SAU-Net: A Unified Network for Cell Counting in 2D and 3D Microscopy Images

Yue Guo, Oleh Krupa, Jason Stein, Guorong Wu, and Ashok Krishnamurthy

Abstract—Image-based cell counting is a fundamental yet challenging task with wide applications in biological research. In this paper, we propose a novel unified deep network framework designed to solve this problem for various cell types in both 2D and 3D images. Specifically, we first propose SAU-Net for cell counting by extending the segmentation network U-Net with a Self-Attention module. Second, we design an extension of Batch Normalization (BN) to facilitate the training process for small datasets. In addition, a new 3D benchmark dataset based on the existing mouse blastocyst (MBC) dataset is developed and released to the community. Our SAU-Net achieves state-of-the-art results on four benchmark 2D datasets - synthetic fluorescence microscopy (VGG) dataset, Modified Bone Marrow (MBM) dataset, human subcutaneous adipose tissue (ADI) dataset, and Dublin Cell Counting (DCC) dataset, and the new 3D dataset, MBC. The BN extension is validated using extensive experiments on the 2D datasets, since GPU memory constraints preclude use of 3D datasets. The source code is available at https://github.com/mzlr/sau-net.

Index Terms—Cell counting, batch normalization, deep learning, neural networks

1 Introduction

Image-based cell counting is a powerful technique for computational analysis in various biological studies [1], [2] because the quantification of a specific cell type can provide valuable insights into the underlying cellular mechanism. Cell biological applications using optogenetic tools [3], for example, often require the count of opsin expressing cells for cancer biology and neurobiology. Despite previous successes focusing on specific cell images [4], [5], [6], universal image-based cell counting remains a challenging task due to the large variations in the cell type, staining technique and imaging modality. Additionally, most recent image-based cell counting studies focus on 2D images [7], [8], [9] and the lack of reliable 3D cell quantification methods hinders the impact brought about by the rapid advancement in 3D microscopy [10]. Therefore, it is of great importance to provide a unified framework to perform cell-counting for various cell types imaged with different techniques.

Due to the large number of cells in images, complete annotations for each cell are usually costly. Instead, dot-annotations are widely used in public benchmarks for image-based cell counting [4], [8], [9], [11]. As shown in Figure 1, each cell is represented by a single pixel at its center, and these dot-annotations are considered density maps. Given this labeling, recent methods [4], [6], [9], [12] approach the cell counting task by learning a mapping from an input image to the corresponding density map, and the cell count can be inferred by integration of the density map. Inspired by the recent success of U-Net based methods for image segmentation [13], [14], [15], we choose U-Net as the base mapping model in both the 2D and 3D cases. One of the main obstacles in this learning process is that the density maps are dot-annotated and thus extremely sparse, rendering the model difficult to train. Previous methods [4], [6] remedy this problem by applying a Gaussian kernel at each dot annotation to blur the density map, which also maintains the same cell count. Building upon this technique, we propose to use Self-Attention module [16] in conjunction with U-Net to learn long-range, non-local dependencies in images, forcing the model to "focus" on the foreground instead of background pixels. The proposed structure, named SAU-Net, outperforms U-Net in both 2D and 3D scenarios. In addition, given the lack of a 3D public benchmark, we introduce a 3D cell counting dataset based on the work in [17]. This dataset was originally released for inter-cellular communication research, and largely overlooked by the cell counting research community. We have modified and release that dataset for 3D evaluations.

Another challenge with Deep Learning methods is hyperparameter optimization, such as for Batch Normalization (BN) [18] when training small datasets. BN has been widely adopted in various state-of-the-art deep learning models, e.g., ResNet [19], DenseNet [20], and EfficientNet [21]. Despite its successes, it remains challenging to analytically determine the optimal value of the BN hyperparameter moving average momentum. A commonly adopted strategy for hyperparameter optimization is to perform a grid-search or random search [22], but this process is computationally expensive since it requires repeated trials. Instead, BN moving average momentum is often empirically set to 0.9 [19], [20] or 0.99 [21], which is also the default value in popular deep

- *Y. Guo and A. Krishnamurthy are with the Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599. E-mail: {yueguo, ashok}@cs.unc.edu.*
- *O. Krupa is with the Joint Department of Biomedical Engineering, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599. E-mail: ok37@email.unc.edu.*
- *J. Stein is with the Department of Genetics, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599. E-mail: jason_stein@med.unc.edu.*
- *G. Wu is with the Department of Psychiatry, University of North Carolina at Chapel Hill, Chapel Hill, NC, 27599. E-mail: guorong_wu@med.unc.edu.*

Manuscript received Sept. 15, 2020. (Corresponding author: Yue Guo.)

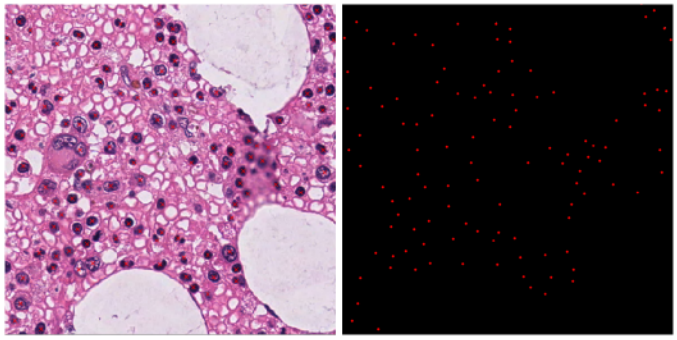

Fig. 1: A sample image from Modified Bone Marrow (MBM) dataset [11] with dot annotations shown as red cross overlays (left image) and the corresponding dot annotations used in training (right image).

learning libraries, e.g., TensorFlow[1], MXNet[2], or PyTorch[3]. These recommended empirical values are usually based on large datasets, like CIFAR [23] or ImageNet [24], which have thousands, if not millions, of images, and it does not guarantee these values work for small datasets, a common scenario in cell counting research. In our experiments, we find these values often lead to sub-optimal performance with smaller datasets. Rather than performing the costly manual search, we propose a simple extension of BN: only during inference, we re-estimate the statistics of the training data and use these values for normalization rather than the moving averages. Our proposed extension of BN has a negligible impact on computation since it does not affect training process, meaning that we can easily take advantage of standard BN trained models.

To validate the effectiveness of SAU-Net, we test on four public 2D benchmarks for cell counting, plus the newly introduced 3D benchmark. In 2D cases, state-of-the-art performance is obtained in the Modified Bone Marrow (MBM) dataset [11], human subcutaneous adipose tissue (ADI) dataset [9] and Dublin Cell Counting (DCC) datasets [8]. On the synthetic fluorescence microscopy (VGG) dataset [4], our result is on par with the leading method. Similarly, for the 3D datasets [17] on mouse blastocyst (MBC), SAU-Net outperforms other popular 3D methods. Moreover, results on four 2D datasets also sustain our hypothesis that our BN extension expedites the training process. We argue this should also hold in 3D cases, although we are unable to empirically verify this conjecture due to memory limitation in our GPUs, which opens up a possibility for further exploration.

In summary, our contributions are three-fold:

- We propose a novel deep architecture, SAU-Net, to incorporate U-Net [13] with Self-Attention module [16] for cell counting and provide a unified framework in both 2D and 3D cases. An extension of Batch Normalization is also developed to facilitate the training in small datasets.
- SAU-Net outperforms state-of-the-art methods in four out of five cell counting benchmarks and achieves competitive results in the other synthetic benchmark, highlighting the universal nature of the proposed method.
- The source code is available to the research community at https://github.com/mzlr/sau-net, aimed at stimulating further investigations on this topic. The 3D benchmark used in the experiment based on [17] is also released to promote more research for emerging 3D microscopy.

The paper serves as an extended version of a preliminary work in [25]. Since then, we have made two key improvements: 1) we expand the original architecture from 2D use case to 3D. Along with the release of a new 3D benchmark modified from [17], a unified framework for image-based cell counting is presented; 2) To address computational overhead in the online version of BN in [25], we propose a BN extension as an efficient implementation of online BN and conduct more numerical analysis to offer an empirical explanation for the performance gain of the proposed extension.

The rest of the paper is organized as follows: We will first review the related work for cell counting in Section 2 and then present the proposed method in Section 3. Next, Section 4 will briefly describe the benchmarks used in this study, followed by Section 5, which will conduct an ablation study on each proposed component and compare our results with the state-of-the-art methods for each benchmark. Additional analysis for the results is provided in Section 6. Finally, Section 7 will conclude the paper with a discussion for future work.

2 Related Work

There are generally two categories of cell counting methods: detection based [26], [27] and regression based [4], [6], [8], [9], [12]. The former approaches use a detector to localize each cell, and the cell count can be obtained by counting the detected cells. Cell detection, however, remains a challenging task due to occlusion, shape variation, etc., and training such a model typically requires whole-cell annotations for individual cells, which is time-consuming given the high cell density in images. On the other hand, regression based methods require simple dot annotation for training, and cell counts can be obtained via integration on the predicted density maps. Therefore, this paper focuses on the regression-based method.

2.1 2D Cases

Earlier regression based methods learned a mapping from dense local image features to a density map. Lempitsky and Zisserman [4] first used linear regression with dense SIFT features to predict the density map. Later, a regression forest was proposed in [5] to replace the linear regression for better density map estimation. Arteta et al. [28] extended the pipeline by a local feature vocabulary and ridge regression in an interactive fashion.

Recent methods favor Deep Neural Network due to its versatility in various research areas, e.g., computer vision [29], [30], medical imaging [15], [31], natural language

1. https://www.tensorflow.org/api_docs/python/tf/keras/layers/BatchNormalization
2. https://mxnet.apache.org/versions/master/api/python/docs/api/npx/generated/mxnet.npx.batch_norm.html
3. https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html#torch.nn.BatchNorm2d

processing [16]. Xie et al. [6] relied on fully convolutional regression networks [32] to estimate the density map filtered by a Gaussian kernel. Cohen et al. [9] followed this direction but constructed the density map based on the receptive fields of the networks. They first filtered the dot annotations with a square kernel and then summed the value in a sliding window fashion, with each window corresponding to a receptive field in the network. This redundant counting improved the counting accuracy but could over-fit to the background in some cases. This is because the network no longer obtained the cell count by identifying individual cells in a density map, and it could simply average the regression prediction when large areas of background exist in images. Alternatively, Xue et al. [7] divided the input into multiple sub-images and tested multiple neural networks to map sub-images to a single scalar, namely the cell count. Similarly, [8] used a pre-trained network to learn the same mapping for cross-domain counting. This direct mapping, however, is difficult to learn since the mapping from an image to a number for cell count is highly non-linear and the model tends to focus on artifacts instead of cells in images [9].

2.2 3D Cases

Fewer efforts have been made to tackle the 3D image-based cell counting task. Due to the lack of public benchmarks, most data-driven approaches, including the family of modern deep learning methods, have not been explored in the context of 3D image-based cell counting. Most algorithms first perform cell segmentation via thresholding or watershed and then run connected component analysis to obtain the cell count. We refer readers to [33] for a detailed review of related models and datasets. Notably, MINS [34] improves the generic pipeline by refining segmentation results via a Geodesic model and applying RANSAC and ellipse fitting for outlier removal and is widely used in the community. Compared to modern end-to-end deep neural networks, this multi-stage model still lacks the representational power and is likely to yield suboptimal results, as indicated by our experiment results in Section 5.

To address the limitations mentioned above, we propose SAU-Net, a U-Net [13] with a Self-Attention module [16] tailored for counting cells in both 2D and 3D image data. We also extend the Batch Normalization [18] to expedite the training process. It exhibits outstanding performance across various datasets and avoids the background over-fitting problem. As part of this study, a 3D dataset based on [17] is released to spur further investigations. Note that there is a similarity between our work and an attention based U-Net for image segmenting in [14], but our model differs in the way we incorporate the attention module and in the scope of data dimension; also our application is different.

3 Methods

In this section we will first introduce the overview of our regression based cell counting method and then present our novel contributions.

3.1 Overview

Following the idea from [4], the goal of our regression based cell counting method is to learn a mapping F from an image I to a density map D, denoted as

$$F : I \rightarrow D \quad (I, D \in R^{H \times W} \text{ or } I, D \in R^{H \times W \times D}), \quad (1)$$

where I and D can be either 2D or 3D images of the same shape.

The density map D is obtained by convolution with a Gaussian kernel on the dot-annotated binary labels L, where the center of a cell is set to 1 otherwise 0. This process is necessary since L is too sparse for learning F. For simplicity, we choose a fixed bandwidth for all the Gaussian kernels. Finally, the cell count can be computed by integration over the density map D. The method overview is shown in Figure 2 in the 2D case, and the 3D case is a generalization of this approach.

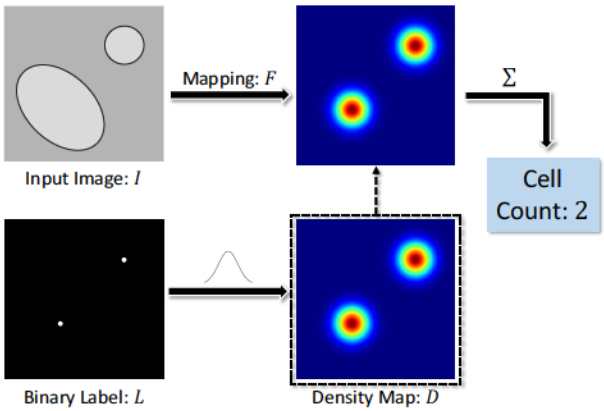

Fig. 2: Overview of our regression based cell counting method with a 2D image containing two cells. The counting problem is addressed in two steps: 1) learn a mapping F from the input image I to the density map D and 2) integrate D to predict the final count. We apply convolution with a Gaussian kernel on the binary labels L to obtain D for the learning process.

3.2 SAU-Net

We propose a novel model, called SAU-Net, to represent the mapping F in Equation 1. SAU-Net incorporates the Self-Attention module [16] on top of a U-Net architecture [13], whose encoding-decoding structure proves suitable for various medical imaging tasks [13], [14], [15]. We expand both the underlying U-Net and the Self-Attention module so that SAU-Net is capable of handling both 2D and 3D images. In addition, we add a Batch Normalization [18] layer after every convolution and deconvolution layer in U-Net. The overall structure of SAU-Net is illustrated in Figure 3.

Self-Attention can be viewed as a non-local weighted averaging operation in deep neural networks [35]. Mathematically, we have

$$Self\text{-}Attention(X) = softmax(f(X) \otimes g(X)^T) \otimes h(X), \quad (2)$$

where $X \in R^{H \times W \times C}$ or $R^{H \times W \times D \times C}$ are the activations from the previous layers and used as inputs to the Self-Attention module. H, W, D and C represent the height,

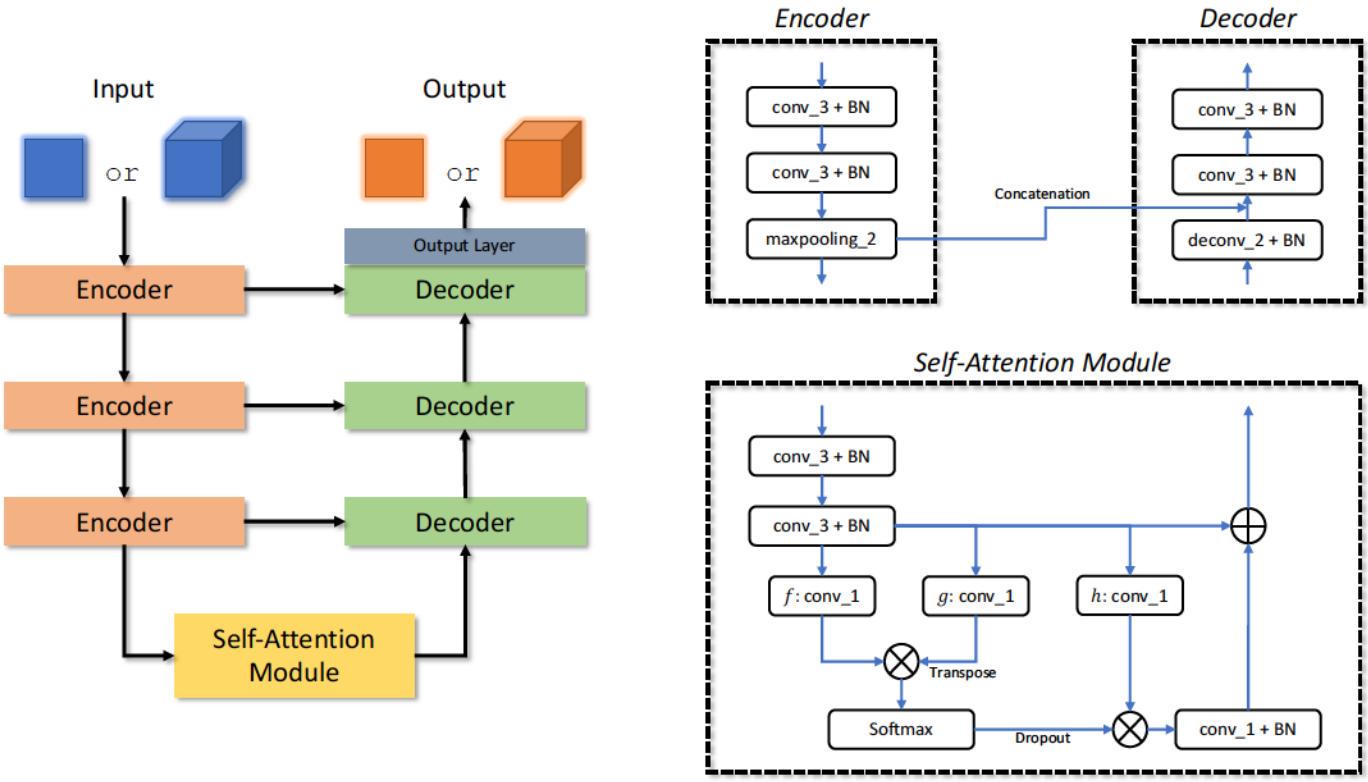

Fig. 3: The overall structure of SAU-Net (left) with details of each block (right). SAU-Net seamlessly unifies 2D and 3D image-based cell counting. The number after `conv`, `maxpooling`, `deconv` indicates the filter size, e.g., 3 for 3×3 or $3 \times 3 \times 3$ depending on the dimension of the input images. The output layer is implemented as `conv_1`. All the convolutions and deconvolutions use ReLU nonlinearity except the linear embeddings, f, g and h in the Self-Attention module. $\otimes$ denotes matrix multiplication.

width, depth and channel number of the activations, respectively. f, g and h are linear embeddings, implemented as 1×1 or $1 \times 1 \times 1$ convolution, and proper reshaping is performed for matrix multiplication, i.e., *reshape* : $R^{H \times W \times C} \rightarrow R^{(HW) \times C}$ or $R^{H \times W \times D \times C} \rightarrow R^{(HWD) \times C}$. *softmax* here works as a scaling function so that the weights for the averaging sum to 1, and $\otimes$ denotes matrix multiplication.

Self-Attention computes weights based on feature relationships, i.e., $softmax(f(X) \otimes g(X)^T)$ in Equation 2, across the whole image region, whereas conventional convolutional layers can only process local information, e.g., 3×3 convolution. By combining the Self-Attention module with convolutional layers in U-Net, SAU-Net can enjoy a richer hierarchy that can learn the mapping based on both local and global relationship. Given the encoding and decoding structure of U-Net, it is generally believed the deeper convolutional layer of U-Net has the more abundant feature information. Therefore, we choose to incorporate the Self-Attention module at the end of U-Net's encoding path for more accurate feature relationships.

3.3 Batch Normalization Extension

Batch Normalization [18], as outlined in Algorithm 1 below, is a technique to accelerate the training of deep neural networks, which introduces additional network layers to normalize the inputs for subsequent layers, effectively stabilizing the distributions of layer inputs. BN applies normalization on each sample in a batch[4] B (Line 5-7), which is randomly subsampled from the training set T. For normalization during inference, BN tracks the moving averages of the mean $\hat{\mu}$ and variance $\hat{\sigma}$, using the momentum m (Algorithm 1, Lines 8-9). During inference, batch normalization uses the moving average estimates as an approximation of the population statistics, since tracking moving averages is compatible with stochastic optimization (Algorithm 1, Line 12-14). This relies on a hyperparameter, the moving average momentum m, to control the estimation.

When dealing with small datasets, it is natural to assume the training process would require fewer iterations because from an optimization standpoint, the loss function is an average over the training set [36], and we have less data to perform the averaging operation. In practice, however, we have observed the training still needs to iterate thousands of steps, as in other much larger datasets. This motivated us to investigate the reason behind this. We argue that the empirical values of BN moving average momentum from larger datasets are not suitable for small datasets, and it would be computationally prohibitive to perform a grid search for the optimal value of m.

Rather than using the estimated moving averages, we are proposing recalculating the mean and standard deviation derived from the entire training set to obtain more accurate

4. Some publications also refer to it as minibatch.

Algorithm 1: A layer of Batch Normalization [18]

Input: Training set of size t with activations x from prior layers: $T = \{x_1, \ldots, x_t\}$; test set of size t': $T' = \{x'_1, \ldots, x'_t\}$; trainable affine transformation parameters: γ, β; constant for numerical stability: ϵ; current batch statistics: μ_B, σ_B; moving average estimation: $\hat{\mu}, \hat{\sigma}$; moving average momentum: m.

Output: Normalized output for next layers: y'

Training:

1 **repeat**

2 randomly sample a batch B with size b from T

/* calculate batch statistics */

3 $\mu_B \leftarrow \frac{1}{b}\sum_{x_i \in B}$

4 $\sigma_B \leftarrow \sqrt{\frac{1}{b}\sum_{x_i \in B}(x_i - \mu_B)^2}$

/* normalize using batch statistics */

5 **for** $x_i \in B$ **do**

6 $y_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$

7 **end**

/* update moving averages */

8 $\hat{\mu} \leftarrow m\hat{\mu} + (1-m)\mu_B$

9 $\hat{\sigma} \leftarrow m\hat{\sigma} + (1-m)\sigma_B$

10 optimize γ, β via back-propagation

11 **until** *number of iteration reaches* $n_{iteration}$

Inference:

/* normalize using moving averages */

12 **for** $x'_i \in T'$ **do**

13 $y'_i \leftarrow \gamma \frac{x'_i - \hat{\mu}}{\sqrt{\hat{\sigma}^2 + \epsilon}} + \beta$

14 **end**

Algorithm 2: A layer of Proposed BN Extension

Input: Training set of size t with activations x from prior layers: $T = \{x_1, \ldots, x_t\}$; test set of size t': $T' = \{x'_1, \ldots, x'_t\}$; trainable affine transformation parameters: γ, β; constant for numerical stability: ϵ; current batch statistics: μ_B, σ_B.

Output: Normalized output for next layers: y'

Training:

1 **repeat**

2 randomly sample a batch B with size b from T

/* calculate batch statistics */

3 $\mu_B \leftarrow \frac{1}{b}\sum_{x_i \in B}$

4 $\sigma_B \leftarrow \sqrt{\frac{1}{b}\sum_{x_i \in B}(x_i - \mu_B)^2}$

/* normalize using batch statistics */

5 **for** $x_i \in B$ **do**

6 $y_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$

7 **end**

8 optimize γ, β via back-propagation

9 **until** *number of iteration reaches* $n_{iteration}$

Post-Training:

/* calculate population statistics */

10 $\mu_T \leftarrow \frac{1}{t}\sum_{x_i \in T} x_i$

11 $\sigma_T \leftarrow \sqrt{\frac{1}{t}\sum_{x_i \in T}(x_i - \mu_T)^2}$

Inference:

/* normalize using population statistics */

12 **for** $x'_i \in T'$ **do**

13 $y'_i \leftarrow \gamma \frac{x'_i - \mu_T}{\sqrt{\sigma_T^2 + \epsilon}} + \beta$

14 **end**

population statistics as a post-training, pre-inference step. Algorithm 2 implements our proposed BN extension. In Line 10-11 of Algorithm 2, we add a one-time post-training step to calculate population statistics, denoted μ_T and σ_T, from the training set T, and then, in Line 12-14, normalize test data using population statistics. Our empirical results show that our proposed extension to BN enables faster convergence. As a result, our method effectively eliminates the hyperparameter of the BN moving averaging momentum m.

Notice that our approach does not affect the BN training process since the moving averages are only used for inference. This property makes it ideal to apply our extension on existing BN-trained models for immediate potential improvement. Our extension can be easily implemented in popular deep learning libraries, as shown in our released code. Compared to Online Batch Normalization (OBN) in our early work [25], we have made a key improvement in terms of efficiency: OBN incurs computational overhead because it estimates the population statistics using the testing sample in addition to the training data. This means it has to perform the calculation each time when making a prediction. Since a single sample has a negligible effect on the population statistics estimation, we compute the population statistics once on the training data and use these values during inference.

3.4 Loss Functions

We use a pixel-wise $L2$ loss for our model:

$$\min_{\Omega} \frac{1}{N}\sum_i (S_i - D_i)^2, \tag{3}$$

where Ω represents the trainable parameters set in SAU-Net, and S is the mapping, $F(I)$ in Equation 1, namely the output from the SAU-Net. The subscript i denotes individual pixel or voxel, and N is the total pixel or voxel number. During training, the actual loss value can be very small due to the Gaussian filtering, which renders the network difficult to train. Following [6], we solve this by scaling the loss with a large constant value, e.g., 100. This technique is applied throughout all the experiments.

We also tried to add a loss function to minimize the cell counts explicitly but this change only learned the artifacts in the images. We believe this is caused the ambiguity in the loss function since the loss function only requires the model to predict the overall cell count instead of detecting actual cells; as a result, there are numerous mappings from

the input image to yield a correct cell count. A similar result was observed in [9].

4 DATASETS

We evaluate the proposed method on four public 2D benchmarks: synthetic fluorescence microscopy (VGG) dataset [4], Modified Bone Marrow (MBM) dataset [11], human subcutaneous adipose tissue (ADI) dataset [9], and Dublin Cell Counting (DCC) dataset [8]. Additionally, a 3D mouse blastocyst dataset (MBC) based on [17] is used to experiment in 3D cases. We also provide results on a 3D light-sheet dataset for mouse cortex from recent work [37]. For conciseness, we defer to Appendix A the details regarding this dataset.

- **VGG**: Lempitsky and Zisserman [4] used the method in [38] to create VGG dataset, which simulated bacterial cells from fluorescence-light microscopy at various focal distances.
- **MBM**: Cohen et al. [9] introduced the Modified Bone Marrow (MBM) dataset based on the dataset first released by Kainz et al. [11]. This dataset contains real images of human bone marrow with various cell types stained blue.
- **ADI**: Human subcutaneous adipose tissue (ADI) dataset [9] is constructed from the Genotype Tissue Expression Consortium [39] with densely packed adipocyte cells.
- **DCC**: Marsden et al. [8] built Dublin Cell Counting (DCC) dataset to represent a wide range of cells, including embryonic mice stem cells, human lung adenocarcinoma, human monocytes, etc. The image size ranges from 306×322 to 798×788, intended to increase the variation of the dataset.
- **MBC**: Saiz et al. [17] first constructed this mouse blastocyst dataset (MBC) to model cellular proliferation during embryonic development for intercellular communication study. Since their work focused on biological impact from the cell count, the cell locations sometimes were approximated by the nearest neighbor during the manual labeling, rendering them unfit for our work. After removing the images containing duplicate cell locations, we obtained 158 valid 3D images for experiments. These 3D confocal images have a resolution of $0.29 \times 0.29 \times 1\ \mu^3$m/voxel with a fixed size of 512×512 in XY-axis and Z-axis ranging from 53 to 159.

A comparison of the datasets is listed in Table 1 and sample images from each dataset are shown in Figure 4.

5 EXPERIMENTS

In this section, we first describe implementation details for training and then the configurations for the experiments, followed by quantitative results across multiple datasets. The implementation parameters mentioned in the section, except the batch size, are shared across all five datasets to further demonstrate the versatility of the proposed model. Based on the size of the image for each dataset, the batch size is set to 15 for MBM dataset, 16 for MBC dataset, and 75 for the remaining three datasets.

5.1 Implementation Details

5.1.1 Prepossessing

Our network can work with the raw image without sophisticated prepossessing and we simply normalize images, i.e., subtracting the mean and dividing by the standard deviation. Given the varied image size in DCC dataset, we resize the image and the density map together to 256×256 and linearly scale the density map to ensure the same cell count. We also pad the image edge from 150×150 to 152×152 with zeros in ADI dataset so that the image size is a multiple of 8, convenient to max-pooling in the network. The image size in the other three datasets remains the same.

5.1.2 Data Augmentation

During training, we randomly crop 87.5% region of the images in four 2D datasets and a patch of $128 \times 128 \times 32$ in the 3D dataset. Random horizontal flipping, vertical flipping, and 90-degree-rotation are also applied. Other non-90 degree rotation is not considered due to potential information loss during interpolation. Dropout is also added in the Self-Attention module of SAU-Net to reduce over-fitting, as shown in Figure 3.

5.1.3 Memory Limitation for 3D

The increase of the model dimension from 2D to 3D comes with a surge of GPU memory usage, and special attention is needed to accommodate this increase. Besides using a smaller patch size in training, a modification is adapted for the inference stage. Since it is infeasible to fit a whole 3D image in the GPU memory, we cut each image into tiles with the size of $128 \times 128 \times 32$, the same as training inputs. For example, given a input of $512 \times 512 \times 100$, we 1) zero-pad along the Z-axis to 128, a multiple of 32, 2) split it by a $4 \times 4 \times 4$ grid, 3) feed these 64 sub-images into the trained model, and 4) merge the predictions for those sub-images accordingly to the final prediction.

Despite our efforts to tackle the memory issue, we are unable to apply the proposed BN extension for 3D cases. As shown in the previous section, this extension relies on more accurate population statistics estimation using all the training images, while our current hardware settings encounter difficulty fitting one image. Significant downsampling, e.g., by a factor of 64, could alleviate the memory shortage, but it would lead to inferior results due to the loss of fine details at high resolution. Alternatively, we attempted to employ the proposed BN extension for those sub-images separately and average the statistics afterward; this workaround, however, did not achieve the similar improvement as in 2D cases given that the standard deviation operation is not linear. This is further complicated by the fact that the subsequent layers in deep networks take as inputs the normalized activations from prior layers, and the error could propagate throughout the networks. As a result, we need to train the model for substantially more steps with standard BN compared to 2D cases with our BN extension empirically shown in the next section. A potential solution would be distributed training with multi-GPU hardware [40], and we leave this for future work.

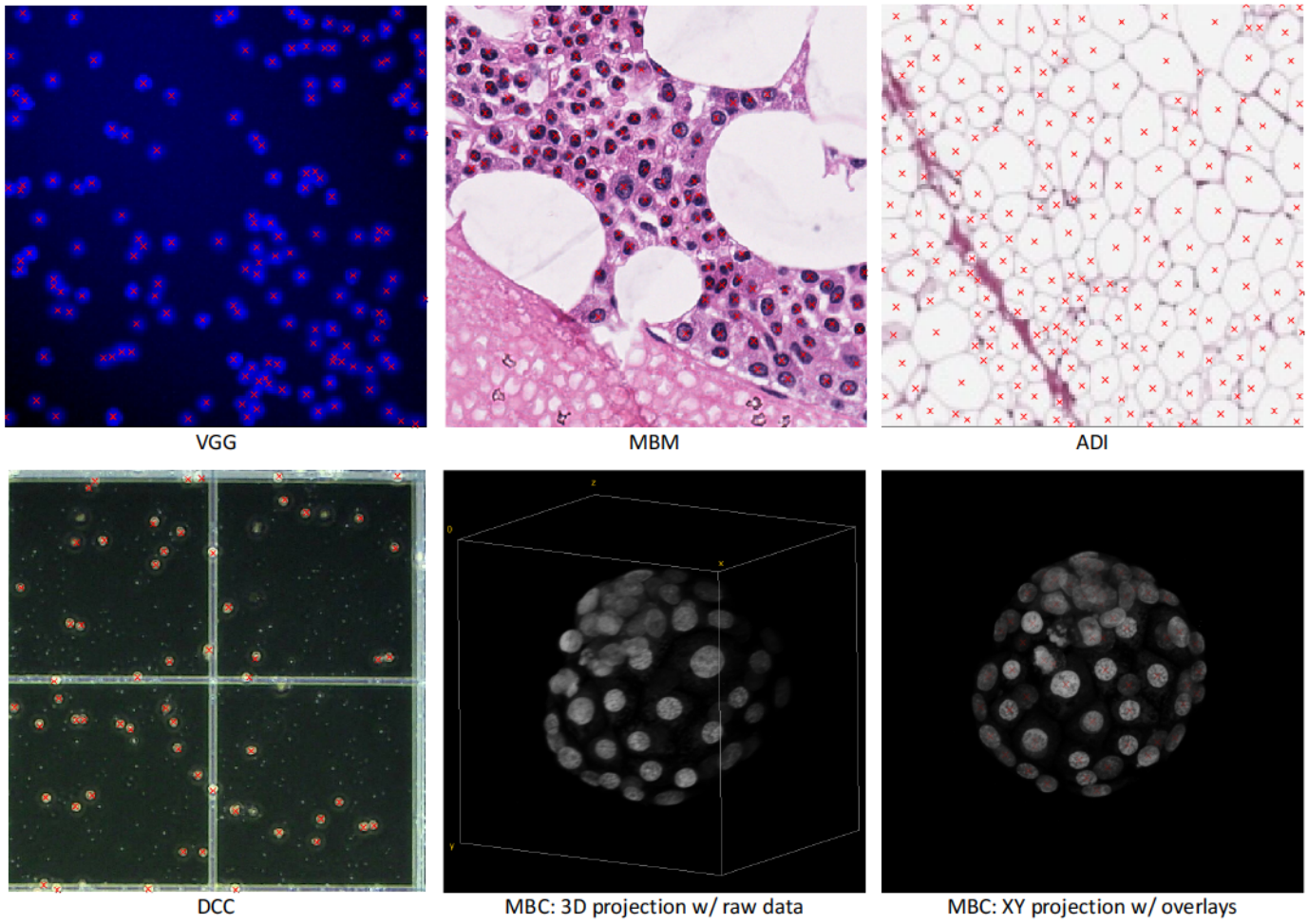

Fig. 4: Sample images from the five datasets used in the experiment. The dot annotations are represented as the red cross overlays. The projections are generated using Volume Viewer in ImageJ.

TABLE 1: A comparison of five datasets. N_{train} and N_{total} are the training image number and total image number, respectively. N_{train} is chosen to facilitate comparison with competing methods. Cell Count denotes the mean cell count per image and the corresponding standard deviation over all the images.

Dataset Name	Image Size	N_{train}/N_{total}	Cell Count	Type	Dimension	Bit Depth	Color
VGG [4]	256×256	64/200	174 ± 64	synthetic	2D	8	RGB
MBM [11]	600×600	15/44	126 ± 33	real	2D	8	RGB
ADI [9]	150×150	50/200	165 ± 44	real	2D	8	RGB
DCC [8]	varied	100/176	34 ± 22	real	2D	8	RGB
MBC [17]	varied	58/158	63 ± 24	real	3D	8	Grayscale

5.1.4 Optimization

We choose Adam optimizer with decoupled weight decay [41], which explicitly enforces weight decay instead of L_2 regularization, to train our network. The weight decay for the Adam optimizer is set to 0.001. The weights of the network are randomly initialized by Glorot method [42] for every experiment. For the learn rate, we use a cosine annealing schedule with warm restarts [43], which incrementally lowers the learning rate based on a cosine decay function. The initial value for the schedule is set to 0.001, and the restart step is set to 50 with a multiplier of 2. Each experiment is iterated for 350 steps for the 2D datasets and 12750 for the 3D dataset. On an NVIDIA Tesla V100, training on 2D and 3D cases takes approximately 5 minutes and 3 hours, respectively. In total, 2D SAU-Net has 2.2 million parameters, and 3D SAU-Net has 5.9 million parameters for optimization.

5.2 Experiment Configurations

We use Mean Absolute Error (MAE) of cell counts between prediction and ground-truth per image as the evaluation criteria. The training and testing split in each dataset are randomly selected for each trial, and we repeat the experiment 10 times, reporting the mean and variance of MAE. We first conduct an ablation study to examine the effectiveness of each proposed component, and then we compare our method with the state-of-the-art techniques in each dataset and list the best performance from their original work.

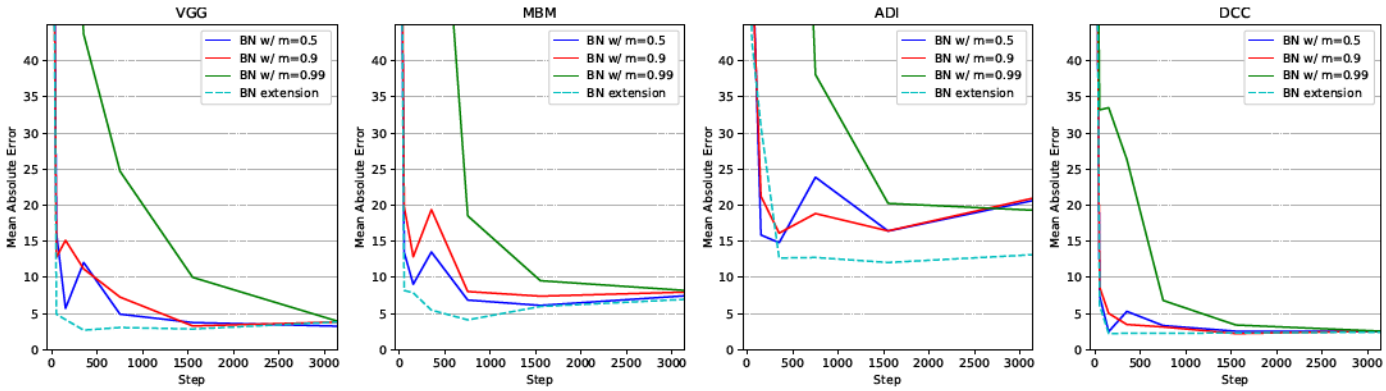

Fig. 5: The evaluation comparison between the proposed BN extension and standard BN with three common moving average momentums, m, on VGG, MBM, ADI, and DCC. All models were initialized with zero-mean and unit-variance at Step 0 (not shown) and are trained on SAU-Net under the same settings.

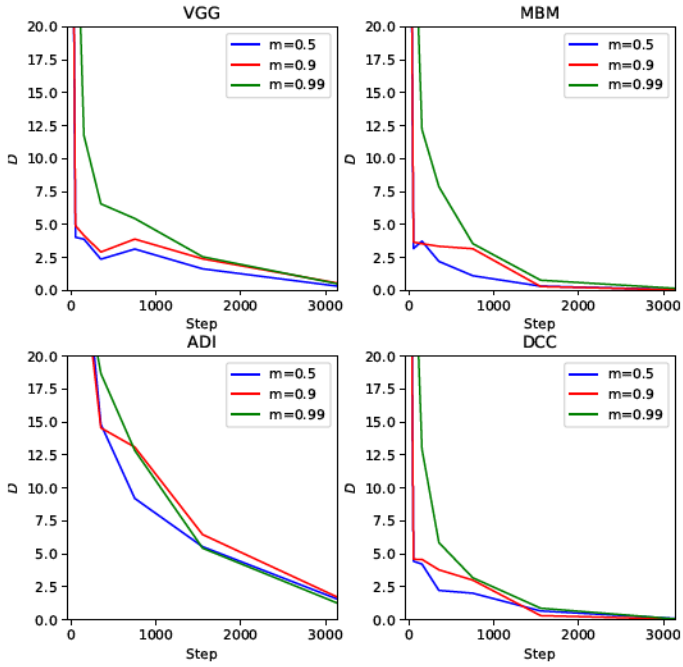

Fig. 6: The L1 distance of the statistics between standard BN and our extension under three common moving average momentums, m.

5.3 Results

5.3.1 Ablation Study

We conduct the ablation study to demonstrate the benefits of the proposed BN extension and SAU-Net and replace one component at a time with either standard BN or U-Net to evaluate its separate improvement. For our proposed BN extension, we test it against standard BN with varying moving average momentums on four 2D datasets. All experiments use SAU-Net with the same configurations. Due to the memory bottleneck, we skip the 3D dataset. Similarly, we report the results between SAU-Net and U-Net, both with the proposed BN extension on all five datasets, except the 3D dataset, which we only present with the difference between SAU-Net and U-Net with standard BN for the same memory issue.

5.3.1.1 *BN extension vs. BN*: We attribute our method's advantage against the standard BN to this reason: standard BN requires more training steps for the moving average to estimate the population statistics while our method directly calculates the population statistics. To show this, we conduct additional experiments and analyze them from a dynamic perspective. We train the model continuously for a large period of time and evaluate it periodically. Figure 5 plots the test performance as a function of training steps with BN under three common moving average momentum values, i.e., $m = 0.5, 0.9$ and 0.99, against our BN extension. These empirical values cover all the common values recommended in [36] and are also the default values in mainstream deep learning libraries, e.g., TensorFlow, PyTorch, and MXNet. The experiments show our BN extension helps achieve more stable results in a much earlier stage during training, further validating our argument: the training steps can be significantly reduced by applying our BN extension instead of moving averaging with commonly used empirical values, as in standard BN.

We also investigate the difference in the population statistics between standard BN and our extension. The difference, D, is measured in L1 distance over the statistics in all the BN layers:

$$D = \sum_{x \in \Omega} \| x_{\text{BN extension}} - x_{\text{BN}} \|_1, \tag{4}$$

where Ω denotes the set of BN statistics, namely, the mean and standard deviation, and $\| \cdot \|_1$ is the L1 distance. Figure 6 shows that statistics from both methods eventually converge during the course of training, and this also explains why the performance across different methods converges in Figure 5. Note that these statistics are based on the activations from each layer, and these activations are also updating at each step via optimization. Each dataset has its unique training dynamics, and some datasets may take longer for convergence, e.g., ADI dataset. Nonetheless, the general converging trend remains the same for all the datasets.

Overall, the results in Figure 5 and 6 indicate that the difference of standard BN, and our extension diminish gradually and the moving average momentums, m control the convergence speed. High m leads to fast convergence with large oscillation, while low m yields a stable yet

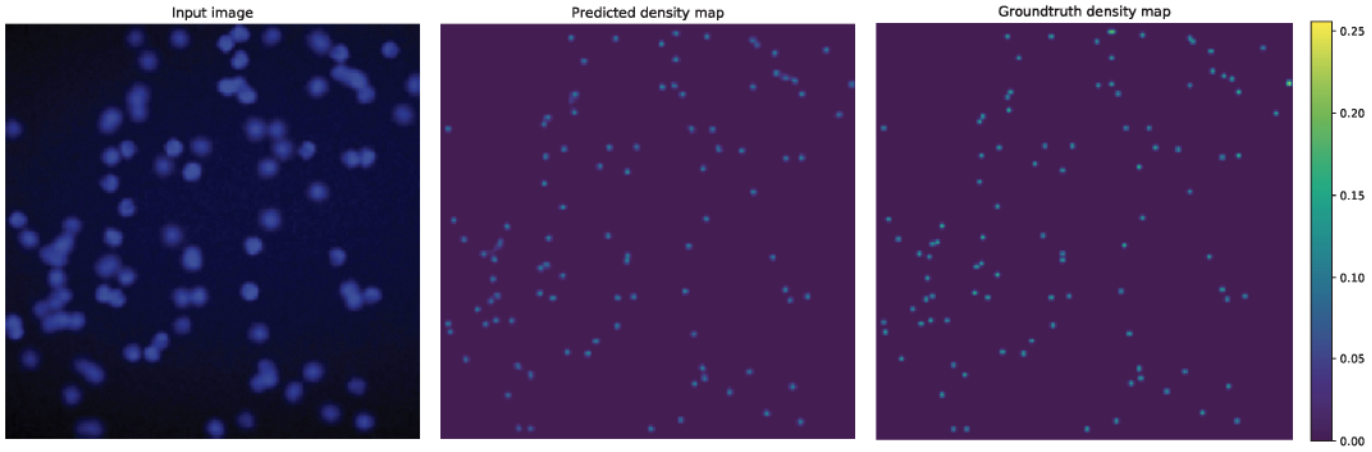

Fig. 7: Sample predicted density map on the test set for VGG dataset. Groundtruth Cell Count: 100, Predicted: 100.9

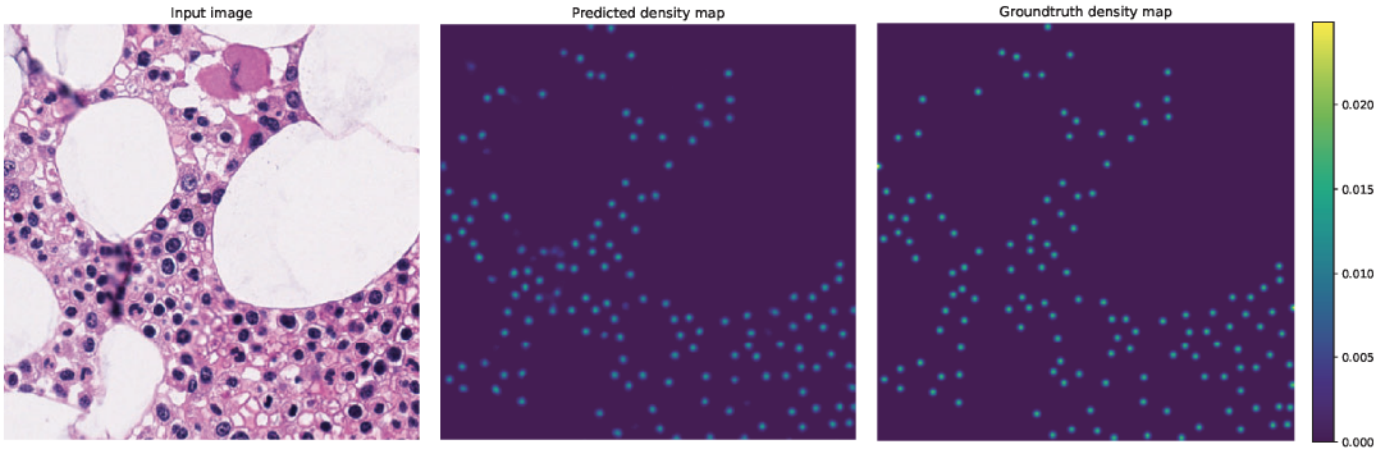

Fig. 8: Sample predicted density map on the test set for MBM dataset. Groundtruth Cell Count: 134, Predicted: 135.8

slow convergence. This process is further complicated by varying training dynamics, and it is challenging to develop a heuristic way to obtain the optimal other than the costly grid search. Therefore, this observation highlights the merits of the proposed extension: it facilitates training by reducing the steps needed for convergence and achieves more stable performance than the standard BN.

5.3.1.2 *SAU-Net vs. U-Net:* Table 2 and 3 demonstrates that the proposed network expansion with the Self-Attention module can further improve the performance of our model, and SAU-Net compares favorably to U-Net in both 2D and 3D. Our BN extension is applied in the experiment with all the 2D datasets, while in the 3D MBC dataset, we still reply on standard BN with two empirical moving average momentum values due to memory limitation. This leads a prolonged training, 12750 iterations, as compared to 350 for the 2D case. Regardless of BN choice, each experiment is conducted using the same BN method, with Self-Attention module being the only variable.

5.3.2 VGG

In this synthetic dataset, our performance is on par with the leading methods. The result and a sample prediction are shown in Table 4 and Figure 7. Note that the leading models are highly engineered for this synthetic dataset, and our method achieves state-of-the-art performance in the rest real datasets.

TABLE 2: MAE between U-Net and SAU-Net for 2D datasets w/ BN extension.

Method	VGG	MBM	ADI	DCC
U-Net	2.9 ± 0.7	7.8 ± 1.6	17.8 ± 2.3	3.2 ± 0.5
SAU-Net	$\mathbf{2.6 \pm 0.4}$	$\mathbf{5.7 \pm 1.2}$	$\mathbf{14.2 \pm 1.6}$	$\mathbf{3.0 \pm 0.3}$

TABLE 3: MAE between U-Net and SAU-Net for 3D dataset w/ standard BN. Due to memory limitation, BN extension is replaced by standard BN with various common momentums, denoted by m.

Method	MBC $m = 0.5$	MBC $m = 0.9$	MBC $m = 0.99$
U-Net	3.5 ± 0.4	3.3 ± 0.4	3.2 ± 0.4
SAU-Net	$\mathbf{2.6 \pm 0.3}$	$\mathbf{2.6 \pm 0.3}$	$\mathbf{2.7 \pm 0.3}$

5.3.3 MBM

Our method outperforms other leading methods in this real dataset. Table 5 and Figure 8 show the result and a sample prediction.

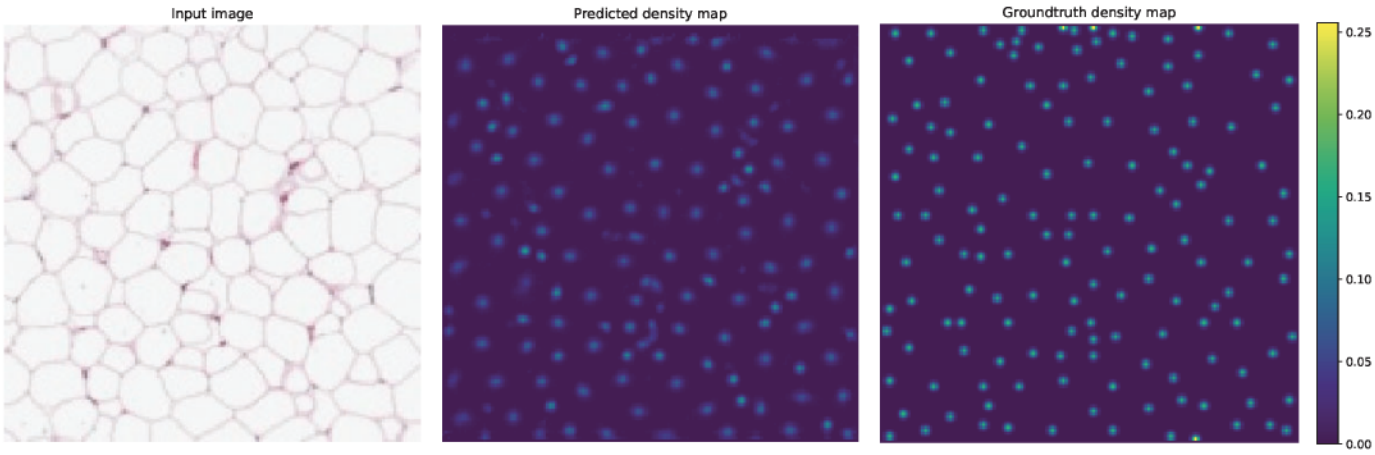

Fig. 9: Sample predicted density map on the test set for ADI dataset. Groundtruth Cell Count: 149, Predicted: 142.1

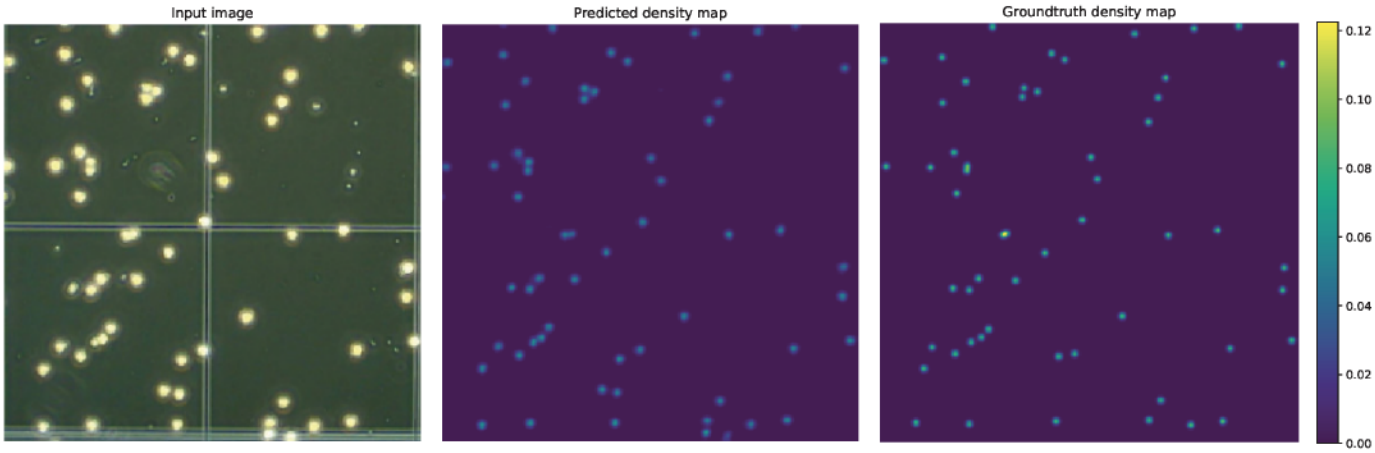

Fig. 10: Sample predicted density map on the test set for DCC dataset. Groundtruth Cell Count: 55, Predicted: 58.9

TABLE 4: VGG (200 images in total)

Method	MAE	N_{train}
ResNet-152 (R), Xue et al. [7]	7.5 ± 2.2	100
GMN, Lu et al. [12]	3.6 ± 0.3	32
FCRN-A, Xie et al. [6]	2.9 ± 0.2	64
CCF, Jiang et al. [44]	2.6 ± 0.1	50
Count-Ception, Cohen et al. [9]	$\mathbf{2.3 \pm 0.4}$	50
SAU-Net (proposed)	2.6 ± 0.4	64

TABLE 5: MBM (44 images in total)

Method	MAE	N_{train}
FCRN-A, Xie et al. [6]	21.3 ± 9.4*	15
Marsden et al. [8]	20.5 ± 3.5	15
Count-Ception, Cohen et al. [9]	8.8 ± 2.3	15
CCF, Jiang et al. [44]	8.6 ± 0.3	15
SAU-Net (proposed)	$\mathbf{5.7 \pm 1.2}$	15

* *Implemented by Cohen et al. [9]*

5.3.4 ADI

Similarly, the proposed method achieves state-of-the-art performance in this dataset. Table 6 shows the result include Adiposoft [45], an unsupervised method specialized for adipose tissues like in ADI, and other general supervised methods. As illustrated in Figure 9, this dataset represents a challenging scenario given the complicated cell structure.

TABLE 6: ADI (200 images in total)

Method	MAE	N_{train}
Count-Ception, Cohen et al. [9]	19.4 ± 2.2	50
Adiposoft, Galarraga et al. [45]	14.8 ± 13.6*	–
CCF, Jiang et al. [44]	14.5 ± 0.4	50
SAU-Net (proposed)	$\mathbf{14.2 \pm 1.6}$	50

* *Implemented by Jiang et al. [44]*

5.3.5 DCC

Table 7 also shows the superior result of our method in this recently released real dataset. A sample prediction is provided in Figure 10.

TABLE 7: DCC (176 images in total)

Method	MAE	N_{train}
Marsden et al. [8]	8.4^a	100
Shi et al. [46]	3.2^b	100
U-Net, Ronneberger et al. [13]	3.2 ± 0.5^c	100
SAU-Net (proposed)	$\mathbf{3.0 \pm 0.3}$	100

[a,b] *Reported in a fixed split in their work.*
[c] *Implemented in this work.*

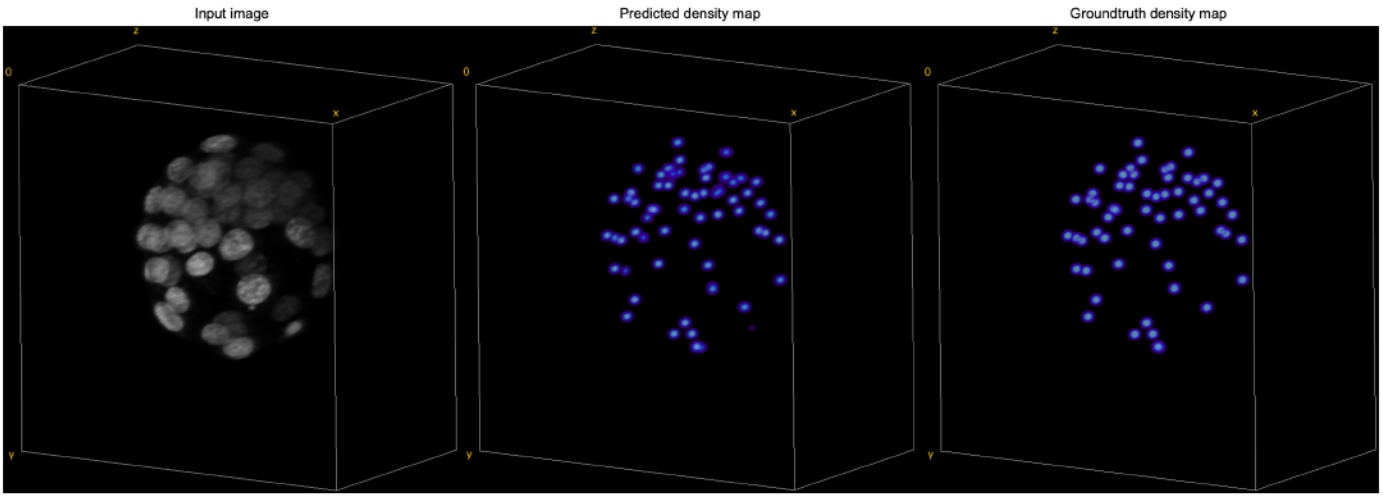

Fig. 11: Sample predicted density map on the test set for MBC dataset. Groundtruth Cell Count: 59, Predicted: 60.4

5.3.6 MBC

In this 3D dataset, our method compares favorably to standard 3D U-Net [47] and a popular conventional method, MINS [34], as observed in Table 8. MINS, as an unsupervised method, does not require training data but still performs worse than our method, given the fact it was initially designed for this specific mouse blastocyst data like MBC. A sample prediction is provided in Figure 11.

TABLE 8: MBC (158 images in total)

Method	MAE	N_{train}
MINS, Lou et al. [34]	5.8 ± 0.2^a	-
3D U-Net, Çiçek et al. [47]	3.3 ± 0.4^b	58
SAU-Net (proposed)	$\mathbf{2.6 \pm 0.3}$	58

[a] *Implemented by Saiz et al. [17]*
[b] *Implemented in this work.*

6 Discussion

6.1 Cell Detection

In addition to cell counts, cell detection can be obtained from the predicted density map via post-process techniques. Following [37], we apply Connected Component Analysis[5] after thresholding and then calculate cell centroid as the final positions. A detection is considered successful if the predicted centroid lies within a tolerance R from the groundtruth location, where R is set as the empirical cell radius. Table 9 lists the precision, recall, and F_1 score for cell detection.

TABLE 9: The detection results across five datasets.

Dataset Name	Precision	Recall	F_1 score
VGG [4]	99.94 ± 0.02	89.65 ± 0.28	94.51 ± 0.16
MBM [11]	88.76 ± 1.19	92.83 ± 1.35	90.73 ± 0.40
ADI [9]	88.57 ± 0.97	93.13 ± 0.78	90.78 ± 0.48
DCC [8]	92.52 ± 0.79	93.10 ± 0.72	92.81 ± 0.48
MBC [17]	92.52 ± 0.44	98.11 ± 0.24	95.23 ± 0.25

5. https://scikit-image.org/docs/dev/api/skimage.measure.html#label

An analysis of the detection results indicates that false positive detection usually happens when the background is cluttered with other objects. On the other hand, the border area with only partial cells visible or the area with clustered cells tends to have false negative detection. Typical results can be found in Appendix B.

6.2 Impact of Batch Size

Since the batch size is another hyperparameter for our proposed BN extension, we investigate the impact by varying the batch size, B. Specifically, we shrink the batch size by using a fraction of B, e.g., $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$, and B is the same as in the previous Section, namely 15 for MBM and 75 for the rest three 2D datasets (3D MBC excluded due to memory limitations). All the experiments trained on SAU-Net with the identical settings except the training steps extended accordingly, which is needed for convergence. The results in Figure 12 show that our BN extension is not sensitive to batch size, and larger batch sizes can reduce the training time. We additionally test the extreme case with $B = 1$, and the gradient descent training becomes unstable since the gradients are calculated only based on 1 sample and become very noisy. In general, we recommend using a large batch size to avoid unstable training and reduce the training time.

7 Conclusion

In this paper, we propose a novel deep network structure that employs a self-attention module with U-Net and is capable of universally solving the cell counting task for different cell types. Furthermore, the proposed SAU-Net applies to both 2D and 3D images, allowing a unified framework for multi-dimension counting. We also extend the current Batch Normalization method to better adapt to small datasets. Based on prior work, we introduce a new benchmark for 3D image-based cell counting. Together, we assess the proposed method on five benchmarks, and our model has consistently shown superior or competitive performance to the state-of-the-art methods.

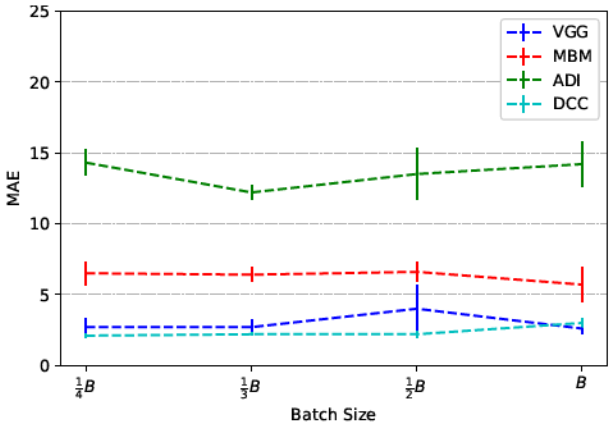

Fig. 12: The impact of batch size B on our proposed BN extension with various datasets. B is set to 15 for MBM and 75 for the rest three 2D datasets as in the previous section.

Despite our model only requiring simplified dot-annotated labels, there is still an urgent need to reduce the reliance on laborious manual annotations. Future work in this area includes adversarial adaption, which transfers knowledge from source data type to target data type via adversarial training. On the other hand, semi-supervised learning utilizes unlabeled data jointly with labeled data to facilitate training. Both directions aim to take advantage of unlabeled data or data from other sources, effectively minimizing the need for labels and could lead to further improvement.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Nestor Saiz for his helpful comments on the 3D dataset MBC. This work was supported in part by the National Science Foundation under grants OCI-1153775 and OAC-1649916.

REFERENCES

[1] R. Bernier, C. Golzio, B. Xiong, H. A. Stessman, B. P. Coe, O. Penn, K. Witherspoon, J. Gerdts, C. Baker, A. T. Vulto-van Silfhout *et al.*, "Disruptive chd8 mutations define a subtype of autism early in development," *Cell*, vol. 158, no. 2, pp. 263–276, 2014.

[2] M.-Y. C. Polley, S. C. Leung, L. M. McShane, D. Gao, J. C. Hugh, M. G. Mastropasqua, G. Viale, L. A. Zabaglo, F. Penault-Llorca, J. M. Bartlett *et al.*, "An international ki67 reproducibility study," *Journal of the National Cancer Institute*, vol. 105, no. 24, pp. 1897–1906, 2013.

[3] A. Mukherjee, N. A. Repina, D. V. Schaffer, and R. S. Kane, "Optogenetic tools for cell biological applications," *Journal of thoracic disease*, vol. 9, no. 12, p. 4867, 2017.

[4] V. Lempitsky and A. Zisserman, "Learning to count objects in images," in *Advances in neural information processing systems*, 2010, pp. 1324–1332.

[5] 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)*, Nov 2012, pp. 2685–2688.

[6] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting and detection with fully convolutional regression networks," *Computer methods in biomechanics and biomedical engineering: Imaging & Visualization*, vol. 6, no. 3, pp. 283–292, 2018.

[7] Y. Xue, N. Ray, J. Hugh, and G. Bigras, "Cell counting by regression using convolutional neural network," in *European Conference on Computer Vision*. Springer, 2016, pp. 274–290.

[8] M. Marsden, K. McGuinness, S. Little, C. E. Keogh, and N. E. O'Connor, "People, penguins and petri dishes: Adapting object counting models to new visual domains and object types without forgetting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8070–8079.

[9] J. P. Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio, "Count-ception: Counting by fully convolutional redundant counting," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 18–26.

[10] T. Dumur, S. Duncan, K. Graumann, S. Desset, R. S. Randall, O. M. Scheid, H. W. Bass, D. Prodanov, C. Tatout, and C. Baroux, "Probing the 3d architecture of the plant nucleus with microscopy approaches: challenges and solutions," *Nucleus*, vol. 10, no. 1, pp. 181–212, 2019.

[11] P. Kainz, M. Urschler, S. Schulter, P. Wohlhart, and V. Lepetit, "You should use regression to detect cells," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 276–283.

[12] E. Lu, W. Xie, and A. Zisserman, "Class-agnostic counting," *arXiv preprint arXiv:1811.00472*, 2018.

[13] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[14] O. Oktay, J. Schlemper, L. L. Folgoc, M. C. H. Lee, M. P. Heinrich, K. Misawa, K. Mori, S. G. McDonagh, N. Y. Hammerla, B. Kainz, B. Glocker, and D. Rueckert, "Attention u-net: Learning where to look for the pancreas," *CoRR*, vol. abs/1804.03999, 2018.

[15] Y. Guo, Q. Wang, O. Krupa, J. Stein, G. Wu, K. Bradford, and A. Krishnamurthy, "Cross modality microscopy segmentation via adversarial adaptation," in *Bioinformatics and Biomedical Engineering*, I. Rojas, O. Valenzuela, F. Rojas, and F. Ortuño, Eds. Cham: Springer International Publishing, 2019, pp. 469–478.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[17] N. Saiz, K. M. Williams, V. E. Seshan, and A.-K. Hadjantonakis, "Asynchronous fate decisions by single cells collectively ensure consistent lineage composition in the mouse blastocyst," *Nature communications*, vol. 7, no. 1, pp. 1–14, 2016.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[21] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[22] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[23] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[25] Y. Guo, J. Stein, G. Wu, and A. Krishnamurthy, "Sau-net: A universal deep network for cell counting," in *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, 2019, pp. 299–306.

[26] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect cells using non-overlapping extremal regions," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2012*, N. Ayache, H. Delingette, P. Golland, and K. Mori, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 348–356.

[27] ——, "Detecting overlapping instances in microscopy images using extremal region trees," *Medical Image Analysis*, vol. 27, pp. 3 – 16, 2016, discrete Graphical Models in Biomedical Image Analysis.

[28] ——, "Interactive object counting," in *European conference on computer vision*. Springer, 2014, pp. 504–518.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.
[30] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.
[31] Yue Guo, J. Wrammert, K. Singh, A. KC, K. Bradford, and A. Krishnamurthy, "Automatic analysis of neonatal video data to evaluate resuscitation performance," in *2016 IEEE 6th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)*, Oct 2016, pp. 1–6.
[32] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3431–3440.
[33] F. Xing and L. Yang, "Robust nucleus/cell detection and segmentation in digital pathology and microscopy images: a comprehensive review," *IEEE reviews in biomedical engineering*, vol. 9, pp. 234–263, 2016.
[34] X. Lou, M. Kang, P. Xenopoulos, S. Munoz-Descalzo, and A.-K. Hadjantonakis, "A rapid and efficient 2d/3d nuclear segmentation method for analysis of early mouse embryo and stem cell image data," *Stem cell reports*, vol. 2, no. 3, pp. 382–397, 2014.
[35] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803.
[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[37] O. Krupa, G. Fragola, E. Hadden-Ford, J. T. Mory, T. Liu, Z. Humphrey, B. W. Rees, A. Krishnamurthy, W. D. Snider, M. J. Zylka, G. Wu, L. Xing, and J. L. Stein, "Numorph: tools for cellular phenotyping in tissue cleared whole brain images," *bioRxiv*, 2021.
[38] A. Lehmussola, P. Ruusuvuori, J. Selinummi, H. Huttunen, and O. Yli-Harja, "Computational framework for simulating fluorescence microscope images with cell populations," *IEEE transactions on medical imaging*, vol. 26, no. 7, pp. 1010–1016, 2007.
[39] J. Lonsdale, J. Thomas, M. Salvatore, R. Phillips, E. Lo, S. Shad, R. Hasz, G. Walters, F. Garcia, N. Young *et al.*, "The genotype-tissue expression (gtex) project," *Nature genetics*, vol. 45, no. 6, p. 580, 2013.
[40] M. Yamazaki, A. Kasagi, A. Tabuchi, T. Honda, M. Miwa, N. Fukumoto, T. Tabaru, A. Ike, and K. Nakashima, "Yet another accelerated sgd: Resnet-50 training on imagenet in 74.7 seconds," *arXiv preprint arXiv:1903.12650*, 2019.
[41] I. Loshchilov and F. Hutter, "Fixing weight decay regularization in adam," *CoRR*, vol. abs/1711.05101, 2017.
[42] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 2010, pp. 249–256.
[43] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.
[44] N. Jiang and F. Yu, "A cell counting framework based on random forest and density map," *Applied Sciences*, vol. 10, no. 23, p. 8346, 2020.
[45] M. Galarraga, J. Campión, A. Muñoz-Barrutia, N. Boqué, H. Moreno, J. A. Martínez, F. Milagro, and C. Ortiz-de Solórzano, "Adiposoft: automated software for the analysis of white adipose tissue cellularity in histological sections," *Journal of lipid research*, vol. 53, no. 12, pp. 2791–2796, 2012.
[46] Z. Shi, P. Mettes, and C. G. Snoek, "Counting with focus for free," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4200–4209.
[47] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2016, pp. 424–432.

Yue Guo is working toward his Ph.D. degree in computer science from the University of North Carolina at Chapel Hill. His research interests include deep learning algorithms and their applications broadly in bioinformatics and computer vision, with an emphasis on neonatal video analysis and image-based cell counting.

Oleh Krupa received a BS degree in biological engineering and a MEng degree in biomedical engineering from Cornell University, Ithaca, New York. He is currently working toward a PhD degree in the Joint Department of Biomedical Engineering at the University of North Carolina - Chapel Hill and North Carolina State University. His research focuses on identifying the genetic influences on human brain structure at cellular resolution.

Jason Stein is an assistant professor in the Department of Genetics and the UNC Neuroscience Center at the University of North Carolina at Chapel Hill. His research interests are on how genetic variation influences brain development and structure, leading to risk for neuropsychiatric disorders. His lab uses tissue clearing in brain samples to study microscale brain structure.

Guorong Wu received the Ph.D. degree in computer science and engineering from Shanghai Jiao Tong University, Shanghai, China. He is currently an Assistant Professor at the Department of Psychiatry and Computer Science, The University of North Carolina, Chapel Hill, as an Assistant Professor. He has published more than 200 papers in international journals and conference proceedings. His research interests focus on fast and robust analysis of large population data, computer assisted diagnosis, and image guided radiation therapy. In 2016, he received the Independent Career Award from NIH for the contribution on non-invasive neurobiological basis for early diagnosis of Alzheimer's disease. He won the best paper award in MICCAI 2017. Dr. Wu is the associate editor for Neuroinformatics since 2019.

Ashok Krishnamurthy is Deputy Director at the Renaissance Computing Institute (RENCI), a Research Professor in the Department of Computer Science at the University of North Carolina at Chapel Hill. He is also Faculty Director for Informatics and Data Science at NC TraCS. His research interests are in Data Science, Biomedical Informatics, Machine Learning, and Research Cyberinfrastructure. Ashok received a PhD in electrical and computer engineering from the University of Florida. He is a member of IEEE and ACM.

APPENDIX A
3D LIGHT-SHEET MOUSE CORTEX DATASET

In this appendix, we provide additional results on a 3D light-sheet dataset for mouse cortex from recent work [37]. This dataset has a resolution of $0.75 \times 0.75 \times 2.5$ μ^3m/voxel and contains 16 patches of $224 \times 224 \times 64$ voxels. Each patch includes 908 ± 270 cells, and the cell count is significantly higher than the previous datasets. Note that this dataset is intended for segmentation-based methods and contains whole-cell annotation. For consistency, we convert it into dot-annotations by using the centroids of whole-cell annotations. We compare our SAU-Net against the standard 3D U-Net, and both models are trained with standard BN. The results from the original segmentation-based work are omitted since it requires whole-cell annotations, preventing a direct comparison.

We follow the $1:2$ training/test split ratio as in most previous datasets and randomly select 6 for training with the rest for evaluation. Similarly, we repeat this process 10 times and report the mean and standard deviation of MAE, precision, recall, and F_1 score in Table 10. Although the MAE is higher than the previous datasets, this is caused by the high cell density, and the performance under other metrics is on par with the other datasets. The results on this high-density dataset further corroborate the versatility of SAU-Net. Figure 13 shows a sample input data with the corresponding prediction and groundtruth density map.

TABLE 10: 3D Light-sheet Dataset (16 images in total)

Method	MAE	Precision	Recall	F_1 score	N_{train}
3D U-Net*, Çiçek et al. [47]	85.3 ± 28.2	97.89 ± 1.00	93.70 ± 0.92	95.74 ± 0.60	6
SAU-Net (proposed)	72.7 ± 19.7	98.26 ± 0.19	95.34 ± 0.36	96.78 ± 0.17	6

* *Implemented in this work.*

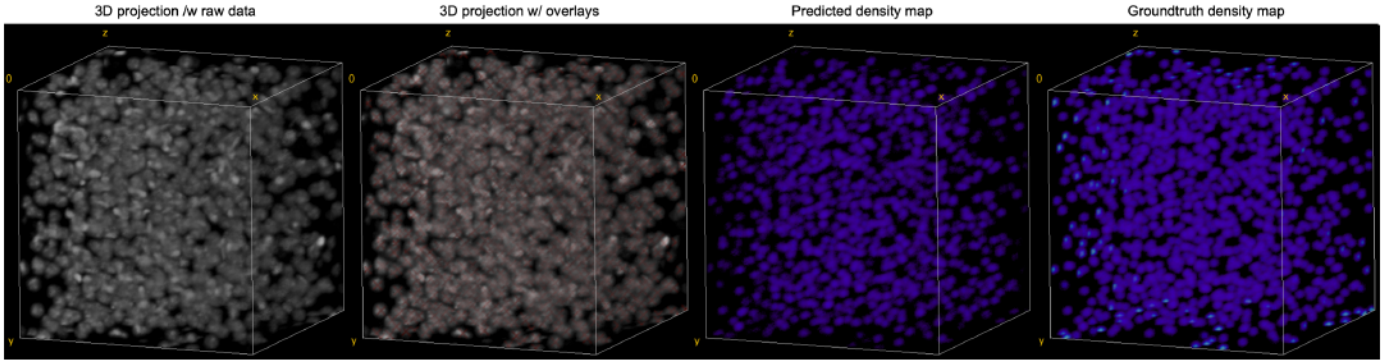

Fig. 13: Sample predicted density map on the test set for the 3D light-sheet dataset. The dot annotations are represented as the red cross overlays. Groundtruth Cell Count: 1257, Predicted: 1194.6.

APPENDIX B
FAILURE CASES

Figure 14 shows some typical failure cases across all six datasets. These cases show that our method is able to deal with a large variety of scenarios, including both 2D and 3D data. However, scenes with extremely dense objects or noisy backgrounds remain a challenge, opening up opportunities for future work.

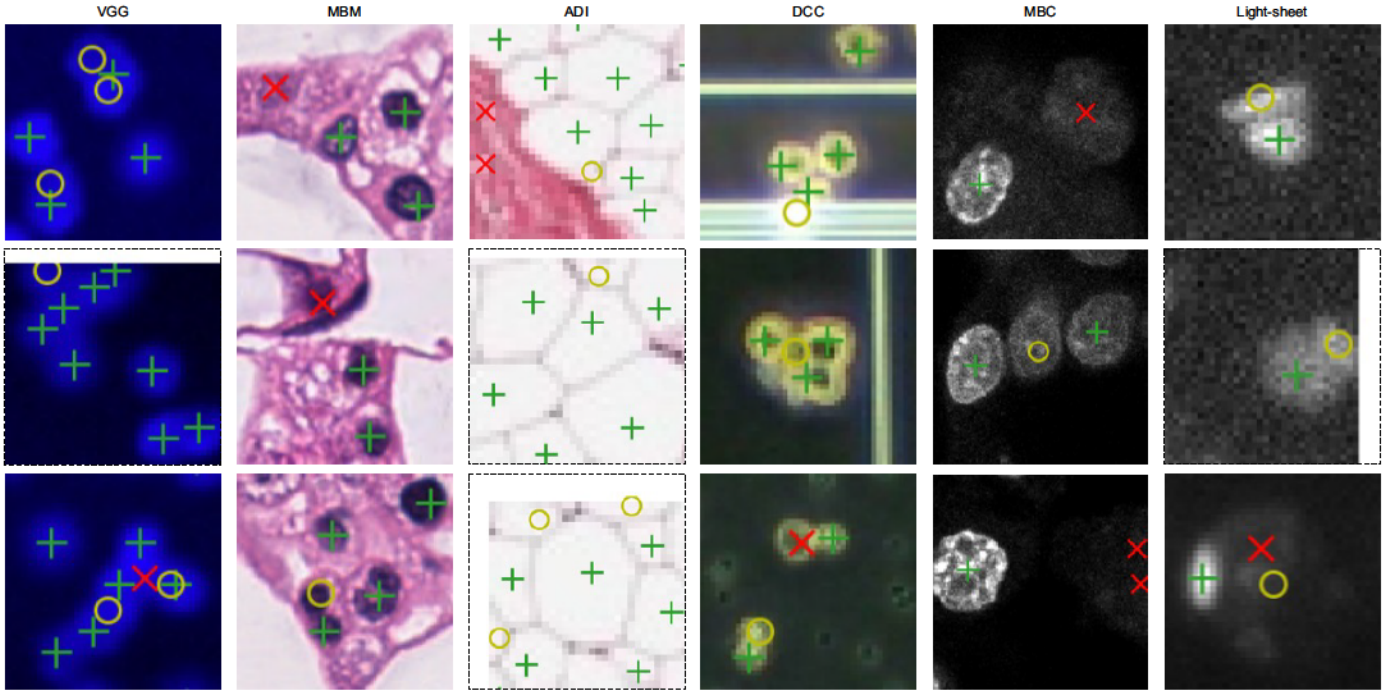

Fig. 14: Typical failure cases across all six datasets. Images are zoomed in for details. Dashed boxes denote the border cases, and the blank space within the boxes is beyond the border.