Comparative Results of VGG16 & RESNET50 for a Cell Counting Regression Task

Amal Alshammary, Iowa State University, Ames, Iowa 50011, USA
Geng Ding, Iowa State University, Ames, Iowa 50011, USA
Elvis Kimara, Iowa State University, Ames, Iowa 50011, USA

1. INTRODUCTION

In the field of medical diagnostics, the precise and rapid counting of cells in images is a significant challenge. Traditional methods, often manual and time-intensive, are prone to inaccuracies, limiting their utility in urgent diagnostic scenarios. The emergence of machine learning, with a particular emphasis on regression techniques, presents a promising avenue for the automation of cell counting, aiming to substantially enhance the accuracy and speed of this essential task.

In this context, we present our innovative approach to developing an artificial intelligence (AI) model that effectively automates cell counting in high-resolution images of 800 pixels by 600 pixels. Our research is rooted in the IDCIAv2 dataset—a meticulously annotated collection of cell images compiled by biology students at Iowa State University. This dataset, consisting of 250 images paired with their corresponding cell count data in CSV format, forms the bedrock of our investigation. Complicating our research are challenges intrinsic to the dataset: its right-skewed distribution with a preponderance of images featuring minimal cell counts, its modest size of merely 250 images, and the inconsistent quality across the collected images. These hurdles underscore the necessity of a robust model that is adept at managing such variances.

To address these issues, we introduce an incremental strategy to construct a dependable cell-counting AI, illustrated in Figure 1. Our methodology is characterized by the following innovations:

- A probability function inversely related to cell count in the images, ensuring a balanced representation in the sampling pool and mitigating the skewness of the dataset.
- A versatile augmentation algorithm that dynamically applies a spectrum of image transformations, including flipping, rotation, blurring, and noise induction, to augment the dataset while preserving the integrity of the cell images.
- The strategic application of transfer learning, harnessing the capabilities of two pre-trained models—VGG16 and ResNet50—to lay the foundation for our machine learning endeavors.

Our experimental design employs a stepwise approach, allowing us to dissect and evaluate the impact of each individual component within our data preprocessing and model training pipeline.

The complete codebase and the trained models are accessible at our GitHub repository here.

In the ensuing sections, we will delve into the proposed work, where we'll talk a little more about our approach; the experimental setup, where we articulate the framework for our comparative analysis; followed by a thorough exposition of our results.

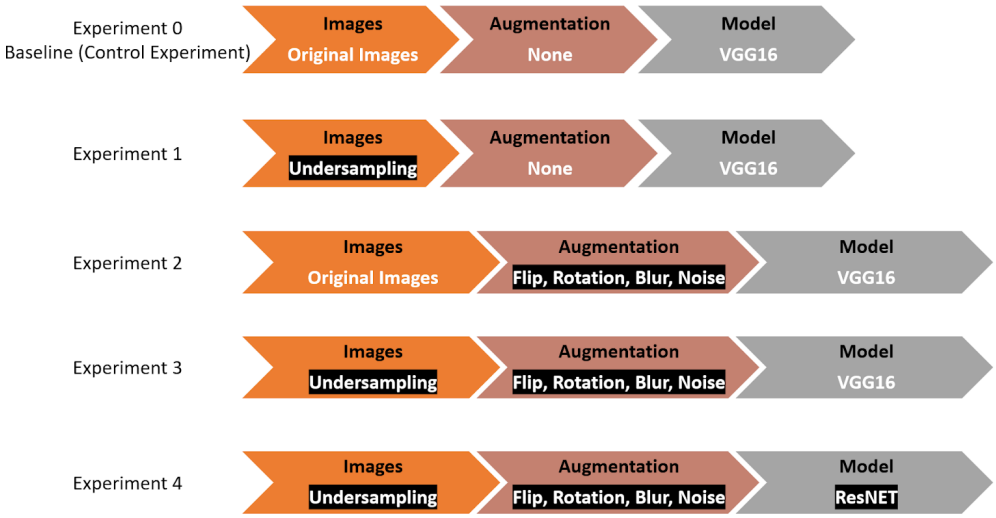

Figure 1: Overview of the experimental design, illustrating the stepwise approach from the baseline to the deployment of advanced machine learning models.

2. PROPOSED WORK

Our proposed methodology is structured as follows:

- Baseline (Control Experiment, Experiment 0): We begin by establishing a performance baseline using the original dataset to train the VGG16 model without preprocessing of the dataset and without any form of image augmentation. We used the “control experiment” concept in biology, and it will serve as a baseline for comparing the effects of subsequent manipulations.

- Experiment 1 – Dataset Balancing: In response to the challenges posed by imbalanced datasets, we implement an undersampling strategy for the first experiment. Inspired by Branco et.al., we designed an algorithm to select images according to the cell count. We aim to create a more balanced dataset that could potentially lead to more accurate and generalizable results.

- Experiments 2 & 3 – Augmentation Techniques: Taking cues from the seminal work on image augmentation, we proceed with Experiments 2 and 3, where the original and undersampled datasets are subjected to a series of augmentation techniques—flips, rotations, blurs, and the introduction of noise. These experiments are designed to test the hypothesis both undersampling and augmentation can effectively improve the model robustness.

- Experiment 4 – Model Comparison: This experiment introduces the ResNET50 architecture into our experimental lineup. Using the undersampled dataset with the full spectrum of augmentation techniques, we

aim to compare its performance against the VGG16 model. This head-to-head comparison will help us understand the strengths and weaknesses of each architecture for the specific task of cell counting.

Methodological Approach
For each experiment, we will:

- Preprocess the Data: Images will be preprocessed to ensure uniformity in size and color normalization. For undersampled datasets, a probabilistic method will be used to select which images to exclude, aiming to retain the richness of the dataset while mitigating the effects of imbalance.
- Apply Augmentation: A consistent set of augmentation techniques will be applied across experiments to ensure comparability. The choice of augmentations—flip, rotation, blur, and noise—is intended to simulate the variations encountered in real-world medical imaging.
- Train the Models: Both VGG16 and ResNET50 will be fine-tuned on the preprocessed and augmented datasets. We will monitor for overfitting and adjust hyperparameters accordingly to optimize each model's performance.
- Evaluate and Compare: Performance will be evaluated using metrics appropriate for regression tasks, such as Mean Absolute Error (MAE) and Mean Squared Error (MSE). We will also consider additional metrics that may offer insights into the models' behavior in a medical context.

Expected Outcomes
Through this rigorous experimental approach, we expect to:

- Gain insights into the effects of dataset balancing and image augmentation on model performance.
- Determine the most effective data preprocessing strategies for cell counting tasks.
- Compare the performance of VGG16 and ResNET50 in a controlled setting to identify the more suitable architecture for our application.

The findings from these experiments will not only contribute to the field of medical image analysis but also provide a reference for future research in applying machine learning to complex biomedical tasks.

3. EXPERIMENTAL SETUP

3.1 Datasets.

Table 1. shows statistics from the IDCIAv2 dataset, which consists of 250 images. On average, each image contains about 198 cells. It's important to note that these cells are not uniformly distributed. The number of cells per image varies significantly as shown in figure 2 making the dataset challenging.

Antibody	**No. of Images**	**Mean Cell Count per Image ± Std**
DAPI	250	198 ± 202

Table1. Statistics of IDCIAv2. Image resolutions: 800 pixels x 600 pixels. Range of number of cells per image [0, 979].

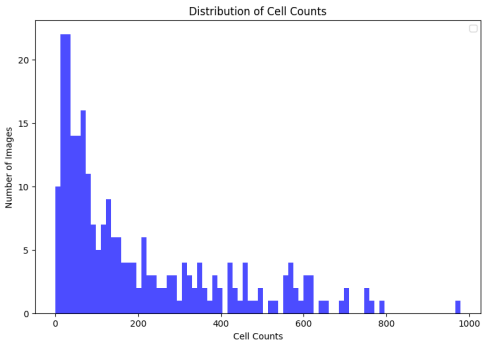

Figure 2. Distribution of the original dataset.

The original dataset exhibits a right-skewed distribution. Images with low cell count are very abundant but images with high cell count are notably scarce.

3.2 Comparison Methods and Experimental Setup.

In this section, we detail the methodologies employed for comparative analysis and the experimental setup.

Undersampling function:

For creating a trustworthy cell-counting machine learning strategy, we want to provide an unbiased dataset. Due to the right-skewed distribution of the original dataset, we propose that if we train our models on these data, the models will be biased toward low-count prediction. Thus we designed an algorithm to select k images based on the ground truth of the cell count. The basic idea is images with lower cell count will have a lower chance of being selected in the dataset for further training and testing. However, using probability will inevitably choose repetitive images at low probability. We implemented the algorithm such that no repetitive images will be chosen. The time complexity is *O(n)* in the worst case. The pseudo-code of this algorithm is shown below, and a more detailed implementation is in our code that is deposited in Git Hub.

UnderSampling(images, k)
1 **sort** images **in ascending order based on the counts**
2 **for each** image **in** images
3 probability←(rank + 1) / sum of all ranks
4 k images← **select** k image **in** images **with** probability
5 **return** k images

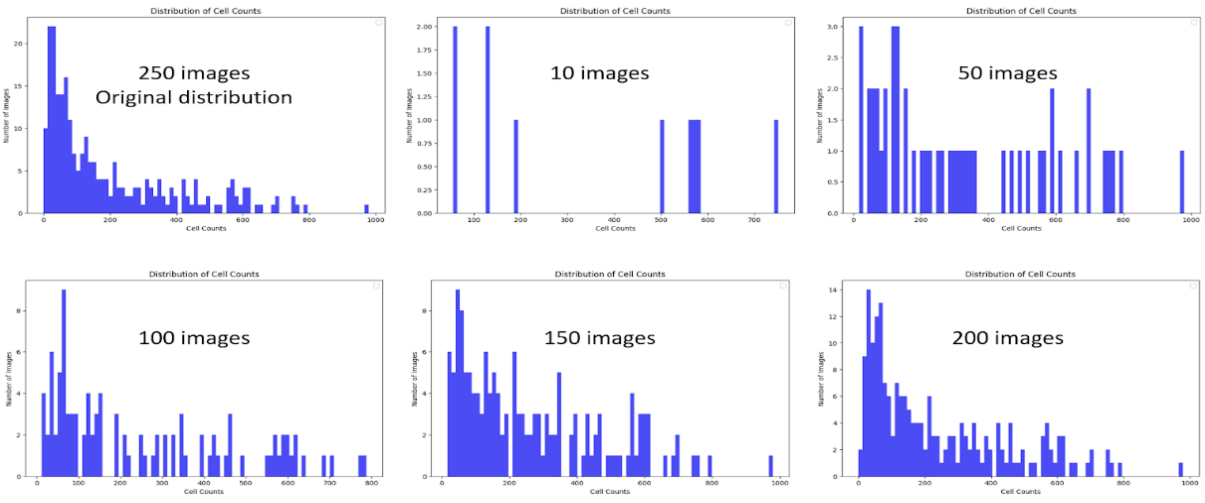

Figure 3. Distribution comparison after applying the undersampling algorithm.

After applying the undersampling algorithm, the distribution of the dataset improved. When we select a smaller number of images, the distribution exhibits a more uniform distribution.The correction to the right-skewness of the distribution is inversely related to the number of the images as shown in Figure 3. However, it is commonly known that training with too few images would end up with an overfitting model which will not perform well for unseen images. Therefore, we chose 150 as the size of our dataset to balance the size of the dataset and the distribution of the dataset.

Augmentation function:

The Need for Augmentation

Given the challenges presented by the limited size and skewed distribution of the IDCIA v2 dataset, it was imperative to enrich the dataset without compromising the quality of the images or the integrity of the cell counts. Augmentation techniques serve as a bridge to this gap, providing a means to artificially expand the dataset and introduce a variety of transformations that enhance the robustness of the machine learning models.

Designing the Augmentation Strategy

We developed a customized augmentation function that employs a probabilistic approach to randomly apply a set of transformations to each image. The augmentation process includes flipping, rotation, blurring, and adding noise—each selected to mirror potential real-world variations in medical images. These techniques not only diversify the visual data but also simulate potential deviations encountered during cell image acquisition, such as changes in orientation, focus, and image quality.

Implementation

The augmentation function was implemented in Python using the PIL library, which provides a comprehensive suite of image-processing capabilities. Our function, augment_image, takes an input image and applies at least one random transformation to it, ensuring that each augmented image retains the fundamental characteristics necessary for accurate cell counting.

For each original image, the function decides whether to:

- Flip the image horizontally, simulating the lateral inversion of cells.
- Rotate the image by 180 degrees, representing a complete inversion, a common occurrence in slide preparations.
- Apply a blur filter, mimicking the slight out-of-focus scenarios a microscope might present.
- Add Gaussian noise, reflecting the electronic noise or grain that can be present in digital imaging.

Each transformation is applied with a 50% probability, and at least one transformation is guaranteed to ensure that the augmentation process yields a result distinct from the original image.

Batch Augmentation Process

To augment the entire list of images, we crafted the augment_image_list function. It iterates through all the 150 images in our undersampled dataset and, using the augmentation multiplier, generates a specified number of augmented versions for each original image. This function not only populates the new dataset with these augmented images but also maintains a balance between the original and transformed images.

The process ensures that the augmented dataset reflects a realistic variety of cell imaging conditions, thereby training the AI models to be adaptive and accurate across a spectrum of potential real-world scenarios.

Augmentation Outcomes

The augmentation process with a multiplier of 3 successfully generated a dataset of 600 images (450 new, 150 original). This was significantly larger and more diverse. This enriched dataset was crucial in training the VGG16 and ResNet50 models, leading to improved model performance as observed in subsequent experimental results. The augmentation techniques proved instrumental in mitigating the effects of overfitting, thereby enhancing the generalizability of the models.

Our Models:

VGG16-Based Regression Model

The VGG16 model, initially introduced for image classification, has been adapted for regression tasks. The model utilizes the pre-trained VGG16 architecture, comprising convolutional and pooling layers, followed by custom regression layers(Figure 3). The VGG16 features are frozen to retain learned representations, and the network is extended with a fully connected regression head consisting of Flatten, Linear (from 51277 to 256), ReLU, and a final Linear layer (from 256 to 1) to output the regression value.

ResNet50-Based Regression Model

The ResNet50 model, another prominent architecture in image classification, is also adapted for regression purposes. This model begins with a pre-trained ResNet50 base, excluding the original fully connected layers. Like VGG16, the pre-trained parameters are frozen. The regression head for ResNet50 includes Flatten, Linear (from 2048 to 256), ReLU, and a final Linear layer (from 256 to 1) (Figure 3).

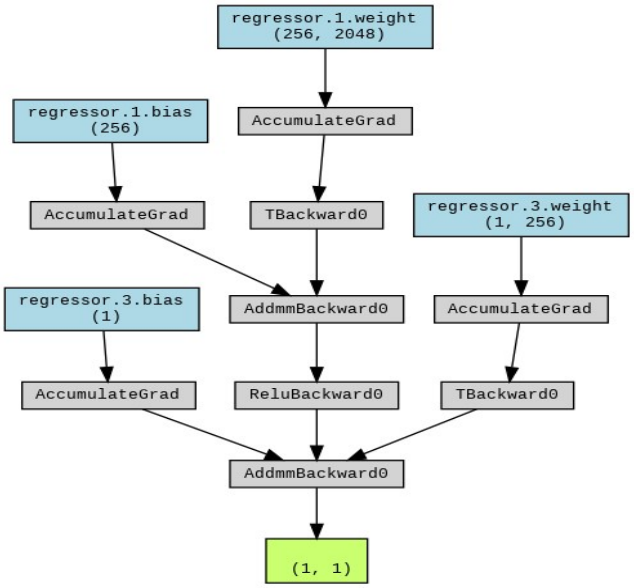

Figure 4. The customized layers for regression in both VGG16 & ResNet50

Hyperparameters and Training

In our comparative analysis of the VGG16 and ResNet50 models adapted for regression tasks, both models are uniformly trained using the Adam optimizer set at a learning rate of 0.001. For performance assessment, particularly in the context of cell counting, we utilize a set of comprehensive metrics:

1. Mean Absolute Error (MAE): MAE measures the average magnitude of errors in a set of predictions, without considering their direction. It's calculated as the average of the absolute differences between the predicted values and the observed (actual) values. MAE gives an idea of how large the errors are on average and is particularly useful because it has the same unit as the data, making interpretation straightforward. (1)
2. Mean Squared Error (MSE): MSE calculates the average of the squares of the errors or deviations—that is, the difference between the estimator and what is estimated. It gives a measure of the quality of an estimator; the smaller the MSE, the closer we are to finding the line of best fit. MSE is sensitive to outliers and tends to penalize larger errors more heavily than MAE, as errors are squared before they are averaged. (2)
3. Acceptable Error Count Percent (ACP): Introduced specifically in this study[Mohammed,et..al.], ACP calculates the percentage of predictions that fall within a 5% error margin of the true counts, using Iverson brackets for binary evaluation of the error threshold. (3)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \quad (1)$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \quad (2)$$

$$ACP = \frac{1}{n} \times 100 \sum_{i=1}^{n}\{|\hat{y}_i - y_i| \leq 0.05 \cdot y_i\} \quad (3)$$

4. RESULTS

4.1 Experimental Results and Analysis

Experiment	**Test Loss**	**MAE**	**ACP 5%**
0: VGG16	6606.4	51.1	18.4
1: Undersampling + VGG16	3755.8	49.0	4.4
2: Augmentation + VGG16	3551.0	38.7	19.5
3: Undersampling + Augmentation + VGG16	4995.0	46.9	27.8
4: Undersampling + Augmentation + ResNet50	4591.7	47.1	11.9

Table 2. Summary of experimental results.

Based on (Table 2), which summarizes the results of our experiments involving different combinations of undersampling, augmentation, and neural network architectures (VGG16 and ResNet50), we can draw some conclusions about the scenarios in which the method performs well and where it might not be as effective.

- Experiment 0, which uses neither undersampling nor augmentation and employs VGG16, has the highest test loss (6606.4). This could indicate that the base model without any data preprocessing or enhancement techniques is insufficient for the task.

There's a trade-off visible between different metrics. For instance, a lower test loss doesn't always coincide with a lower MAE or an ideal ACP 5% value, indicating the complexity of optimizing all metrics simultaneously.

- Experiment 1, which combines undersampling with the VGG16 architecture, demonstrates interesting outcomes when evaluated under the ACP metric. This experiment achieved an ACP of 4.4%, indicating that only a small percentage of the model's predictions fell within the acceptable 5% error margin of the true counts. This low ACP value, in conjunction with a test loss of 3755.8 and a Mean Absolute Error (MAE) of 49.0, suggests a limited ability of the model to make highly accurate predictions on the validation dataset.

- Experiment 2, employing data augmentation with the VGG16 architecture, shows a significant improvement in performance metrics. It records the lowest test loss (3551.0) and Mean Absolute Error (MAE) of 38.7 among all the experiments, alongside a high ACP of 19.5%. This indicates a substantial enhancement in the model's predictive accuracy and its ability to generalize well on validation data. The success of this experiment shows the effectiveness of data augmentation in enhancing model performance, particularly in achieving a balance between accuracy (low MAE) and generalizability (high ACP).

- Experiment 3, combines undersampling, augmentation, and the VGG16 architecture, leading to a mixed outcome. With a test loss of 4995.0 and an MAE of 46.9, the model performs less effectively on these metrics compared to Experiment 2. However, it achieves the highest ACP of 27.8%, suggesting a strong capability in making predictions within the acceptable error margin. This experiment highlights the complex interplay between different data preprocessing techniques and their impact on various performance metrics, illustrating the challenge of optimizing them simultaneously.

- Experiment 4, explores the combination of undersampling, augmentation, and a switch to the ResNet50 architecture. This setup results in a test loss of 4591.7 and an MAE of 47.1, which are improvements over Experiment 3 but still, it is not the optimal choice. The ACP stands at 11.9%, indicating a moderate level of prediction accuracy within the desired error margin.

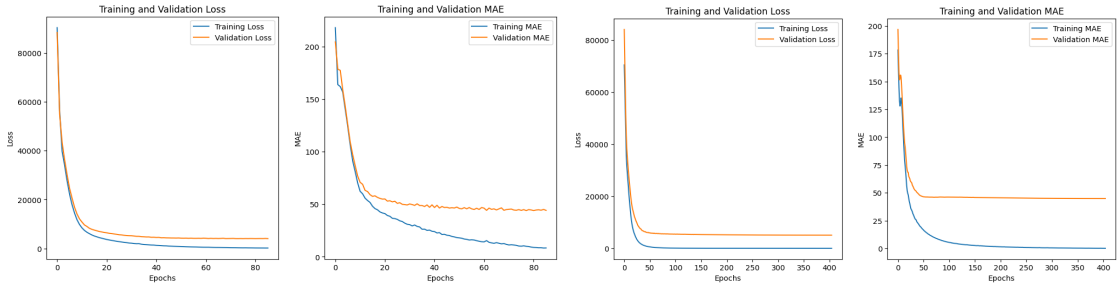

Figure 4. Training and validation Loss and MAE results of experiment 3 (Left) and control/baseline Model (Right)

To sum up, our extensive experiments with undersampling, augmentation, and different neural network architectures have revealed crucial insights into optimizing machine learning models. The results indicate that both undersampling and augmentation can significantly enhance model performance. However, the results also imply that merely employing these techniques is not a cure-all solution. To achieve the best results, it is imperative to fine-tune hyperparameters and consider additional strategies, such as unfreezing layers in pre-trained models. This method might enhance the precision of the model by enabling a more detailed and tailored learning process from the given data set.

5. CONCLUSION AND FUTURE WORK

We successfully completed the task by addressing the challenges we described earlier. We provided a novel strategy to address the dataset imbalance problem. We deployed image augmentation techniques which significantly expanded the dataset size and improved model accuracy. We applied transfer learning with pre-trained models, e.g. VGG16 and RESNET50 and we demonstrated that these networks, when fine-tuned, can yield competitive results, simplifying the need for extensive literature review and complex model tuning while still offering an alternative to the tedious and long process of manually counting the cells.

The use of VGG16 and RESNET50 has been beneficial, but there is scope for exploring other pre-trained models or experimenting with deeper levels of fine-tuning. It is important to acknowledge that our study did not incorporate a specific method for the selection of the best hyperparameters. In future research, implementing a systematic approach for hyperparameter optimization could further improve model performance by more accurately tailoring the model to the unique characteristics of the data. Moreover, given the complexity of neural networks, future work could also focus on making these models more interpretable and explainable. This is crucial for applications in sensitive fields where understanding model decisions is as important as the accuracy of predictions.

We have tried to implement a new metric which we named normalized MSE as shown below (4). However, due to time constraints, we were unable to deploy it functionally. We would like to evaluate this metric in future experiments.

$$Normalized\ MSE = \frac{1}{n}\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{y_i} \quad (4)$$

ACKNOWLEDGMENTS

This work is partially supported by National Science Foundation Award No. 2152117. Findings, opinions, and conclusions expressed in this paper do not necessarily reflect the view of the funding agency. We thank Prof. Surya Mallapragada, Prof. Donald Sakaguchi, Catherine Fonder, Abdurahman Ali Mohammed, Madison K Chng, Noah M Shepardson, and a team of undergraduate students who contributed to the data acquisition process. Specifically, we want to express our sincere thanks to Dr. Wallapak Tavanapong and Abdurahman Ali Mohammed for their great effort in designing the project, direction of the research, and invaluable assistance.

REFERENCES

Abdurahman Ali Mohammed, Catherine Fonder, Donald S. Sakaguchi, Wallapak Tavanapong, Surya K. Mallapragada, and Azeez Idris. 2023. IDCIA: Immunocytochemistry Dataset for Cellular Image Analysis. In Proceedings of the 14th Conference on ACM Multimedia Systems (MMSys '23). Association for Computing Machinery, New York, NY, USA, 451–457. https://doi.org/10.1145/3587819.3592558

Paula Branco, Luis Torgo, Rita P. Ribeiro. Pre-processing approaches for imbalanced distributions in regression. Neurocomputing, Volume 343, 2019, Pages 76-99, https://doi.org/10.1016/j.neucom.2018.11.100.

Ilya Kostrikov, Denis Yarats, Rob Fergus. Image Augmentation Is All You Need: Regularizing Deep Reinforcement Learning from Pixels, https://arxiv.org/abs/2004.13649

Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition.2016 IEEE Conference on Computer Vision and Pattern Recognition