Exploring Cell counting with Neural Arithmetic Logic Units

Ashish Rana*, Taranveer Singh†, Harpreet Singh‡, Neeraj Kumar‡ and Prashant Singh Rana§
Department of Computer Science, TIET Patiala, Punjab, India.
Email: *{arana_be15, tsingh_me17, harpreet.s, neeraj.kumar, prashant.singh}@thapar.edu

***Abstract*—The big problem for neural network models which are trained to count instances is that whenever test range goes higher than training range generalization error increases i.e. they are not good generalizers outside training range. Consider the case of automating cell counting process where more dense images with higher cell counts are commonly encountered as compared to images used in training data. By making better predictions for higher ranges of cell count we are aiming to create better generalization systems for cell counting. With architecture proposal of neural arithmetic logic units (NALU) for arithmetic operations, task of counting has become feasible for higher numeric ranges which were not included in training data with better accuracy. As a part of our study we used these units and different other activation functions for learning cell counting task with two different architectures namely Fully Convolutional Regression Network and U-Net. These numerically biased units are added in the form of residual concatenated layers to original architectures and a comparative experimental study is done with these newly proposed changes . This comparative study is described in terms of optimizing regression loss problem from these models trained with extensive data augmentation techniques. We were able to achieve better results in our experiments of cell counting tasks with introduction of these numerically biased units to already existing architectures in the form of residual layer concatenation connections. Our experiments on a batch size of 16 shows a 20.22% relative improvement as compared to original architecture and 34.84% relative improvement on our custom created high count dataset created from BBBC005 synthetic cell count dataset achieving higher generalization capabilities for U-net based architectures. These results confirm that above mentioned numerically biased units does help models to learn numeric quantities for better generalization results.**

***Index Terms*—Neural Arithmetic Logic Units, Cell Counting, ResNets, Fully Convolutional Regression Networks, U-net.**

I. Introduction

Ability to generalize concepts is fundamental component of intelligence and core for designing smart systems [6], [7]. Neural networks simulates this behavior with hierarchical learning of concepts. When it comes to automation, counting is an important task from machine vision application [1] to cell counting [2]. While neural networks manipulates numerical quantities but it is not associated with systematic generalization [3], [4]. These networks fail to generalize as evident from high generalization error while predicting quantities that lie outside the training numerical range as shown in table I or in figure 5 elaborated in experiment section. This highlights memorization behavior in neural networks instead of generalization abilities.

Neural accumulators (NAC) and neural arithmetic logic units (NALU) [5] are biased to learn systematic numerical computation and performs relatively better than non linear activation functions for arithmetic operations. This numerical bias of learning computations makes them excellent choice for counting tasks which are essentially is an increment addition operation only. Deep learning models generally take either segmentation approach with explicit counting trainer or end-to-end counting via a regression loss. In this paper we will go through the latter approach [2] in detail for automation of cell counting process. As cell counting is cumbersome task and dense cell images with higher cell counts containing data outside training numeric range are common in real world scenarios. Achieving true cell automation with less generalization errors is the prime objective of this paper.

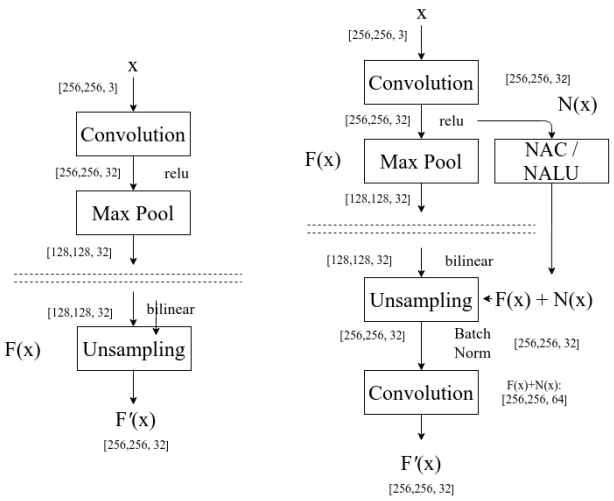

Fig. 1. Abstract representation of proposed modified architecture (*Right*) with either neural accumulator (NAC) or neural arithmetic logic unit (NALU) as compared to previous Fully Convolution Regression Networks (*Left*). Here, as clearly specified with dimensional analysis of tensor blocks (*Right*) the layers in the newly proposed model are concatenated and again squeezed into same size as earlier model architecture.

In regression loss approach Fully convolutional regression networks [2] and U-net [8] architectures learns mapping between an image I(x) and a density map D(x), given by F: $I(x) \rightarrow D(x)$ ($I \in R^{m \times n}$, $D \in R^{m \times n}$) for a $m \times n$ pixel image. Later on, variations of these architectures with different activation functions are discussed in this paper. And their

performance is compared with Neural accumulators (NAC) and Neural arithmetic logic units (NALU) concatenated architectures in the form of numerically biased residual connections , similar to ResNets [9] as shown in figure 1 but instead of adding the previous layer input they are concatenated. On the surface, this proposed architecture changes leads to accuracy improvements due to increased model capacity with these numerically biased units. But, our results with custom high count dataset created from BBBC005 [29] reflects increased generalization counting abilities for high cell count images with higher relative decrease in mean absolute error(MAE).

Our concatenation based residual architecture utilizes the fundamentals of batch normalization like specified identity mapping architecture in ResNets [17]. But, instead of using convolution operation directly this network leverages numerical bias information obtained from NAC and NALU operations applied on input layer and then finally uses convolution operation on the concatenated layer. Before and after this concatenation of this numerical bias learning operation, batch normalization is carried out and output of this operation is added back again to our next main network layer, as shown in figure 1.

With means of this paper we introduce changes in current regression based model architectures for end-to-end counter training and produce systems with higher accuracies for higher count testing ranges as well. Also, we validate our trained models on a different specially tailored validation dataset with approximately five times higher counts of cells as compared to training dataset created from BBBC005 synthetic cell dataset [29].

II. Related Work

Intuitive numerical understanding is important in learning and by adjunct important in deep learning [6] for creating better models with higher generalization capabilities. Counting objects [2], [10]–[12], [14] in given image is a widely studied task. Trained models for counting tasks either use a deep learning model to segment instances of given object then count them in a post-processing step [15] or learn end-to-end predict count via a regression loss [2]. Networks like Count-ception [16] added the concept of average over redundant predictions with its deep Inception family network. Also, recent architectures like ResNets [17], Highway Networks [18] and Densenets [19] advocate linear connections like Count-ception to promote better learning bias. Such models have better performance, though additional computational overhead due to increased depth of given architectures do arise. Our work highlights the generalization capabilities of the network, that extrapolate well on unseen parts of solution space which highlights underlying structure of behavior governing-equations [20]. We introduce architectural changes in models that learns residual functions and preserves input information which is numerically biased with reference to input layers. It is somewhat similar to ResNets [9], which are easier to optimize and gain accuracy with increasing depth. With our experiments we aim to highlight that models with numerically biased concatenated residual functions helps in achieving better results with their addition in the form of a comparative study with original architectures. Also, with our results in this paper we demonstrate with our results that backpropagation learns this numerical bias without any explicit numeric quantity being provided as input implying that better computer vision counters can be trained with this module when added to existing convolutional neural network architectures.

Density based estimation doesnt require prior object detection or segmentation [13], [21], [22]. In previous years, several works have investigated this approach. In [13], the problem is stated as density estimation with a supervised learning algorithm, $D(x) = c^T \phi(x)$, where D(x) represents ground-truth density map, and $\phi(x)$ represents local features and parameters c are learned by minimizing the error between predicted and true density with quadratic programming over all possible sub-windows. In [22], regression forest is used to exploit patch-based idea for learning structured labels, then for new input image density map is estimated averaged over structured patch-based predictions. Also, in [21] an algorithm is used that allows fast interactive counting with ridge regression.

Cell counting [13] problem is classified into supervised learning problem that learns mapping between an image I(x) and a density map D(x), denoted by $F: I(x) \rightarrow D(x)$ ($I \in R^{m \times n}$, $D \in R^{m \times n}$) for a $m \times n$ pixel image, see figure 2. Density function D(x) function is defined on pixels in given image, integrating this map over an image region gives an estimate of number of cells in that region. CNNs [23], [24] are quite popular in the bio-medical imaging because of their simple architecture and achieve great results. Like in mitosis detection [26], neuronal membrane segmentation [25] and analysis of C. elegans embryos development [27]. Previously, fully convolutional regression networks (FCRNs) and Count-ception have given state-of-the-art results in cell counting, with potential for cell detection of overlapping cells.

Also, U-Nets [8] a type of fully convolutional network, uses a modified version of architecture proposed by Ciresan et al. [25] as latter is slow and trade-off between localization and use of context are present. In U-Nets pooling operations are replaced by upsampling operations to supplement usual contracting network. For localization high resolution features from contracting path are combined with unsampled output. Based on this information a successive convolution layer then learn to assemble more precise output. For our experimentation we selected fully convolution regression networks(FCRN) and U-net based on simple architecture and relative similarity in architecture with the difference being that U-net already uses inputs from previous layer for localization.

III. Experiments

In this section first we conceptually explore numerical accumulators (NACs) and numerical arithmetic logic units (NALUs) and compare their addition capabilities with multi-layer perceptrons equipped with different activation functions. With this study we aim to select concatenated residual connection variants from above mentioned numerically bias

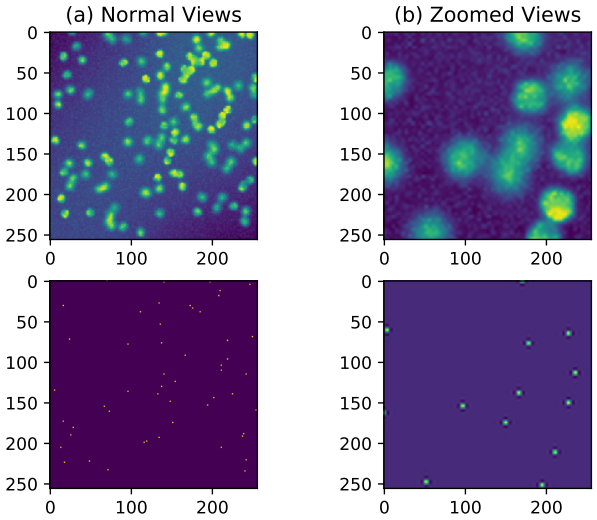

Fig. 2. **(a) Normal Views:** column represents training image in first row and corresponding annotated image in second row from synthetic dataset [28]. **(b) Zoomed Views:** This column represents zoomed in version of earlier mentioned images for better visualization.

units which can best approximate the counting behavior and compare them with standard FCRN and U-net neural network architecture for regression loss approach in the following experiment done on synthetic dataset [28]. For validation of counting generalization achieved, our trained models are tested against synthetic cell image dataset with approximately five times higher counts than training data.

A. Visual understanding of neural accumulators (NACs) and neural arithmetic logic units (NALUs)

Neural accumulators (NACs) [5] supports accumulation of numerical quantities additively, a desirable bias for linear exploration while counting. It is special type of linear layer with transformation matrix **W** being continuous and differentiable parameterization for gradient descent. W = $\tanh(\hat{W})\odot\sigma(\hat{M})$ consists of elements in [-1, 1] with bias close to 1, 0, and 1. See figure 3 for ideation of this concept with following equations for NAC: a = Wx, W = $\tanh(\hat{W})\odot\sigma(\hat{M})$ where $\hat{W}$, $\hat{M}$ are learning parameters and W is transformation matrix.

For complex mathematical operations like multiplication and division we use neural arithmetic logic units (NALUs) [5]. It uses weighted sum of two sub-cells, one for addition or subtraction and another of multiply, division or power functions. It demonstrates that neural accumulators (NACs) can be extended for learning scaling operations with gate-controlled sub-operations. See figure 3 for ideation of this concept with following equations for NALU: y = g$\odot$a + (1-g)$\odot$m; m = expW(log(|x| + ϵ)), g = σ(Gx) where m is subcell that operates in *log* space and g is learned gate, both contains learning parameters.

B. Comparative analysis of addition operation

Here, we use neural networks with NACs/NALUs and multilayer perceptrons (MLP) with different activation functions

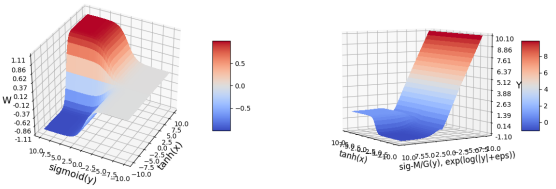

Fig. 3. **Left:** Neural accumulators (NACs) are biased towards learning [-1, 0, 1] as highlighted with large plateau regions around these values. This means its outputs are either addition or subtraction of input vectors not scaling. **Right:** Approximate surface curve of neural arithmetic logic units (NALUs) with some dimensional constraints for 3-D plotting. It highlights the ability to scale, along with earlier numerical biases around [-1, 0, 1] as shown with plateau region surfaces.

but same structures. These are trained with two randomly generated inputs from uniform distribution a and b with each having 2^{14} data points for training. Prediction capabilities on test data with values ranging up to 10 times the training range are evaluated. Refer figure 4 to observe architecture for both these trained models in this comparative experiment.

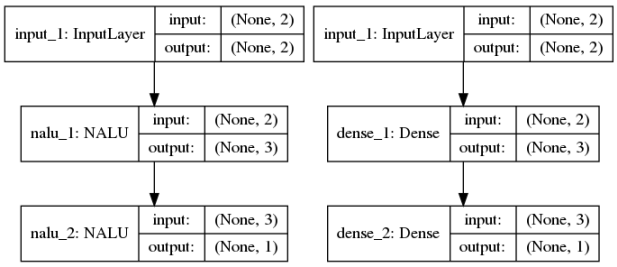

Fig. 4. The model on *left* represents neural network with NAC or NALU units and on *right* a multi-layer perceptron (MLP) which will be trained with different activation functions. Each variant of above models have two inputs, three hidden units, one output.

Comparative analysis is summarized in table I with mean absolute error (MAE) as an accuracy measure for MLP variants, NAC, NALU and its variants with changed learned gate for extrapolation. Also, in NALU-Tanh and NALU-Hard Sigmoid the learning gate *g's* is changed to observe any improvements in NALU's performance based on this change.

From these results stated in table I and visualized in figure 5 we conclude that Linear, LeakyReLU, ReLU activations and NAC, NALU, NALU-Tanh modules were the top performers in extrapolation task for numeric addition operation task. Hence, these top performers are used further in cell-counting task on synthetic dataset for learning end-to-end counting mechanism.

C. Counting experiment

[1]

In this experiment section the first subsections elaborates the datasets used for training and validation of our trained models, plus the data augmentation techniques used in our experiment. After that we elaborate onto different architectures

[1]*Code repository:* https://github.com/ashishrana160796/nalu-cell-counting

TABLE I
COMPARATIVE RESULT SUMMARIZATION FOR MULTIPLE MODELS

Layer Configuration/Activations	Mean Absolute Error (*a+b*)
Linear MLP	3.63×10^{-6}
Sigmoid MLP	29.830
Tanh MLP	15.743
ELu MLP	0.019
ReLU MLP	0.001
Leaky ReLU MLP	9.83×10^{-4}
PReLU MLP	0.001
NAC	2.70×10^{-6}
NALU	2.71×10^{-6}
NALU-Hard Sigmoid	3.24×10^{-6}
NALU-Tanh	3.18×10^{-6}

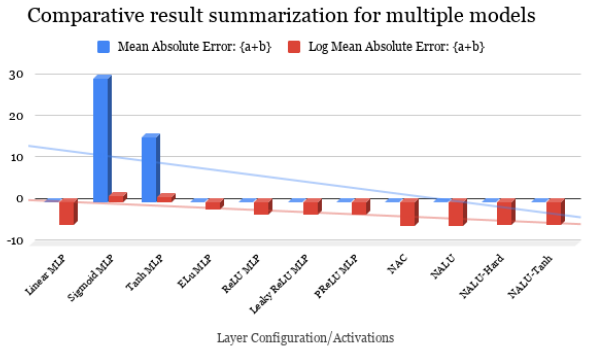

Fig. 5. Above visualization of Mean Absolute Error and $\log_{10}$(Mean Absolute Error) for different models learning identity mapping, demonstrates that our standard activation functions like *sigmoid* and *tanh* doesn't perform that well with higher data ranges during testing. Whereas, *NACs* and *NALUs* clearly overpowers the identity learning task and *linear*, *relu* based activation functions somewhat manages to provide sane results.

used for training having different activation layers on standard architectures and residual concatenated connection modules on modified proposed model architectures.

1) Datasets and data augmentation: Synthetic dataset which is generated by system [28]. 200 highly-realistic synthetic fluorescence microscopic images of bacterial cells are used for experimentation with a 75/25 train-test split for training each model architecture and its variants. Images are having average of 17464 cells.

For validation of trained models and checking true generalization capabilities we use BBBC005 from the the Broad Institutes Bioimage Benchmark Collection [29]. 600 images have a corresponding foreground mask are classified as completely in-focus for ground truth. We take a subset of this dataset with highly focused *F1* images only and coalesce image into one after replicate same image 16 times with some padding around each sub-image. After these changes the ground truth values are accordingly changed and then image is resized to the same dimensions as original dataset, see figure 6.

Data augmentation with elastic deformations to training images is applied for teaching network the desired invariance and

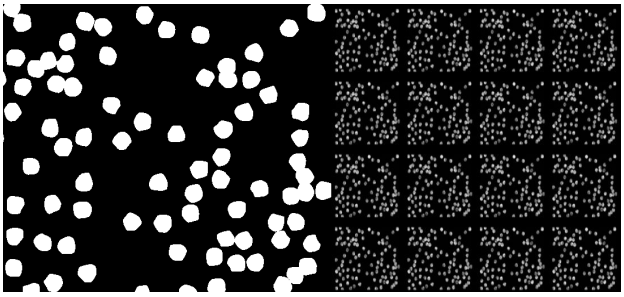

Fig. 6. *Left:* Original image from BBBC005 dataset. *Right:* Repeated image generated from a *4X4* grid repetition operation.

robustness properties, like specified in figure 8. These elastic deformations are introduced in the form of angular shear in the training images. Translation and rotation invariance along with robustness to gray value variations and deformations is main focus of augmentation process for microscopic images. Disfigurement using random displacement vectors on a coarse 3x3 grid were also generated. These data augmentation techniques especially are helpful for our custom data which is created just by repeating the original image in order to supplement a more robust dataset for the model to train on.

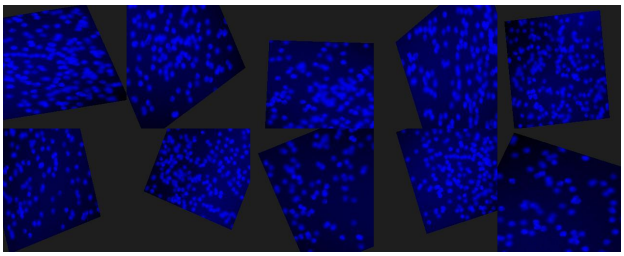

Fig. 7. This figure specifies ten different types of elastic deformations for the input images and corresponding target images are also deformed in such manner for more robust training.

2) Defining regression task and architecture details: In training dataset ground truth is provided as dot annotation corresponding to each cell. For training, dot annotations are represented by Gaussian and density surface D(x) which is formed from superposition of Gaussians. The optimization task is to regress density surface from corresponding image I(x). This is achieved by training convolutional neural networks (CNN) using mean square error between output heat map and target density surface as the loss function. Hence, at inference given an input I(x), the model predicts density heat map D(x).

FCRNs are inspired from VGG-net, we only used small kernels of size 3x3 pixels. Feature maps are increased for avoiding spatial information loss. Activation layers like convolution-ReLU-Pooling are popular in CNN architectures [23]. Here, we have altered these layers to create different activation maps which contains some numerical bias in the form of residual connections and regularization by batch normalization. The first layers contains convolutions-pooling operations, then we undo spatial reduction by upsampling operations for learning end-to-end training. Also, for dimensional compatibility of

residual NAC or NALU modules we did pooling and upsampling operations on these residual modules after batch normalization. See figure 8 for comparison between earlier original model and newly proposed architecture along with parameter details.

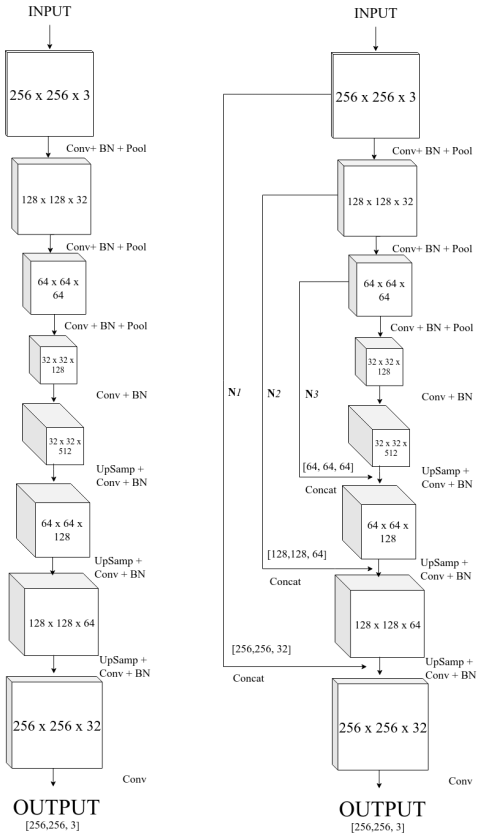

Fig. 8. *Network Structure Left:* FCRN with 3x3 convolution operations. *Network Structure Right:* FCRN added with residual concatenation connections of **N*i*** numerically biases units. Also, for dimensional compatibility **N*i*** residual layers are fed to corresponding main network layers and after that normalized for regularization. At last, 1x1 **Conv** operation output in the form of density map of result image is generated .
Conv + BN + Pool: A 3x3 convolutional operation with batch normalization regularization and 2x2 max pooling layer.
Unsample+Conv + BN: Unsampling + Convolutional operation with batch normalization regularization.
Concat: Feature maps concatenated along depth dimension.
N*i*: NAC or Variants of NALUs as residual concatenated connection.

U-net is modified upon the previously discussed FCRN architecture by having large number of feature channels for upsampling to propagate context information to high resolution layers. That makes expansive path almost symmetric to contracting path yielding a u-shape. Similar to above FCRNs optimization problem formulation remains the same, residual concatenated connection addition with NACs and NALU units along with batch normalization is done. Also, U-net architecture used in this paper is more computationally expensive than FCRN having approximately thrice the number of parameters leading to more feature learning capacity. See figure 9 for comparison between earlier original U-net model and newly proposed architecture along with parameter details.

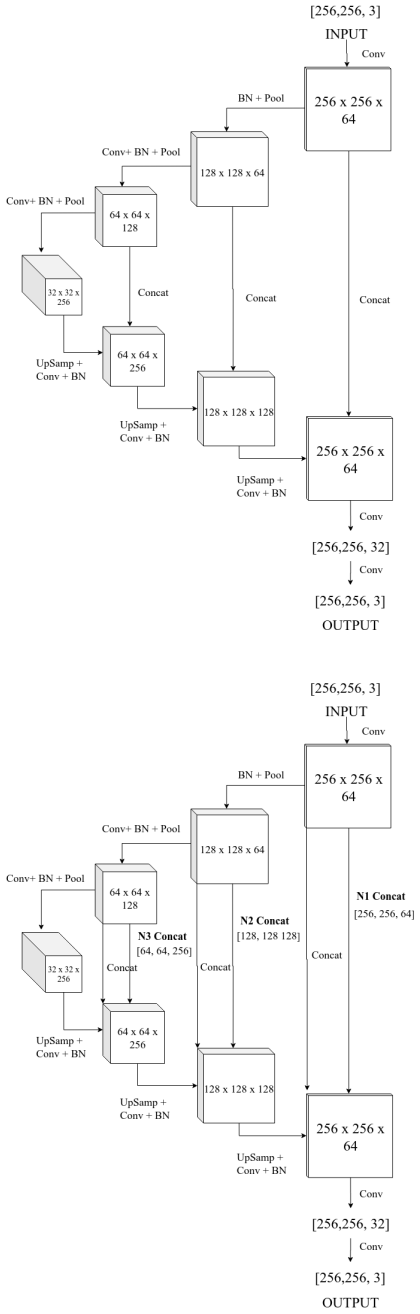

Fig. 9. *Network Structure Top:* U-net with 3x3 convolution operations and increasing depth of filters from 64 to 256 for feature abstraction & learning and after that these layers are fed to upsampling layers. *Network Structure Bottom:* U-nets added with residual concatenation connections of **N*i*** numerically biased units and after concatenation they are batch normalization for regularization. Finally, the last 1x1 **Conv** operation output in the form of density map is generated. Operations annotations are same as mentioned in figure 8

For the concatenation of residual connections of these units the dimensional consistency is maintained by added pooling and upsampling operations accordingly for these units to merge with the base network. FCRNs implementation resembles that of MatConvNet [30] as upsampling in Keras

is implemented by repeating elements, instead of bilinear sampling. In U-nets [8], for implementation low-level feature representations are fused during upsampling, aiming to compensate the information loss due to max pooling.

IV. Results

Mean absolute error (MAE) is the metric used in this paper for measuring results for cell counting on the synthetic cell dataset [28] and custom BBBC005 synthetic modified high cell count validation dataset.

- **Mean Absolute Error (MAE):** Mean Absolute Error (MAE): The mean absolute error is an average of the difference between the predicted value and true value.

$$AE = \|e_i\| = \|y_i - x_i\| \tag{1}$$

$$MAE = \sum_{i=1}^{n} \frac{|e_i|}{n} \tag{2}$$

- **Relative Improvement Percentage** Relative Improvement Percentage (RIP): Here, in context of this paper it defined as percentage improvement in MAE of a given model with respect to baseline ReLU models for FCRN and U-net architectures. In below equation, M_r is MAE from baseline ReLU model and M_i is model under consideration.

$$RIP_{\%} = ((M_r - M_i)/M_r) * 100 \tag{3}$$

Result table II compares earlier FCRN, U-net architectures with new numerically biased ResNet like connection modules with NACs and NALUs units under current training setup. With our setup we able to obtain similar results as mentioned in earlier reference papers and also we have equipped earlier model architectures with different regularization activations as specified in the table. From earlier ReLU implementation clearly Linear and LeakyReLU activation regularization based models have performed well. Also, for both model structures NAC and NALUs residual modules have outperformed all the earlier specified regular FCRN architecture. And similar arguments and results are extended by U-net model results where NALU layer concatenation based U-net outperforms all the models trained for our experiment.

TABLE II
Result summarization for trained models

FCRN-Models	MAE	U-Net-Models	MAE
ReLU	3.43	ReLU	1.78
LeakyReLU	3.39	LeakyReLU	1.74
Linear	3.34	Linear	1.73
NALU-tanh	3.21	NALU/tanh	1.56
NALU	**3.17**	NALU	**1.42**
NAC	3.23	NAC	1.63

Result table III compares performance of above trained models on a new validation dataset containing much higher cell counts for measuring performance on extrapolation capabilities counting tasks. For validation set we have used 300 images of size 256x256 pixels with cell counts averaging around 120012. Here also, NAC and NALU based residual concatenation module based models outperforms earlier architectures for counting tasks. This time relative improvement in even more for FCRN and U-net models showcasing better generalization abilities of trained models.

TABLE III
Validating trained models for extrapolation cell counting tasks

FCRN-Models	MAE	U-Net-Models	MAE
ReLU	3.04	ReLU	2.87
LeakyReLU	2.99	LeakyReLU	2.62
Linear	2.85	Linear	2.47
NALU-tanh	2.32	NALU-tanh	1.95
NALU	**2.27**	NALU	**1.87**
NAC	2.40	NAC	1.92

Relative improvement in predictions is visualized in figure 10 against ReLU based regularization as base result for comparison with other regularization layer based changes in FCRNs & U-nets and concatenation layer NALU/NAC residual connection addition in FCRNs & U-nets. It includes averaged out comparison from multiple executions of training and testing runs for both interpolation testing and extrapolation validation counting tasks for FCRN and U-net variant models with respect to ReLU based FCRN and U-net model. From, this figure it is clearly highlighted that models with NAC and NALUs residual modules have better generalization capabilities for extrapolation counting tasks i.e. they are better generalizers for this given cell counting task with increase in relative improvement in prediction as compared to base ReLU implementation. This figure shows more increase in relative improvement as we move right towards horizontal axis for both testing and validation task with extrapolation where in validation extrapolation task NAC/NALU models performing even better than testing data from which we can conclude that trained models are having better generalization abilities with some learned numerical bias in their trained weights with which even better predictions for higher count cells is made.

V. Summary

We were able to show that addition of newly proposed NACs and NALU units in existing architectures in the form of residual concatenation connection layer modules achieves better results. With numerically biased residual connections, higher accuracy for more dense images having higher counts of cells is achieved. Hence, producing more generalized cell counters that provides better predictions for real life use-cases. Finally, for code implementation details and other extra experimental results refer to this paper's github repository.

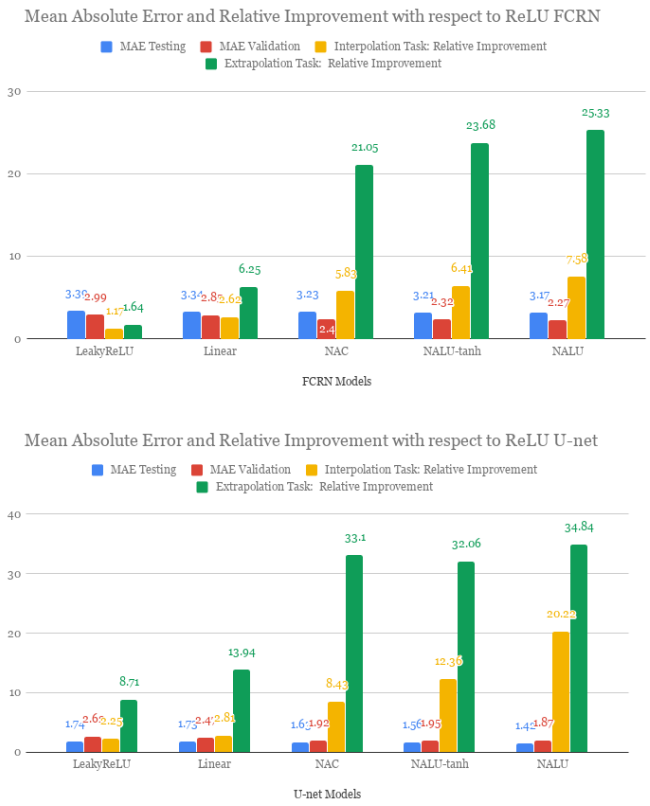

Fig. 10. Moving towards right in above figure and measuring percentage relative improvement metric there is sharp increase with NALU and NAC units based models highlighting increase in relative improvement with respect to baseline ReLU models is even more for extrapolation tasks. For interpolation task there is highest 9% improvement and for extrapolation there is 23% improvement with NAC units as residual connection.

REFERENCES

[1] Baygin, Mehmet, et al. "An Image Processing based Object Counting Approach for Machine Vision Application." arXiv preprint arXiv:1802.05911 (2018).

[2] Xie, Weidi, J. Alison Noble, and Andrew Zisserman. "Microscopy cell counting and detection with fully convolutional regression networks." Computer methods in biomechanics and biomedical engineering: Imaging & Visualization 6.3 (2018): 283-292.

[3] Fodor, Jerry A., and Zenon W. Pylyshyn. "Connectionism and cognitive architecture: A critical analysis." Cognition 28.1-2 (1988): 3-71.

[4] Marcus, Gary F. The algebraic mind: Integrating connectionism and cognitive science. MIT press, 2018.

[5] Trask, Andrew, et al. "Neural arithmetic logic units." Advances in Neural Information Processing Systems. 2018.

[6] Dehaene, Stanislas. The number sense: How the mind creates mathematics. OUP USA, 2011.

[7] Gallistel, C. Randy. "Finding numbers in the brain." Philosophical Transactions of the Royal Society B: Biological Sciences 373.1740 (2018): 20170119.

[8] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[9] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[10] Arteta, Carlos, et al. "Interactive object counting." European conference on computer vision. Springer, Cham, 2014.

[11] Chan, Antoni B., Zhang-Sheng John Liang, and Nuno Vasconcelos. "Privacy preserving crowd monitoring: Counting people without people models or tracking." 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008.

[12] Segu, Santi, Oriol Pujol, and Jordi Vitria. "Learning to count with deep object features." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2015.

[13] Lempitsky, Victor, and Andrew Zisserman. "Learning to count objects in images." Advances in neural information processing systems. 2010.

[14] Zhang, Cong, et al. "Cross-scene crowd counting via deep convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[15] Hernndez, Carlos X., Mohammad M. Sultan, and Vijay S. Pande. "Using Deep Learning for Segmentation and Counting within Microscopy Data." arXiv preprint arXiv:1802.10548 (2018).

[16] Paul Cohen, Joseph, et al. "Count-ception: Counting by fully convolutional redundant counting." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[17] He, Kaiming, et al. "Identity mappings in deep residual networks." European conference on computer vision. Springer, Cham, 2016.

[18] Srivastava, Rupesh Kumar, Klaus Greff, and Jrgen Schmidhuber. "Highway networks." arXiv preprint arXiv:1505.00387 (2015).

[19] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[20] Brunton, Steven L., Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems." Proceedings of the National Academy of Sciences 113.15 (2016): 3932-3937.

[21] Arteta, Carlos, et al. "Interactive object counting." European conference on computer vision. Springer, Cham, 2014.

[22] Fiaschi, Luca, et al. "Learning to count with regression forest and structured labels." Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012). IEEE, 2012.

[23] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[24] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[25] Ciresan, Dan, et al. "Deep neural networks segment neuronal membranes in electron microscopy images." Advances in neural information processing systems. 2012.

[26] Cirean, Dan C., et al. "Mitosis detection in breast cancer histology images with deep neural networks." International Conference on Medical Image Computing and Computer-assisted Intervention. Springer, Berlin, Heidelberg, 2013.

[27] Ning, Feng, et al. "Toward automatic phenotyping of developing embryos from videos." IEEE Transactions on Image Processing 14 (2005): 1360-1371.

[28] Lehmussola, Antti, et al. "Computational framework for simulating fluorescence microscope images with cell populations." IEEE transactions on medical imaging 26.7 (2007): 1010-1016.

[29] Ljosa, Vebjorn, Katherine L. Sokolnicki, and Anne E. Carpenter. "Annotated high-throughput microscopy image sets for validation." Nature methods 9.7 (2012): 637-637.

[30] Vedaldi, Andrea, and Karel Lenc. "Matconvnet: Convolutional neural networks for matlab." Proceedings of the 23rd ACM international conference on Multimedia. ACM, 2015.