 8 and 16 bin histogram prediction. For the *Fly Larvae* data set, our approach reduces the χ^2-distance for histogram prediction by more than 50% compared to the Mask R-CNN baseline. In addition, significant improvement in Kullback-Leibler divergence (*kld*) and weighted L1 difference between histograms (wt_{L1}) indicate that our method captures scale and shape

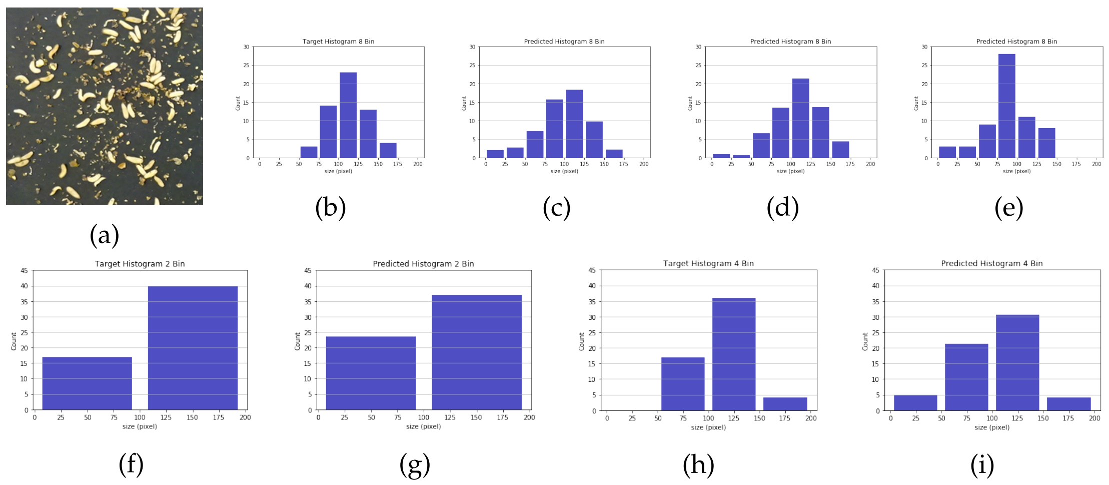


Figure 5.9.: Fly larvae 8-Bin Histogram Prediction: (a) input image (b) Target Histogram (c) HistoNet histogram Prediction, (d) HistoNet-DSN histogram prediction (e) Mask R-CNN histogram prediction (f) Target 2-Bin Histogram (g) HistoNet-DSN 2-Bin histogram prediction (h) Target 4-Bin Histogram (i) HistoNet-DSN 4-Bin histogram prediction

of the histograms much better than Mask R-CNN.

5.3.2. Synthetic Ellipse dataset

We get similar results for experiments on the synthetic ellipse dataset without any background clutter, but having much higher variance of size distributions, density, and object overlaps. Fig. 5.10 shows the performance comparison for an image consisting of small size thin ellipse objects as evident from ground truth size histogram in Fig. 5.10f. Similar to *Fly Larvae* dataset, our method outperforms explicit instance segmentation method for 8-bin size distribution histogram prediction, which due to object overlap predicts small

Method	MAE ↓	*kld* ↓	wt_{L1} ↓	*isec* ↑	χ^2 ↓	*corr* ↑	*bhatt* ↓
Mask R-CNN 8	7.84	0.64	4.31	0.72	16.37	0.77	0.25
HistoNet 8	2.38	0.25	2.72	0.81	6.57	0.91	0.16
HistoNet-DSN 8	**2.06**	**0.23**	**2.51**	**0.83**	**5.74**	**0.93**	**0.15**
Mask R-CNN 16	7.84	0.95	2.62	0.69	22.73	0.69	0.32
HistoNet 16	2.28	0.26	1.74	0.76	10.03	**0.86**	**0.21**
HistoNet-DSN 16	**1.99**	**0.25**	**1.70**	**0.77**	**9.8**	**0.86**	**0.21**

Table 5.2.: Result Summary FlyLarvae data set

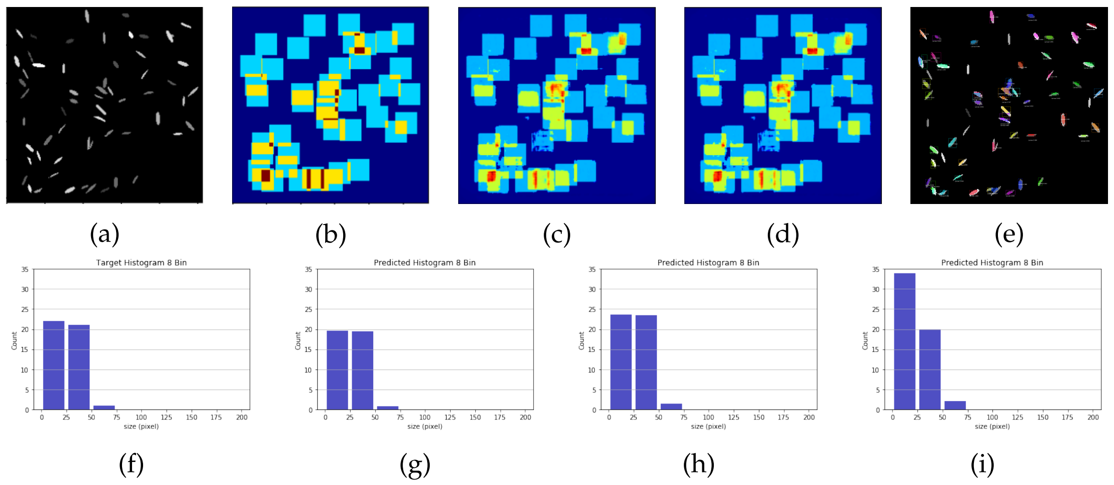


Figure 5.10.: Synthetic Ellipse 8-Bin Histogram Prediction: (a) input image (b) Target redundant count map (c) HistoNet countmap prediction (d) HistoNet-DSN countmap prediction (e) Mask R-CNN instance segmentation mask prediction (f) Target Histogram (g) HistoNet histogram Prediction, (h) HistoNet-DSN histogram prediction (i) Mask R-CNN histogram prediction

size masks and thus underpredicts the object sizes.

The next synthetic ellipse test image consists of ellipses of having larger variance in their sizes, our methods still better captures the shape and scale of the histogram as compared to Mask R-CNN method as shown in Fig. 5.11. As depicted in the Table 5.3, similar trends are observed on synthetic ellipse data set for χ^2-distance and correlation between histograms. Our method is able to handle high variance in object sizes and thus showing robustness on synthetic ellipse data set. We clearly show that even if the histogram is skewed, *HistoNet* is able to correctly predict its shape. We generalize to the number of objects as well as the size distribution.

Method	MAE $\downarrow$	*kld* $\downarrow$	wt_{L1} $\downarrow$	*isec* $\uparrow$	χ^2 $\downarrow$	*corr* $\uparrow$	*bhatt* $\downarrow$
Mask R-CNN 8	4.02	0.50	1.67	0.75	6.01	0.75	0.22
HistoNet 8	1.64	0.17	1.22	0.73	3.81	0.82	0.17
HistoNet-DSN 8	**1.2**	**0.13**	**1.17**	**0.78**	**3.36**	**0.83**	**0.16**
Mask R-CNN 16	4.02	1.12	1.13	0.67	10.81	0.64	0.32
HistoNet 16	1.45	0.47	0.87	0.68	7.19	0.71	0.27
HistoNet-DSN 16	**1.09**	**0.24**	**0.85**	**0.69**	**6.70**	**0.74**	**0.25**

Table 5.3.: Result Summary Synthetic Ellipse data set

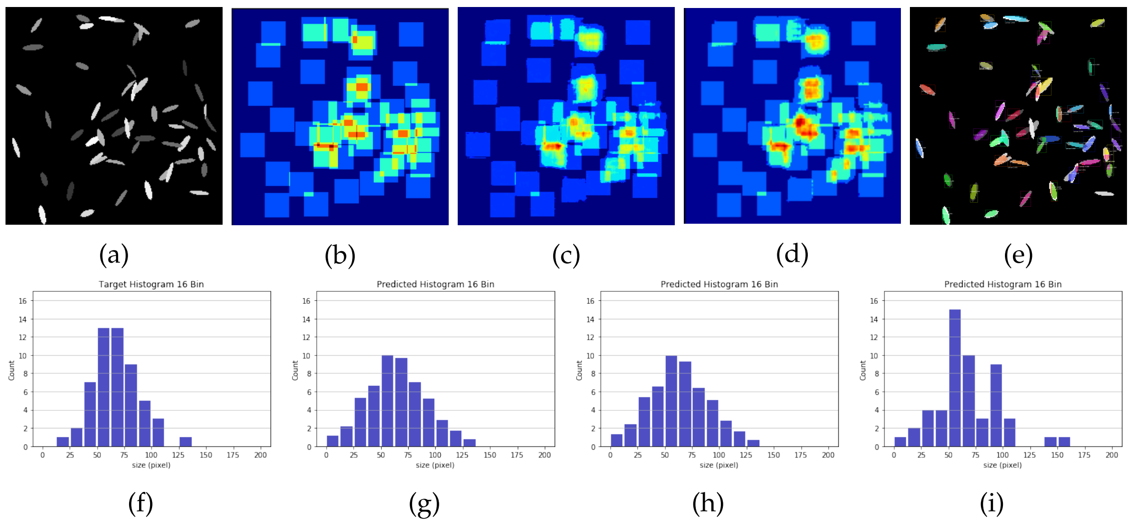


Figure 5.11.: Synthetic Ellipse 16-Bin Histogram Prediction: (a) input image (b) Target redundant count map (c) HistoNet countmap prediction (d) HistoNet-DSN countmap prediction (e) Mask R-CNN instance segmentation mask prediction(f) Target Histogram (g) HistoNet histogram Prediction, (h) HistoNet-DSN histogram prediction (i) Mask R-CNN histogram prediction

5.3.3. Breast Cancer Cell dataset

As discussed before, Cellularity score of a patch depends on the area of malignant cells. For this we first train our *HistoNet* model to predict the count map and size distribution histogram of malignant cells and using these as an intermediate result, we will predict the cellularity score. To directly predict cellularity score from a given image, we have affixed our network architecture with some additional layer as shown in Fig. 5.12. From the redundant count map prediction branch of *HistoNet*, we compute the total count and concatenate it with the size distribution histogram vector. Using this vector representing the statistical summary of malignant cells in the image, we pass it through 3 Fully connected layers, to predict Cellularity score. The whole network is end-to-end trainable. Due to lack of pixel-wise labeled data, we use a multi-part training schedule to train this network.

- **Stage 1:** Using the additional dataset in BreastPathQ challenge, for which nuclei information is provided, we created our target redundant count map. So, we trained only the lower branch of HistoNet to predict the redundant count map.
- **Stage 2:** We manually pixel-wise labeled some of the images from the additional dataset, to train the upper branch of HistoNet, to learn to predict 8-bin size histogram. On this small dataset, we train the whole HistoNet.
- **Stage 3**; We use the main dataset, which have images labeled with their cellualrity

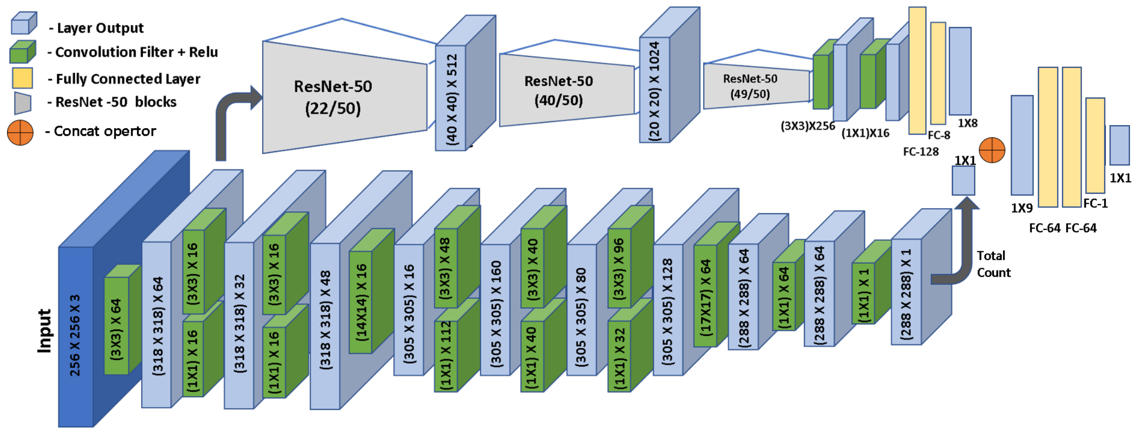


Figure 5.12.: Network Architecture for breast cancer cell dataset training

	P_K
CountCeption	0.56
HistoNet-[fc 128, fc 128]	0.69
HistoNet-[fc 18, fc 18]	0.76
HistoNet-[fc 32, fc 32]	0.79
HistoNet-[fc 64, fc 64]	0.83

Table 5.4.: Breast cancer cell cellularity score prediction

score, to train the remaining part of this architecture. During this training, we fix the weights learned from stage-2 for HistoNet.

	Parameter number ($\times 10^6$)
Mask R-CNN [18]	237.1
HistoNet	30.2
HistoNet DSN	36.5

Table 5.5.: Total number of parameters

By using the countmap and histogram prediction, we train our model to predict the cellularity score. We compare our method with CountCeption-based cellularity score prediction model. In which we only use count map information to predict the cellularity score. We also compare our method with different versions of our method having different size of Fully connected layer. Our method significantly improves cellularity score prediction over CountCeption-based method, which shows that to predict cellularity score only ma-

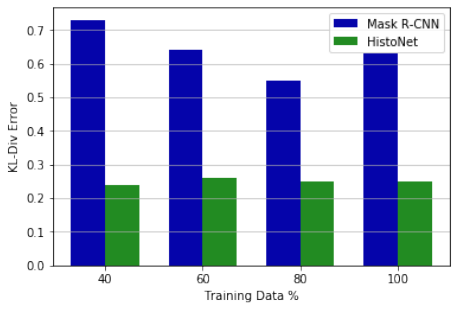


Figure 5.13.: Ablation Study: KL-Divergence Error

lignant cell count information is not sufficient, size distribution histogram is also required (Table 5.4). Among the variants of *HistoNet* we found that using the two fully connected layers of size 64 performs best as shown in Table. 5.4.

5.3.4. Ablation and Parameter Study

To evaluate how the performance of the method changes with the amount of training data, we perform an ablation study. Because of the scarcity of labeled data in biomedical applications, it is important to design methods that are lightweight and can be trained without resorting to large number of labeled examples. We increasingly reduce the amount of training data and evaluate the performance of Mask R-CNN and *HistoNet* on *Fly Larvae* dataset. As the amount of training data is reduced, Mask-RCNN results for *kld* error increase, while our approach obtains almost the same error when trained even with only 40% of the training data as shown in Fig. 5.13. Table 5.5 shows that our method requires drastically less learnable parameters, and hence more computationally efficient. Our approach uses 85% less parameters than Mask R-CNN [18], while being more accurate for counting and size distribution histogram prediction.

5.4. Result Summary

The result shows that our developed approaches *HistoNet* and *HistoNet-DSN* results in an overall improvement in the count and size distribution histogram prediction as compared to explicit instance segmentation method Mask R-CNN [18]. The summary of our results for fly larvae and synthetic ellipse dataset is shown in Fig. 5.14. As clear from Fig. 5.14a, 5.14d, that our methods *HistoNet* and *HistoNet-DSN* better predicts the object count than counting by detection method Mask R-CNN [18] as well as with the building block of our network architecture CountCeption [8]. This shows that while predicting size distribution histogram, the joint training of upper and lower branch of *HistoNet* also improves redundant count map prediction. Our methods reduces the χ^2-square distance by more than 50% and weighted L1 error $wL1$ by more than 30%, between predicted and target histogram. We show that our method is, computationally efficient due to its leaner architecture, and significantly more accurate than explicit instance segmentation method.

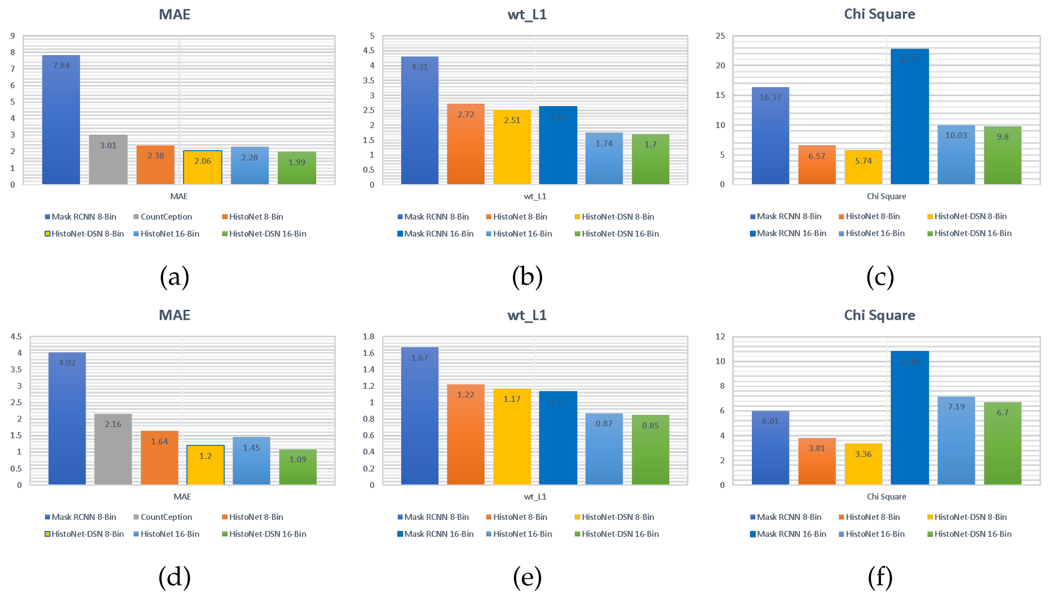


Figure 5.14.: Result Summary: (a) Fly larvae Mean absolute Count error (b) Fly larvae weighted L1 error (c) Fly Larvae chi square distance (d) Synthetic ellipse Mean absolute Count error (b) Synthetic ellipse weighted L1 error (c) Synthetic ellipse chi square distance

6. Conclusion

In this work, we developed two approaches with the aim of directly predicting statistical summary of objects in the form of object count and size distribution histogram in a given image. The motivation for our aim stems from its application in various scenarios and the inadequate nature of the currently available solution for our problem. Such solution i.e. explicit instance segmentation is indirect, computationally expensive and is prone to overlapping and occlusion problem.

We have presented *HistoNet*, a new deep learning approach that predicts object size distributions and total counts in cluttered scenes directly from an input image. We also present *HistoNet-DSN*, which improves the result by providing deep supervision at hidden layers to predict 2-bin and 4-bin histograms and introduces additional constraint along with the main objective. In *HistoNet-DSN*, as the complexity of task increases i.e. from 2-bin histogram to 8-bin histogram, the model complexity also increases. We are providing supervision signals at different stages of our network to calibrate the learned feature maps without incurring any additional labeling cost since 2-bin and 4-bin histogram can be extracted using 8-bin histogram.

Experimental evaluation on a new *Fly Larvae* dataset shows superior performance compared to explicit object instance segmentation method and fully convolutional network method CountCeption [8] predicting only object count. We verify with synthetic images having strongly varying object densities and object overlap, that our method can predict a diverse set of size histogram shapes. We show the application of our method on breast cancer cell dataset to predict cellularity score of malignant cells. However, our method has some limitations. Our method is not scale-invariant because it assumes constant distance from the camera to observation plane. Although count map can localize the objects, it can't provide the exact location of the objects.

In conclusion, we show that directly learning and predicting object size distributions, without a detour via explicit pixel-accurate instance segmentation, significantly improves performance. In addition, we save 85% of model parameters, which leads to a much leaner architecture that can be trained with fewer annotations. We believe that the value of direct histogram prediction goes beyond our specific use cases. In future work, we will investigate its potential to speed up state-of-the-art object detectors by modelling spatial priors on anchor box distributions, which is mostly done in a greedy fashion nowadays. We will also generalize our method for multi-class size histogram and count map prediction.

Appendix

A. Experimental Setup and Hyperparameters

This appendix contains information about the pre-processing of datasets and the settings for different experiments.

A.1. Data pre-processing

Fly larvae dataset: From the images collected during the data acquisition process, we sampled the image patches of size 256×256. Using the pixel-wise labels of images, we extracted object center, bounding box and mask area using publicly available Scikit-Learn library [37]. From this, we created redundant count map and size distribution histogram. The range of larvae instances size is 0-200 pixels. Using this, we created MS-COCO [30] style dataset for training Mask R-CNN [18]. We randomly distributed the image dataset in training(60%), validation(20%) and test(20%) set. We augmented our training set by applying transformations, such as horizontal flip, vertical flip, random noise addition and random contrast variation.

Synthetic ellipse dataset: We created 3000 thin ellipses of different size and shapes. While the size range of ellipses is same as *Fly Larvae* dataset, the variance in ellipse instance sizes is larger than fly larvaes. The distribution among train, validation and test set is the same as *Fly Larvae* dataset. Data augmentation is not applied for this dataset.

Breast cancer cell dataset: For predicting cancer cell cellularity score, three datasets were used. First dataset (*BreastCancer-1*) has malignant cell redundant count maps as ground truth labels which were used for training the lower branch of *HistoNet*. Second dataset (*BreastCancer-2*) has 8-bin size histogram along with redundant count map as ground truth label which is used for training *HistoNet*'s upper branch and fine tuning lower branch. Third dataset (*BreastCancer-3*) contains cellularity score as ground truth label to train the remaining part of modified *HistoNet* architecture (Fig. 5.12). The range of malignant cell size is 0-500 pixels. We created 8-Bin size histogram and redundant count map from 18 manually pixel-wise labeled images. Table A.1 provides the distribution of images for different datasets, the number of images in training data is after the image augmentation process.

Acknowledgment: The BreastPathQ Challenge data used in this research was acquired from Sunnybrook Health Sciences Centre with funding from the Canadian Cancer Society

and was made available for the BreastPathQ challenge, sponsored by the SPIE, NCI/NIH, AAPM, Sunnybrook Research Institute.

Dataset	Train Set	Val Set	Test Set
Fly Larvae	1152	96	96
Synthetic Ellipse	1800	600	600
BreastCancer-1	655	195	216
BreastCancer-2	396	26	32
BreastCancer-3	1455	479	479

Table A.1.: Number of images in datasets

A.2. Experiment settings

We implemented our method *HistoNet* and *HistoNet-DSN* using Lasagne-Theano framework [50]. For implementing state-of-the-art instance segmenetation method Mask R-CNN [18] we used Pytorch framework [36]. We use the Adam optimizer [24] for training our model and a batch size of four images. Our network weights are initialized using Xavier initialization [15] and we trained our network for 100 epochs on Nvidia GeForce GTX 1080 Ti GPU. Table A.2, A.3 summarizes the hyperparameters used for experiments on *Fly Larvae* and synthetic ellipse dataset. As mentioned earlier for training modified *HistoNet* model for predicting cancer cell cellularity score, 3 types of datasets are used. Hyperparameters for these experiments on breast cancer cell datasets are summarzied in Table A.4. Additional hyperparameters for Mask R-CNN experiments are given in Table A.5.

Method	Learning rate	Weight decay
Mask R-CNN	1e-3	1e-4
CountCeption	5e-3	1e-5
HistoNet-8	8e-4	1e-4
HistoNet-DSN 8	1e-3	1e-5
HistoNet-16	8e-4	1e-5
HistoNet-DSN 16	3e-3	1e-4

Table A.2.: Hyperparameters for experiments on Fly larvae dataset

Method	Learning rate	Weight decay
Mask R-CNN	1e-3	1e-4
CountCeption	5e-5	1e-6
HistoNet-8	3e-4	1e-5
HistoNet-DSN 8	8e-4	1e-4
HistoNet-16	5e-3	1e-4
HistoNet-DSN 16	3e-3	1e-5

Table A.3.: Hyperparameters For experiments on Synthetic Ellipse dataset

Method	Learning rate	Weight decay
BreastCancer-1	5e-5	1e-5
BreastCancer-2	1e-4	1e-5
BreastCancer-3	5e-4	1e-5

Table A.4.: Hyperparameters For experiments on Breast cancer cell dataset

Hyperparameter	Value
Backbone strides FPN Pyramid	[4, 8, 16, 32, 64]
RPN anchor scales	[4,8, 16, 24, 32]
RPN anchor ratios	[0.2, 0.5, 1, 2, 5]
RPN anchor stride	1
RPN NMS threshold	0.6
ROI align pool size	7×7
Mask Shape	28×28
Minimum confidence for ROI	0.7
NMS threshold for detection	0.3
Maximum detected instances	200
Learning momentum	0.9

Table A.5.: Hyperparameters settings for Mask R-CNN

Bibliography

[1] A neural-based crowd estimation by hybrid global learning algorithm.

[2] Automatic cellularity assessment from post-treated breast surgical specimens. *Cytometry. Part A*, 91(11):1078–1087, 2017.

[3] Narendra Ahuja and Sinisa Todorovic. Extracting texels in 2.1d natural textures. pages 1–8, 11 2007.

[4] M. Bai and R. Urtasun. Deep Watershed Transform for Instance Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.

[5] Brilliant. Convolutional neural network. brilliant.org. retrieved 18:50, june 13, 2019,. *https://brilliant.org/wiki/convolutional-neural-network/*, 2019.

[6] M. Butenuth and C. Heipke. Network snakes: graph-based object delineation with active contour models. *Machine Vision and Applications*, 23(1):91–109, 2012.

[7] S.-Y. Cho and T.W.S. Chow. A neural-based crowd estimation by hybrid global learning algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 29(4):535–541, 1999.

[8] Joseph Paul Cohen, Henry Z. Lo, and Yoshua Bengio. Count-ception: Counting by fully convolutional redundant counting. *ICCV Workshops*, abs/1703.08710, 2017.

[9] George Cybenko. Approximation by superpositions of a sigmoidal function. *MCSS*, 2:303–314, 1989.

[10] S. Diener, N.M.S. Solano, F.R. Gutiérrez, C. Zurbrügg, and K. Tockner. Biological treatment of municipal organic waste using black soldier fly larvae. *Waste and Biomass Valorization*, 2(4):357–363, 2011.

[11] L. Fiaschi, U. Koethe, R. Nair, and F. A. Hamprecht. Learning to count with regression forest and structured labels. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2685–2688, Nov 2012.

[12] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[13] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[14] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[15] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, 2010.

[16] M. Gold, J.K. Tomberlin, S. Diener S., C. Zurbrügg, and A. Mathys. Decomposition of biowaste macronutrients, microbes, and chemicals in black soldier fly larval treatment: A review. *Waste and Biomass Valorization*, 82:302–318, 2018.

[17] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3):300–316, 2008.

[18] K. He, G. Gkioxari, P. Dollár, and R.B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[20] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[21] T. Kailath. The divergence and bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, 15(1):52–60, 1967.

[22] A. Karpathy. Cs231n convolutional neural networks for visual recognition. *http://cs231n.github.io/neural-networks-1/*, 2017.

[23] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.

[24] D.P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

[25] Dan Kong, Doug Gray, and Hai Tao. A viewpoint invariant approach for crowd counting. In *18th International Conference on Pattern Recognition*, volume 3, pages 1187–1190, 2006.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc.

[27] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[28] Chen-Yu Lee, Saining Xie, Patrick W. Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015*, 2015.

[29] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1324–1332. Curran Associates, Inc., 2010.

[30] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[31] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[32] Aparecido Marana, Sergio Velastin, Luciano da F. Costa, and Roberto Lotufo. Estimation of crowd density using image processing. pages 11/1 – 11/8, 04 1997.

[33] mathworks. https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html. 2019.

[34] medium.com. https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5. 2019.

[35] F. Nielsen and R. Nock. On the chi square and higher-order chi distances for approximating f-divergences. *IEEE Signal Processing Letters*, 21(1):10–13, 2014.

[36] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[38] Vincent Rabaud and Serge Belongie. Counting crowded moving objects. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 705–711, Washington, DC, USA, 2006. IEEE Computer Society.

[39] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[40] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[41] A. Rodriguez and J.D. Wegner. Counting the uncountable: deep semantic density estimation from Space. In *German Conference on Pattern Recognition*, LNCS 11269, pages 351–362, 2018.

[42] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI, Lecture Notes in Computer Science*, volume 9351, pages 234–241. Springer International Publishing, 2015.

[43] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, pages 65–386, 1958.

[44] J.A. Sethian. *Level Set Methods*. Cambridge University Press, The Edinburgh Building, Cambridge CB2 2RU, UK, 1 edition, 1996.

[45] M. Sezgin and B. Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146 – 165 – 20, 2004.

[46] J. Shotton, J. Winn, C. Rother, and A. Criminisi. TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-class Object Recognition and Segmentation. In *European Conference on Computer Vision*, pages 1–15, 2006.

[47] WARREN D. SMITH, ROBERT C. DUTTON, and N. TY SMITH. A measure of association for assessing prediction accuracy that is a generalization of non-parametric roc area. *Statistics in Medicine*, 15(11):1199–1215.

[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[49] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[50] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[51] W. Xie, J. A. Noble, and A. Zisserman. Microscopy cell counting with fully convolutional regression networks. In *MICCAI 1st Workshop on Deep Learning in Medical Image Analysis*, 2015.

[52] Yuanpu Xie, Fuyong Xing, Xiangfei Kong, Hai Su, and Lin Yang. Beyond classification: Structured regression for robust cell detection using convolutional neural network. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 358–365, Cham, 2015. Springer International Publishing.

[53] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017.