Imperial College
London

Final report for MSc Project

R – Net: A Novel Approach Towards The Cell Quantification Of Microscopy Images Using A Regression-based Network

Author name: Owais M Siddiqi

Supervisor(s): Prof Anil Bharath, Dr Faraz Janan

Submitted in partial fulfilment of the requirements for the award of MSc in Biomedical Engineering from Imperial College London

Sept. 2023 Word count: 5,993

Contents

Figures

Tables

Abstract

Microscopy images are vital in cellular biology and medical diagnostics, revealing intricate cellular structures and serving as potential early indicators of diseases. Historically, analysing these images involved manual inspection, but advancements in deep learning and artificial intelligence propose a more accurate and automated alternative. This study develops and introduces 'R – Net,' a bespoke regression-based Convolutional Neural Network (CNN), designed to automate microscopy image analysis by accurately predicting cell types without the conventional need for cell segmentation. Drawing inspiration from the human hierarchical visual system and pre - trained networks like VGG16 and ResNet, R – Net's architecture provides an alternative approach to pattern recognition within cellular structures. Trained on procedurally generated basic approximations of phase-contrast images, R – Net achieved a MSE of 0.00723 and an R^2 score of 0.7576 in a brief 1.5-hour training duration, demonstrating its adeptness in explaining 75.76% of the dataset's variability and surpassing the performance of pre-trained models and alternate architectures. This study highlights R – Net's role in refining the analysis of microscopy images and offering an independent alternative approach as a step towards aiding fault tolerance in cellular analysis.

Acknowledgements

First and foremost, I would like to thank my family for their constant, overwhelming love and support.

I would also like to thank my friends for providing me with entertainment, joy, and companionship throughout this process.

I would like to thank my supervisor Professor Anil Bharath for the generation of the artificial dataset used in this project.

And finally, I would like to thank my supervisors Professor Anil Bharath, again, and Dr Faraz Janan for their support, guidance, and encouragement throughout this project.

"Intellectual growth should commence at birth and cease only at death."

– Albert Einstein

1. Introduction

In cellular biology, medical diagnostics, and various research domains, microscopy images stand as a vital tool. For microscopy images to be used effectively, the accurate segmentation of cell structures in them is essential for different types of analysis [1], [2] .These images, often intricate and rich in detail, have facilitated our understanding of cellular structures and their complex interactions. Furthermore, various imaging techniques, including phase contrast and fluorescence imaging, can reveal diverse insights into the same cell compositions [3]. These viewpoints are useful in potentially detecting early stages of diseases, assessing drug efficacy, and evaluating the potential for tissue regeneration [2] .

Historically, the task of deciphering these images fell upon microbiologists, who, through keen visual inspection, estimated cell type differentiation. However, through technological advancements, such manual methods are becoming redundant.

Such technological advancements consist of the development and introduction of deep learning, specifically convolutional neural networks (CNNs). These networks have emerged and provided an alternative pathway to image segmentation, offering a nuanced approach to image analysis [4], especially in the last 50 years with one of its first introductions in 1980 [5]. These networks, developed through extensive training on comprehensive datasets, have the capability to discern patterns that are often subtle and not immediately apparent to the human eye. This is evident in tasks like cell counting and classification in microscopy images, as demonstrated in studies [6]–[9] among others. CNNs have demonstrated high levels of accuracy and have been efficient in tasks traditionally undertaken by humans. More precisely, in comparison to human experts, CNNs have exhibited increased accuracy and consistency in classification, detection, and diagnostics across a variety of image types. These advancements highlight the potential of CNNs in refining and automating image analysis in diverse fields, contributing and facilitating progress in disciplines such as biology and medicine.

The development of this model was not devoid of challenges. Manual techniques require a painstakingly labour-intensive process of data preprocessing [10] and manual microscopy cell analysis can be both tedious and inconsistent, varying with the expertise of the scientist. Deep learning automates the work and removes such labour, but also comes with its own difficulties. Moreover, the inherent 'black box' nature of these models presents another problem, interpretability, making it difficult to understand their internal mechanisms.

The significance of my report derives from aiding the work of traditional manual methods of cell type quantification in microscopy images by applying state-of-the-art automated techniques using deep learning and convolutional neural networks (CNNs). This method addresses critical challenges in terms of the manual analysis of microscopy images and the ability for a CNN to develop a distinction between the subtle differences of cells and, to differentiation them in one direct approach.

This project works on the development of a bespoke CNN (deemed R – Net), designed for a regression task: predicting the probability of different cell types present in microscopy images. Table 1 shows the processes used today for microscopy image analysis; this project's pipeline removes the need for image cell segmentation. A stage that even the most advanced image analysis pipelines utilise [11].

Manual cell segmentation is often prone to errors and inconsistencies due to the intricate and variable nature of cell structures, as well as image artifacts [12], [13]. This project helps to streamline the analysis pipeline. This saves time and also reduces the risk of human-induced errors, enhancing the overall accuracy of results. Leading to researchers and medical professionals possibly processing microscopy images faster and with a greater confidence in the data's reliability.

The removal of cell segmentation from microscopy analysis introduces an alternative approach to cell quantification and helps to optimise its speed and efficiency in areas such as cellular biology and

medical diagnostics. This approach processes image-based inputs directly into estimations of cell population without the otherwise necessary stage of segmentation and demonstrates an independent methodology, which differs from the two-stage learning introduced by DeFauw et al. [2]. DeFauw's paper denoted the potential of employing a two-stage segmentation process in aiding clinical decision-making processes. It enables the execution of diagnoses and referral recommendations for retinal diseases with a level of accuracy that is, in some instances, superior to that of human experts. Illustrating the benefits of two-stage segmentation processes in enhancing the accuracy and reliability of clinical assessments.

Table 1. Outline of the microscopy processes of both classical and this masters project goes through for microscopy image segmentation.

Steps	**Typical Microscopy Classification Pipeline**		**This Projects Microscopy Regression Pipeline**	
	Action Item	Next Step	Action Item	Next Step
1	Generate Images Based on Cell Types and Parameters	Software produces a set of artificially generated cell images through these parameters and their weights	Generate Images Based on Cell Types and Parameters	Software produces a set of artificially generated cell images through these parameters and their weights
2	Generate Images	Images are pushed through an image segmentation pipeline and pre - processed	The image is analysed using a bespoke regression CNN, with the labels being its percentage of different cell types	The CNN uses this data to quantify the cell types by percentage, learning different cell variations
3	The images are analysed, and relevant information is extracted from the data	This information is used by the CNN or ML technique to classify the cells based on their appearance (cell area, shape, and contrast)	A regression-based prediction from a test data set, outputting the proportion of different cell types present in the image	
4	Classification such as cell type is predicted on a test data, as well as additional information such as distribution and probability		N/A	

2. Background

2.1. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) have established themselves as a dominant force in the realm of image recognition. Their unique architecture allows them to learn spatial hierarchies automatically and adaptively from data, making them particularly adept at handling intricate image patterns [14].

Microscopy images often contain intricate details at various scales, from cellular structures to sub-cellular organelles. Traditional image processing techniques might struggle to consistently identify and analyse these features, especially given the variability and noise inherent in microscopy data and may introduce human error into the equation . CNN's can be used to combat this. Their ability to learn spatial hierarchies means they can discern patterns and structures at multiple scales, from the broadest cellular formations to the finest molecular details. However, CNN's can also be subjected to human error due to the training data they learn from. Since real world data is annotated by human professionals, the data itself is subjective and can lead to a bias in the CNN's training. This project solves this issue by using procedurally generated images derived from software, removing the human error from the equation.

2.2. CNN Architecture

Table 2. Structure of the R - Net Model

R - Net		
Layer Name	**Output Size**	**Block Structure**
Conv1	128 x 128	3 x 3, 16
Batch Norm	128 x 128	-
Conv2	128 x 128	3 x 3, 16
Batch Norm	128 x 128	-
MaxPool	64 x 64	2 x 2
Conv3	64 x 64	3 x 3, 32
Batch Norm	64 x 64	-
Conv4	64 x 64	3 x 3, 32
Batch Norm	64 x 64	-
MaxPool	32 x 32	2 x 2
Conv5	32 x 32	3 x 3, 64
Batch Norm	32 x 32	-
Conv6	32 x 32	3 x 3, 64
Batch Norm	32 x 32	-
MaxPool	16 x 16	2 x 2

Looking at

Appendix II, Table 2 and Figure 3, the architecture of the CNN was developed for a regression task, with microscopy images in mind. The hierarchical structure of the architecture was drawn from the intricate design of the human visual system, the initial layers of the network start off extracting simple features and patterns [14]. These levels of extraction get increasingly abstract the deeper the network goes, eventually working towards extracting complex patterns that could allude to high-level features, some of which may have been disregarded or overlooked by human researchers.

With respect to the architecture in

Appendix II and Figure 3, the design was created as follows:

- Convolutional Layers:

These are the foundations of CNNs. They apply a convolution operation on the input image data using filters. The aim is to detect specific features, such as edges, textures, and patterns of cells, gaining an overall "idea" of the image. Then, as you increase the number of filters, the layers detect progressively intricate details and specialised features that can be used to identify common characteristics to help generalisation and increase accuracy of predictions.

- Activation Functions:

Activation functions determine the output of a neuron based on its input. They introduce non-linear properties to the network, enabling it to learn from the error and make necessary corrections, which is essential for learning complex patterns.

- Batch Normalisation:

After each convolutional layer, batch normalisation is used. It standardises the activations of a given input before passing it to the next layer. This stabilises the learning process and also reduces the training time, allowing for higher learning rates [15] and ensuring efficient processing of microscopy images. Furthermore, batch normalisation can regularise training in the network and thus reduce the need for further regularisation such as dropout layers and L1 or L2.

- MaxPooling Layers:

These layers down sample the spatial dimensions of the input volume [16]. By retaining the most significant features and reducing computational complexity, they play a crucial role in pinpointing, and defining characteristics that help the model ensure generalisation. It helps concentrate information to ensure only the essential pattens are learnt and retained while the more insignificant features are discarded. Accelerating training and reducing the risk of overfitting by reducing the number of parameters and spatial size.

- Flatten Layer:

Post convolutional and pooling layers, the Flatten layer transitions the 2D matrix into a 1D vector [16], preparing the data for the dense layers.

- Dense (Fully Connected) Layers:

These layers interpret the features extracted by the convolutional layers [16]. In the provided architecture, a series of dense layers with decreasing neurons (128, 64, and 32) ensures a hierarchical interpretation of cellular features. By the progressive reduction of dimensionality, the information gathered is refined, and thus equips the network with the capabilities to make accurate, informed decisions and predictions based on the patterns discerned by the convolutional layers.

- Dropout Layers:

Overfitting is a common challenge in deep learning, particularly when working with diverse and intricate datasets like microscopy images. To combat this, dropout layers are strategically incorporated into the network. These layers function as a form of regularisation by randomly deactivating a certain percentage of neurons during the training phase [16].

- Output Layer:

The final Dense layer, featuring three neurons, is for a regression output to the microscopy images. Each neuron is designed to output the probability of the presence of one of the three distinct cell types, providing a percentage composition in the sample.

2.3. Optimisation Techniques in CNNs

2.3.1. Learning Rate Strategies

The learning rate is an important parameter in the training process of a neural network. It determines the level of adjustments to be made to the model's weights after each iteration. An appropriately set learning rate ensures a balance between rapid convergence and the stability of the model.

There are strategies to dynamically adjust the learning rate during the training process to achieve optimal performance:

- **Learning Rate Reduction:** This involves decreasing the learning rate at specific intervals or when the model's improvement reaches a plateau. By doing so, it allows the model to make larger adjustments initially for faster learning and then refine its weights with smaller adjustments as it gets closer to the optimal solution.
- **Schedulers:** These are algorithms designed to modify the learning rate based on certain conditions or benchmarks. For instance, if the model's accuracy doesn't improve for a set number of epochs, the scheduler might reduce the learning rate to encourage finer, more precise weight adjustments and potentially better results.

This approach ensures that the model remains adaptive and responsive throughout the training process, enhancing its ability to generalise from the training data to unseen data.

2.4. Learning Rate Optimisers

- **Adam (Adaptive Moment Estimation):** It maintains a moving average of past gradients and squared gradients [16], adjusting the learning rates of each parameter. This approach makes Adam a versatile optimiser for a wide range of tasks.
- **SGD (Stochastic Gradient Descent):** An optimisation method where each update is based on a single training sample [16]. There are different versions of SGD such as SGD with momentum, which introduces a factor of the previous gradient to the current one, providing a boost and smoothing the optimisation process.
- **RMSprop (Root Mean Square Propagation):** This optimiser enhances the robustness of gradient descent in the context of mini batch learning by adapting the learning rates during training. It utilises a moving average of the squared gradient [16] to normalise the gradient itself, dynamically adjusting the learning rates of each parameter. The adaptability of RMSprop makes it particularly effective for problems with non-stationary objectives and noisy datasets, allowing it to navigate efficiently through complex optimisation landscapes.

2.5. Pre – Trained Networks

Pre-trained networks are deep learning models that have already been trained on large natural image datasets (such as ImageNet dataset), typically for image classification tasks. These networks have learnt to identify various features from these datasets, making them valuable for transfer learning as their weights can be leveraged for a new, but related task.

VGG16 [17]: Developed by the Visual Graphics Group (VGG) at the University of Oxford, VGG16 is a deep convolutional neural network that consists of 16 layers. While it offers excellent

performance, its architecture is relatively straightforward. VGG16 was trained on the ImageNet dataset, enabling it to detect a wide range of features, from simple edges to complex patterns.

ResNet [18] (Residual Networks): Introduced by Microsoft Research, ResNet addresses the vanishing gradient problem encountered in previous deep neural networks by introducing skip or residual connections. These connections allow activations layers to bypass one or more layers, effectively enabling the training of very deep networks without the risk of performance degradation. The ResNet models include ones of varying depths, with ResNet-50, ResNet-101, and ResNet-152 being the most popular.

MobileNet [19]: Designed by Google, MobileNet is optimised for mobile and embedded applications. Its design reduces the computational cost and model size, making MobileNet highly efficient without compromising much on performance.

2.6. Evaluation Metrics

In deep learning, particularly when working with convolutional neural networks (CNNs), evaluation metrics serve as the measurement of a model's performance. While metrics such as accuracy, precision, recall, F1 and RUC curves are common metrics for classification tasks, they are not applicable for regression problems such as this. Instead, metrics such as MSE, MAE, MAPE, R^2 and RMSE are employed to gauge the model's predictive capabilities.

2.7. Mean Squared Error (MSE):

MSE is a one of the most common tools used in regression analysis. By squaring the differences between predicted and actual values, it gives more weight to larger errors, ensuring that the model is penalised by a larger margin for significant mispredictions.

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

y_i is the actual value (ground truth)

$\hat{y}_i$ is the predicted value

Σ denotes the sum of the value in the expression, for each value of i from 1 to n

n is the number of observations

i number of observations

A high MSE indicates a model that's off the mark, especially in its predictions of outliers. It's particularly valuable in contexts where large deviations from the truth can have severe consequences, ensuring that the model is trained to minimise such discrepancies.

2.8. Mean Absolute Percentage Error (MAPE):

MAPE provides a percentage-based perspective on prediction errors, allowing for a relative understanding of the model's accuracy and the relative magnitude of prediction inaccuracies.

$$MAPE = \frac{1}{n}\Sigma_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%$$

y_i is the actual value (ground truth)

$\hat{y}_i$ is the predicted value

Σ denotes the sum of the value in the expression, for each value of i from 1 to n

n is the number of observations

i number of observations

Being percentage based, MAPE is useful for datasets with values spanning multiple magnitudes. However, it is essential to note that MAPE can be sensitive when actual values are close

to zero, as this can lead to extremely high percentage errors. This problem occurs within this project as the actual values used are between 0 – 1 [20].

2.9. Mean Absolute Error (MAE):

MAE is a direct perspective on prediction errors, averaging the absolute differences between predictions and actual values [21]. Unlike MSE, it doesn't square errors, leading to a linear penalty for mistakes.

$$MAE = \frac{1}{n}\Sigma_{i=1}^{n}|y_i - \hat{y}_i|$$

y_i is the actual value (ground truth)

$\hat{y}_i$ is the predicted value

i number of observations

Σ denotes the sum of the value in the expression, for each value of i from 1 to n

n is the number of observations

By treating all errors uniformly, MAE is less influenced by outliers. It provides a clear, interpretable metric, indicating the average magnitude of errors the model is likely to make.

2.10. R-squared (R^2 - Coefficient of determination):

R^2 demonstrates how well the model's predictions match the actual results. A higher R^2 means the model does a better job at explaining the variance in the data.

$$R^2 = 1 - \frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{\Sigma_{i=1}^{n}\left(y_i - \bar{y}_i\right)^2}$$

y_i is the actual value (ground truth)

$\hat{y}_i$ is the predicted value

i number of observations

Σ denotes the sum of the value in the expression, for each value of i from 1 to n

n is the number of observations

$\bar{y}_i$ is the mean of the actual values

R^2 values close to 1 suggest a model that captures most of the variability in the target variable. At the other end, values near 0 indicate a model that fails to explain much of the target's variance. Negative R^2 values, though uncommon, can signal a model performing worse than a simple average [22].

2.11. Room Mean Squared Error

RMSE is the square root of MSE, very similar, however the square root provides a greater degree of interpretability as it is now same unit as the original data [21].

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}$$

y_i is the actual value (ground truth)

$\hat{y}_i$ is the predicted value

i number of observations

Σ denotes the sum of the value in the expression, for each value of i from 1 to n

n is the number of observations

A lower RMSE indicates better model performance. It's useful as it is a metric that's directly comparable to the data's scale.

Selecting the appropriate evaluation metric is imperative to a good deep learning model. While metrics like MSE and MAE provide insights into the model's error magnitudes, R^2 offers a comprehensive view of the model's overall predictive power. The choice of metric often depends on the specific problem and data at hand. For instance, in microscopy image analysis, where precision is of utmost importance, metrics that heavily penalise large errors might be preferred, guiding further refinement and optimisations of the network.

2.12. Loss Functions

In the development of deep learning networks, training a model and the quality of its learning curve is guided by loss functions. These functions act as, and mostly are, evaluation metrics indicating how well the model's predictions align with the actual data. By quantifying the discrepancy between the predicted and true values, loss functions provide a clear objective for the model to optimise, working to minimise the loss functions by learning features and advancing their 'understanding' of important features and characteristics.

The choice of a loss function is vital as it can affect the whole learning process and is therefore influenced by the nature of the task and the data. Different loss functions have distinct ways of penalising errors, and their selection can significantly impact the model's learning trajectory and final performance. Below are 3 common loss functions used on regression tasks:

- Mean Squared Error (MSE): See Evaluation Metrics
- Mean Absolute Error (MAE): See Evaluation Metrics
- Huber Loss: A hybrid between MSE and MAE, Huber loss employs squared error for small discrepancies and absolute error for larger ones. This dual approach makes it less sensitive to outliers than MSE, while still emphasising significant errors.

2.13. Data Augmentation

Data augmentation (Figure 1) is a technique used to artificially expand the size of a training dataset by applying various transformations to the original images [23]. By introducing alterations like rotations, shifts, and flips, it can create a diverse set of training samples, enhancing the model's ability to generalise, and reducing the risk of overfitting.

3. Method

The primary objective of this study was to develop a robust convolutional neural network (CNN) that could accurately predict the composition of cell types in artificially generated microscopy images. The python libraries used in this project were TensorFlow [24] and Keras [25] for the deep learning framework, Imgaug [26] for the data augmentations (Figure 2), and sci-kit learn [27] for data splitting.

Dataset Generation: The dataset comprised of artificially generated 512x512 microscopy images, each containing three distinct cell types (Figure 1). A total of 5,000 such images were generated, with labels indicating the percentage composition of each cell type. These simulations were structured to generate basic approximations of phase-contrast imaging. However, they can also be modified to depict cell bodies with more intricate internal structures, providing a more detailed visualisation of cellular components and arrangements if needed. To optimise computational efficiency without compromising the integrity of the data, these images were down sampled to a resolution of 128x128.

The dataset was split into training, validation, and testing sets using a 70/30 ratio. Specifically, 3,500 images were allocated for training, while 750 images each were set aside for validation and testing.

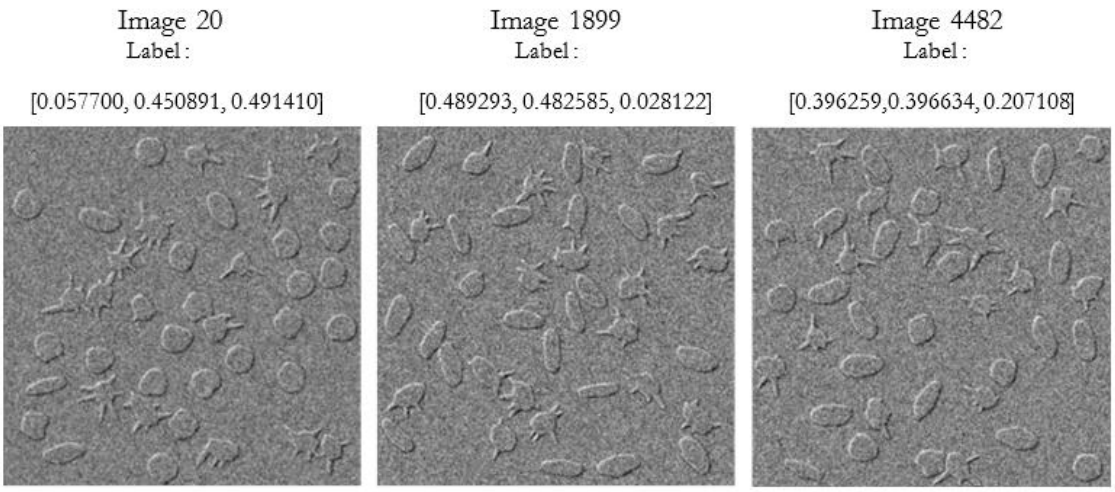

Figure 1. Artificially generated microscopy images (of 5000), composed of 3 cell types.

3.1. Data Augmentation

One of the draw backs of deep learning is its requirements for vast amounts of data, particularly when working with image data, and having a diverse and extensive dataset is crucial for training robust models [23]. However, collecting vast amounts of data is often challenging and resource intensive, therefore data augmentation is applied, with the primary goal of increasing the diversity of training samples without collecting new data.

Techniques Used

- **Rotations:** Images are rotated by ±30°, introducing variations in orientation.
- **Flips:** Images are flipped horizontally or vertically, 50% of the time, creating a mirrored version.
- **Zoom:** Images are zoomed in or out, ±20%, simulating the effect of different distances from the subject.
- **Brightness & Contrast Adjustments:** The brightness and contrast levels of images are varied, by ±20%, ensuring the model is robust to different lighting conditions.
- **Gaussian Blur:** A Gaussian blur is applied to the images, with a sigma value ranging from 0 to 0.5. This simulates the effect of out-of-focus images and ensures the model can handle slightly blurred inputs.
- **Padding:** To ensure that the central content of the image remains visible after rotations or zoom-outs, padding was added. The padding was filled with a constant value of the mean pixel value of the image, to ensure a seamless extension.

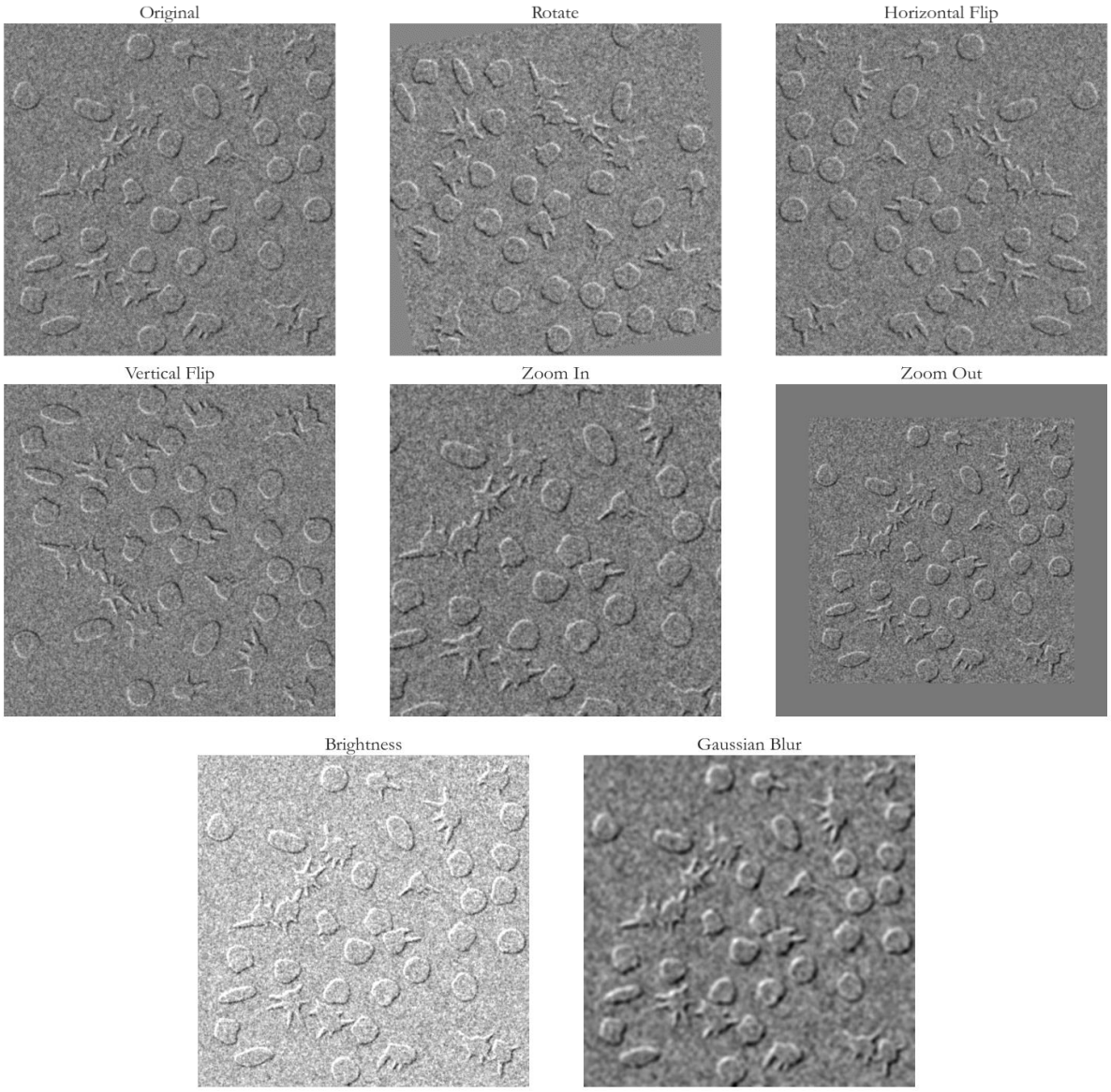

Figure 2. Original Image compared to the variety of data augmentation techniques applied to artificially increase the dataset and add real - world noise and variety to the image.

Data augmentation enhances model training by introducing variations in the dataset, combating overfitting, and promoting better generalisation. By leveraging existing data, it offers a cost-efficient alternative to collecting new samples and increasing the robustness of the model. Augmentations were applied exclusively to the training data; validation data was left in its original form.

3.2. Model Architecture and Training

The regression CNN, termed R-Net (Figure 3 and Table 2), was designed with multiple convolutional layers, each followed by batch normalisation and max-pooling layers. The model culminated in a series of dense layers.

Stacked convolutional layers were used to add depth to the model, with the potential to capture the full range of features from the cells, there are three blocks, each containing two convolutional layers. In the context of image analysis, especially in tasks like microscopy image analysis, the ability to capture hierarchical and spatial features is crucial. Double convolutions aid in capturing both local

details (like cell boundaries) and more global patterns (like tissue structures) [28], making them a valuable tool in such applications. Double convolutions can increase the expressiveness of the network without drastically increasing the number of parameters [28].

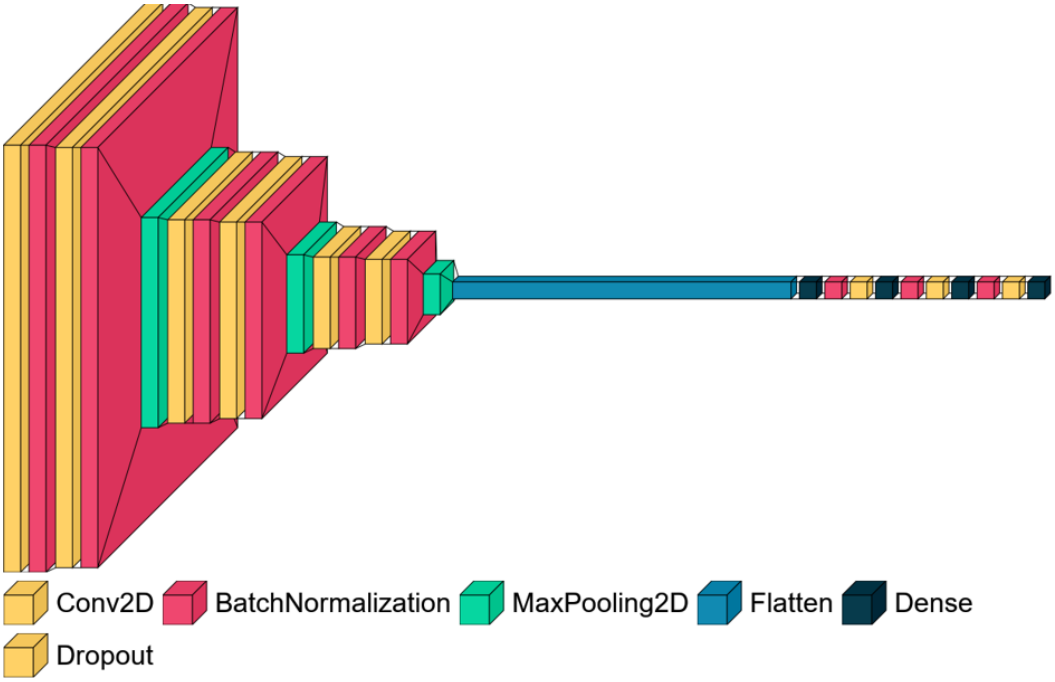

Figure 3. Construction of the R - Net model architecture

The Rectified Linear Unit (ReLU) [29] activation function was chosen for its ability to introduce non-linearity without the risk of vanishing gradients. Importantly, ReLU ensures that no negative values are produced, which is crucial given that the labels are percentages summing up to 1. While LeakyReLU was considered, ReLU was superior because of its simplicity, removal of negative numbers, and finally. being cheaper to compute and train with.

$$ReLU: f(x) = max(0, x)$$

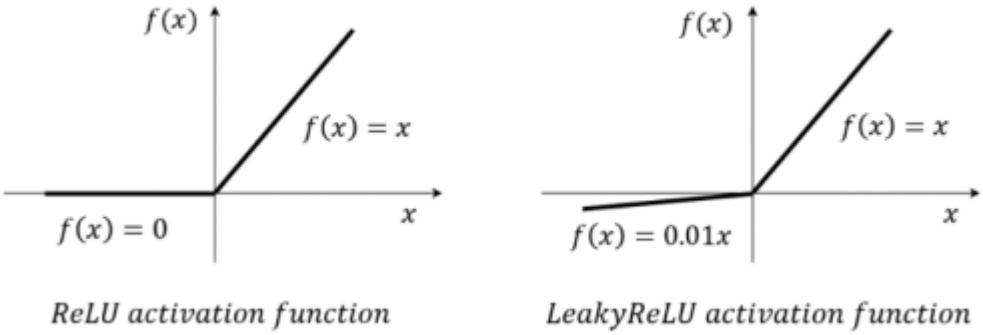

Figure 4. A comparison of the ReLU activation function vs the Leaky - ReLU activation function [30]

R - Net was trained for 200 epochs using the RMSprop optimiser. RMSprop was selected for its ability to achieve faster convergence and offer robust performance, especially when the data present has varied patterns and features, such as in microscopy images.

The loss function employed was the Mean Squared Error (MSE). MSE was chosen for its sensitivity to larger prediction errors. This is particularly vital in microscopy image analysis, where precision is paramount. Other evaluation metrics, including RMSE, MAE, MAPE, and R^2, were also monitored throughout the training process.

In the process of refining the model for optimal performance, a rigorous K-fold cross-validation technique was employed to determine the most suitable loss function. After evaluating various loss functions, Mean Squared Error (MSE) emerged as the most effective, providing the lowest validation error and the best generalisation to unseen data, compared to MAE and Huber Loss. Subsequently, when selecting the optimiser for the R - Net, Adam, RMSProp and SGD were assessed under the same conditions. Adam was considered for the model development in this project due to its adaptive learning rate and ability to handle noisy problems. However, after conducting the cross-validation experiment, RMSprop outperformed Adam. As detailed in Appendix I, the chosen optimisation technique for the model was RMSprop. The distinct advantage of using RMSprop was its capability to adjust learning rates for each parameter individually. This allowed the model to adapt more proficiently, bypassing issues such as vanishing or exploding gradients, which were a main challenge given the intricacy of the data involve.

3.3. Model Evaluation

Post-training, the model's performance was evaluated on the test set using various metrics. The Mean Squared Error (MSE) quantifies the average squared difference between predicted and actual values, emphasising larger discrepancies. The Mean Absolute Percentage Error (MAPE) offers a percentage-based perspective on prediction errors, while the Mean Absolute Error (MAE) provides a direct view on prediction errors. R-squared (R^2) highlights how well the model's predictions align with the actual results. The Root Mean Squared Error (RMSE) offers an interpretable metric directly comparable to the data's scale.

They provide a holistic representation of the model’s accuracy of predictions and understanding of the task at hand, quantifying its ability to generalise and explaining the variability in the data used to predict new test data. This ensures not only precise predictions but also a deeper understanding of the underlying patterns within the microscopy slide images.

4. Results

Table 3. Comparative performance metrics of various deep learning models on the microscopy image dataset with data augmentation

Models	Best Scores						
	MSE	**MAE**	**R2**	**MAPE**	**RMSE**	**Epochs**	**Time (hrs)**
VGG16	0.0116	0.0920	0.3215	N/A	N/A	20	~4
ResNet	0.0107	0.0899	0.4025	N/A	N/A	20	~4
MobileNet	0.0278	0.1017	0.0959	N/A	N/A	20	~3
MNIST Network	0.0316	0.1051	0.3921	112.6001	0.1778	200	~1.5
R - Net	0.00723	0.0684	0.7576	67. 2648	0.0849	200	~1.5

The performance of various deep learning models was assessed based on their ability to predict outcomes on the given dataset. The evaluation metrics used were Mean Squared Error (MSE), Mean Absolute Error (MAE), R-squared (R2), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE). The results for each model are as follows:

The evaluation of various deep learning models on microscopy images highlighted the superior performance of R-Net. VGG16, renowned for image classification, secured an MSE of 0.0116, an MAE of 0.0920, and an R2 score of 0.3215 after 4 hours of training. ResNet, leveraging its deep residual networks, slightly outperformed VGG16 with an MSE of 0.0107, an MAE of 0.0899, and an R2 of 0.4025 within a comparable 4-hour window. MobileNet, achieved an MSE of 0.0278, an MAE of 0.1017, and an R2 of 0.0959 in 3 hours. The MNIST Network (Appendix III), which was developed as a prototype version of R - Net, had an MSE of 0.0316, an MAE of 0.1051, and an R2 of 0.3921 over an extended 200 epochs, concluding in 1.5 hours. R-Net emerged as the top performer, registering the lowest MSE of 0.0072, an MAE of 0.0684, and an RMSE of 0.0849. Furthermore, its R2 score of 0.7576 explained 75.76% of the dataset's variability, all achieved within the same 1.5-hour span as the MNIST Network.

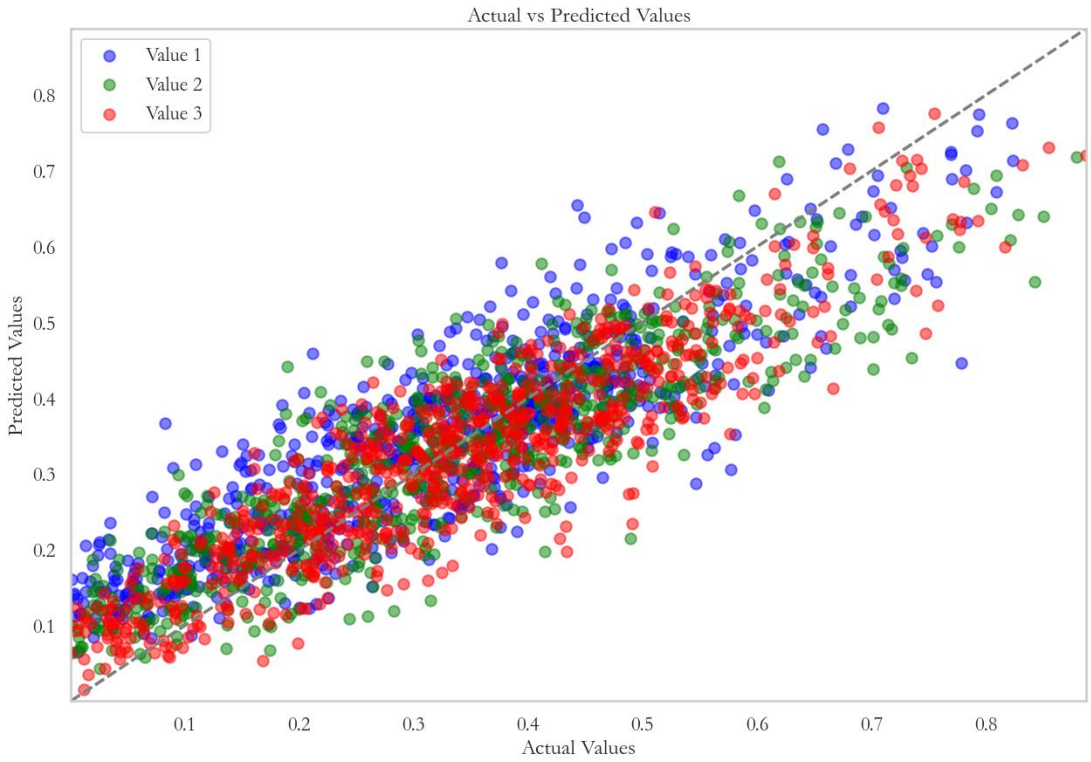

Figure 5. Scatter Plot of Actual vs. Predicted Values. Diagonal dotted line represents a perfect prediction.

The scatterplot, depicted in Figure 5, visualises the correlation between the actual values obtained from the dataset labels and the predicted values computed by R-Net. It serves to illustrate the effectiveness and accuracy of the model in predicting the artificially generated images.

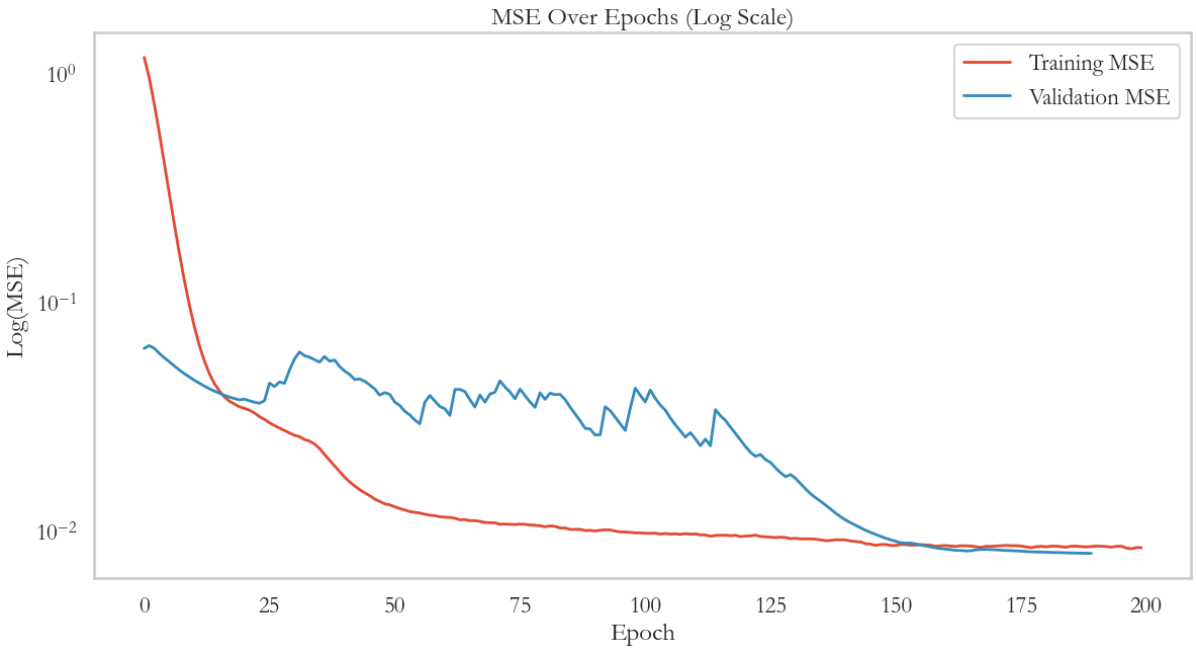

Figure 6. MSE progression over 200 epochs, both training (red) and validation (blue) trends are depicted.

The Mean Squared Error (MSE) graph displayed a final trend in its predictions on microscopy images. The training MSE, beginning from an initial high value of 1.1764, experienced a sharp decline

to 0.0085 by the end of the epochs. At the same time, the validation MSE started at 0.0631 and settled at a 0.0079.

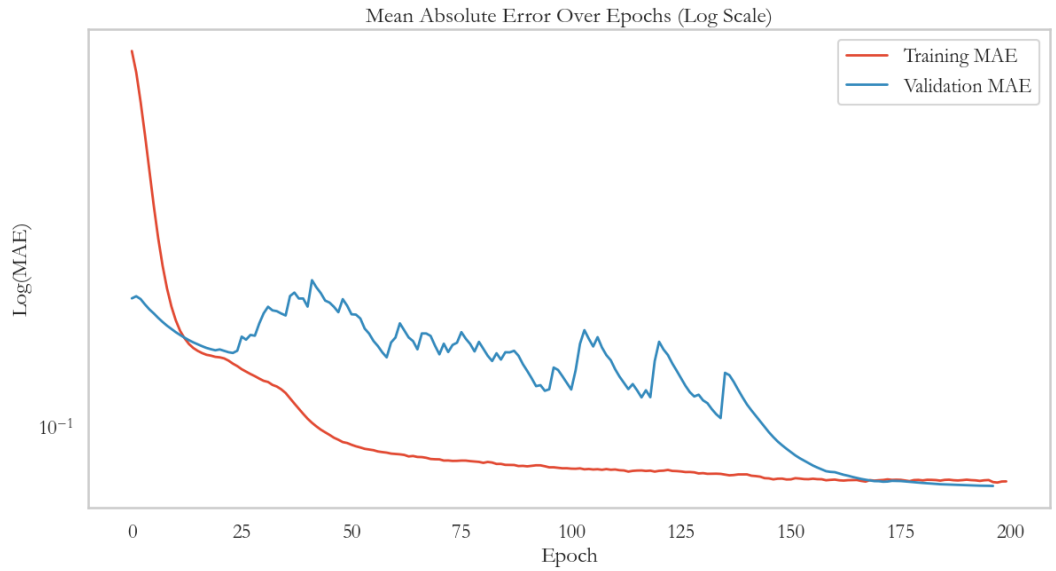

Figure 7. MAE progression over 200 epochs, both training (red) and validation (blue) trends are depicted.

The progression of the Mean Absolute Error (MAE) over 200 epochs also demonstrates the model's capability to predict with increasing accuracy. The training MAE started at a high 0.8321 and significantly improved to 0.0736 by the final epoch. Similarly, the validation MAE began at 0.2063 and saw a reduction to 0.0711.

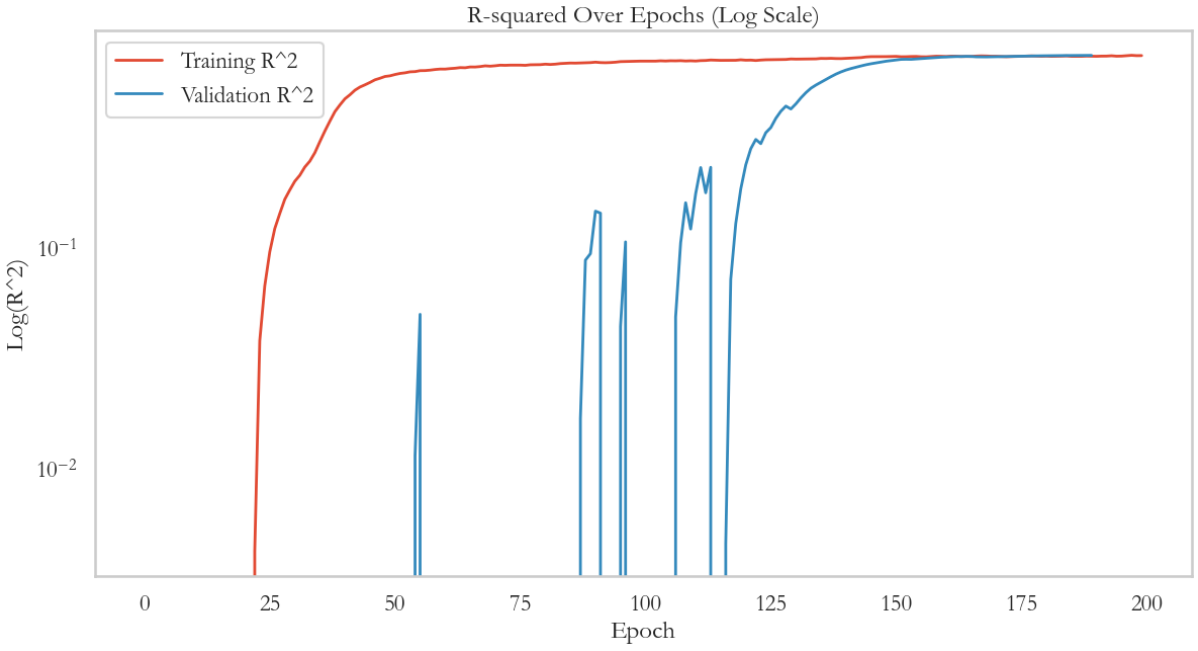

Figure 8. R^2 progression over 200 epochs, both training (red) and validation (blue) trends are depicted.

The R2 value, indicative of the model's explanatory power and ability to understand variance, exhibited an upward trajectory over the 200 epochs. The training R2 started from a negative value of -36.0784, suggesting a poor initial fit, but climbed to 0.7404 by the concluding epoch. Similarly, the validation initiated from -1.0166 and surged to a 0.7441.

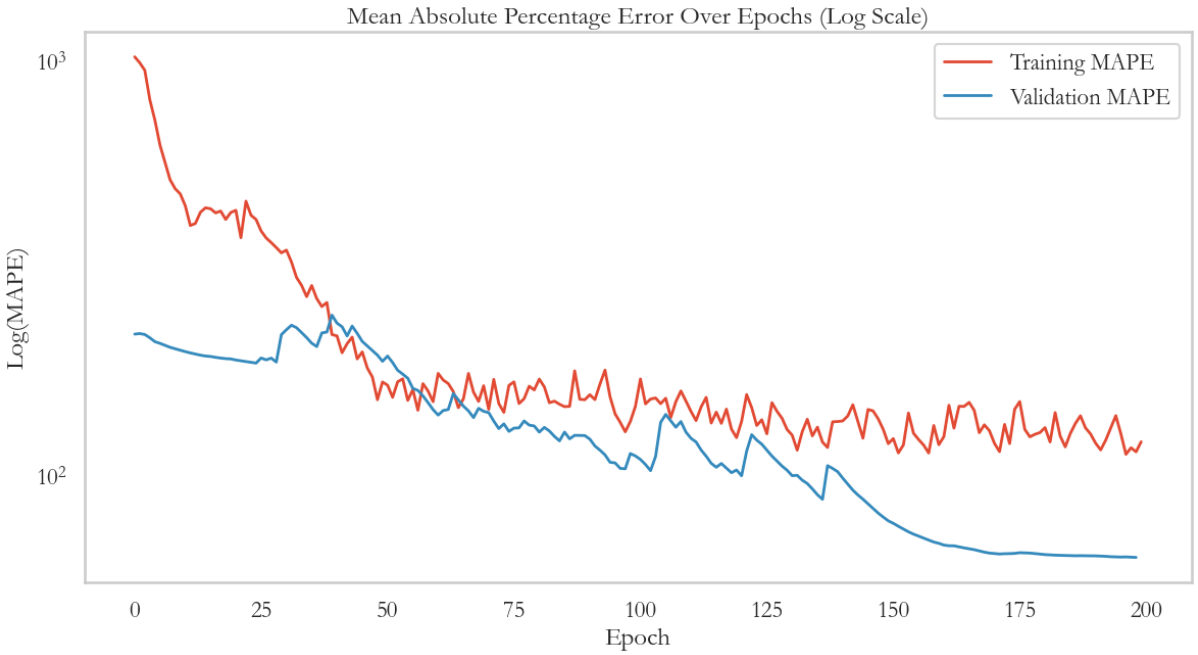

Figure 9. MAPE progression over 200 epochs, both training (red) and validation (blue) trends are depicted.

The trend of the Mean Absolute Percentage Error (MAPE) demonstrates the model's growing accuracy in predictions. The training MAPE, starting from 1024.41%, underwent a gradual reduction, culminating to a lower percentage error of 136.26% by the final epoch. Alongside it, the validation MAPE starting at 219.83 and exponentially decreased to a 62.83%.

5. Discussion

In this study, R – Net was developed and evaluated against VGG16, ResNet, MobileNet, and an MNIST Network, based on their predictive accuracy on microscopy images. Among these, R-Net exhibited superior performance (Table 3), achieving the lowest MSE, MAE, RMSE, and the highest R2 score, indicating its proficiency in accurately predicting percentage-wise cell composition within a training period of 1.5 hours. While the MAPE score might seem high, its score is susceptible to skewness, as it is sensitive to values between 0 - 1 [20] and therefore should not be the sole determinant in assessing robustness. This consistent improvement could be attributed to the model's ability to gradually refine its internal parameters and learn more representative features of the data as it processes more examples.

In analysing the plotted metrics (Figure 6 - 9), a critical observation is the convergence or divergence of the training and validation curves. The training MAE is consistently lower and diverges from the validation MAE, signifying overfitting, suggesting that the model is learning the noise in the training data. The trend of the $R2$ score is pivoting; a decreasing score for the validation compared to the training indicating under/overfitting. Similarly, a divergence between the Training and Validation MSE highlights overfitting. An increasing difference over time is foretelling of a deteriorating model generalisation, to improve the model further, overfitting must be mitigated.

Figure 5 illustrates the efficacy of R-Net in predicting outcomes, the majority of data points aligning closely with the line of perfect prediction, denotes high model accuracy and an indication that the model's predictions are generally in agreement with the actual values. The R2 score is high, indicating a good fit of the model to the data. The MAE and RMSE for all three values are low too, indicating that the model's predictions are generally close to the actual values. However, the points that deviate from the line of perfect prediction suggest areas of potential improvement, specifically the actual values closer upper and lower bounds, becoming worse at generalising.

The results obtained suggest that R-Net may have an advantage in managing the complexities inherent in microscopy images. The selection of the architecture is hierarchical, as highlighted by Louati et al. [31], and seems to have worked, allowing the model to discern both broad and fine-grained features effectively. The stacking of convolutional layers added depth, allowing the network to focus on intricate details, contributing to the model's accuracy, hierarchical feature learning, and regularisation. This can be inferred as they were only implemented into R – Net but not the MNIST originally assessed. It is what is used in successful models such as VGG16 [17], it allowed the network to capture and represent more complex features, as the spatial resolution of the feature maps is preserved longer in the network, ensuring the retention of finer-grained spatial information. Additionally, by stacking two 3x3 convolutional layers, it optimises the network's performance by having the same effect as a 5x5 convolution with 25 parameters but only using a 3x3's 18 parameters. The increased non-linearity introduced by more convolutional layers, coupled with activation functions like ReLU, empowers the model to navigate complex and non-linear relationships in the data effectively, enabling the model to construct a multi-layered understanding of the task at hand.

Using pre-trained networks in microscopy image analysis is advantageous due to the extensive features they have already discerned from previous learning experiences. Such networks can expedite the training process and, often, yield enhanced performance compared to models developed from scratch. However, this may not be true for all image data, especially microscopy images. Models trained from scratch on microscopy data tend to exhibit similar or superior performance [32]. While imaging techniques such as MRI benefit from pre – trained networks [33], microscopy images are extremely diverse and may suffer from a pre – trained network and its weights. When dealing with microstructures and specialised tasks, it might be necessary to develop specialised model architectures to handle such unique requirements effectively [32]. This was certainly the case in Table 3, as while the pretrained networks only ran for 20 epochs, R – Net has a lower time complexity, aiding the efficiency this project set out to seek.

5.1. Challenges, limitations, and improvements

Overfitting was one of the main challenges faced, where the model intricately learns the training data, interpreting noise as actual patterns, and compromising its generalisation. In this project, strategies like dropout and data augmentation were employed to mitigate overfitting. Dropout serves as a regularisation technique, selects a random set of neurons to deactivate during training [34]. This random deactivation ensures that the network doesn't become reliant on any specific neuron or set of neurons. As a result, the model is more likely to generalise well to unseen data, rather than just memorising the training set. Another regularisation used was data augmentation, it artificially alters the training dataset by introducing modifications to the input data [34], offering the model a diversified range of examples and increasing its adaptability and robustness to new and varied data.

Further regularisation could involve the implementation of L1 and L2 parameters [34], enhancing resilience. Together, these methodologies fortify the model's robustness and predictive accuracy, helping to mitigate overfitting.

The model developed operates end-to-end, executing a regression pipeline which provides the significant advantage of holistic learning, encapsulating the entire learning process in a single stage. This approach is advantageous due to its streamlined nature. This addition of cell segmentation induces an inherent layer of complexity. It transforms the model's learning process into a two-stage paradigm where inaccuracies or errors in the segmentation stage can propagate through the next stage, impacting the overall model's accuracy and reliability. However, there is also benefit to cell segmentation, it aids in enhancing the model's interpretability and allows for an analysis of cell features, providing insights into what the model is learning, identifying, or overlooking.

In areas where precision is crucial, such as in hospitals, a redundancy approach is often utilised, combining both classification and regression models to aid the systems robustness and as a backup mechanism to ensure reliability and availability. This methodology is reflective of the strategies employed in the Apollo space missions, which also used multiple independent systems [35]. This fail - safe is useful in critical environments, where the margin for error is minimal, and the need for reliability and fault tolerance is high.

5.2. Future work

R - Net can be refined further by exploring different architectures, such as skip connections introduced by ResNet [18], or fine-tuning hyperparameters using Cross-validation or more aggressive data augmentations that could allow the discovery of hidden patterns in the data. This approach can produce a model with greater accuracy with improved generalisation, enhanced robustness to outliers and noise, and better performance in predicting extreme values, thereby ensuring its reliability and efficacy in varied and dynamic real-world scenarios.

The use of procedurally generated images mitigates the impact of human error but also poses limitations as it may not depict real-life conditions accurately, while it can be minimised with the use of data augmentation, this can still affect the generalisation and robustness of R – Net.

6. References

[1] I. G. Goldberg *et al.*, “The Open Microscopy Environment (OME) Data Model and XML file: open tools for informatics and quantitative analysis in biological imaging.,” *Genome Biol*, vol. 6, no. 5, 2005.

[2] J. De Fauw *et al.*, “Clinically applicable deep learning for diagnosis and referral in retinal disease,” *Nat Med*, vol. 24, no. 9, pp. 1342–1350, Sep. 2018, doi: 10.1038/s41591-018-0107-6.

[3] “Microscope Contrast Techniques.” 2020. [Online]. Available: https://www.microscopeworld.com/p-4440-microscope-contrast-techniques.aspx

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[5] K. Fukushima, “Biological Cybernetics Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position,” 1980.

[6] Y. Liu *et al.*, “Artificial Intelligence–Based Breast Cancer Nodal Metastasis Detection: Insights Into the Black Box for Pathologists,” *Arch Pathol Lab Med*, vol. 143, no. 7, pp. 859–868, Jul. 2019, doi: 10.5858/arpa.2018-0147-OA.

[7] P. Rajpurkar *et al.*, “Deep learning for chest radiograph diagnosis: A retrospective comparison of the CheXNeXt algorithm to practicing radiologists,” *PLoS Med*, vol. 15, no. 11, Nov. 2018, doi: 10.1371/journal.pmed.1002686.

[8] A. Esteva *et al.*, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115–118, Feb. 2017, doi: 10.1038/nature21056.

[9] V. Gulshan *et al.*, “Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs,” *JAMA - Journal of the American Medical Association*, vol. 316, no. 22, pp. 2402–2410, Dec. 2016, doi: 10.1001/jama.2016.17216.

[10] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” 2009.

[11] S. Yang, B. Fang, W. Tang, X. Wu, J. Qian, and W. Yang, “Faster R-CNN based microscopic cell detection,” in *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, 2017, pp. 345–350. doi: 10.1109/SPAC.2017.8304302.

[12] F. Xing, Y. Xie, H. Su, F. Liu, and L. Yang, “Deep Learning in Microscopy Image Analysis: A Survey,” *IEEE Trans Neural Netw Learn Syst*, vol. 29, no. 10, pp. 4550–4568, Oct. 2018, doi: 10.1109/TNNLS.2017.2766168.

[13] J. Zhang, C. Li, Y. Yin, J. Zhang, and M. Grzegorzek, “Applications of artificial neural networks in microorganism image analysis: a comprehensive review from conventional multilayer perceptron to popular convolutional neural network and potential visual transformer,” *Artif Intell Rev*, vol. 56, no. 2, pp. 1013–1070, Feb. 2023, doi: 10.1007/s10462-022-10192-7.

[14] K. Koutini, H. Eghbal-zadeh, and G. Widmer, “Receptive Field Regularization Techniques for Audio Classification and Tagging with Deep Convolutional Neural Networks,” May 2021, doi: 10.1109/TASLP.2021.3082307.

[15] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” Feb. 2015, [Online]. Available: http://arxiv.org/abs/1502.03167

[16] F. Chollet, “Keras.” ONEIROS, 2023. [Online]. Available: https://keras.io/

[17] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Sep. 2014.

[18] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.03385

[19] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 2017, [Online]. Available: http://arxiv.org/abs/1704.04861

[20] S. Kim and H. Kim, “A new metric of absolute percentage error for intermittent demand forecasts,” *Int J Forecast*, vol. 32, no. 3, pp. 669–679, Jul. 2016, doi: 10.1016/j.ijforecast.2015.12.003.

[21] C. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Clim Res*, vol. 30, pp. 79–82, 2005, doi: 10.3354/cr030079.

[22] A. Colin Cameron and F. A. G. Windmeijer, “An R-squared measure of goodness of fit for some common nonlinear regression models,” *J Econom*, vol. 77, no. 2, pp. 329–342, Apr. 1997, doi: 10.1016/S0304-4076(96)01818-0.

[23] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J Big Data*, vol. 6, no. 1, p. 60, Dec. 2019, doi: 10.1186/s40537-019-0197-0.

[24] Martín~Abadi *et al.*, “ TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.” 2015. [Online]. Available: https://www.tensorflow.org/

[25] F. Chollet and others, “Keras.” 2015.

[26] A. B. Jung, “imgaug.” 2018.

[27] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in {P}ython,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[28] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.1556

[29] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.08375

[30] Z. Li, W. T. Nash, S. P. O’brien, Y. Qiu, R. K. Gupta, and N. Birbilis, “cardiGAN: A Generative Adversarial Network Model for Design and Discovery of Multi Principal Element Alloys.” [Online]. Available: https://github.com/anucecszl/cardiGAN

[31] H. Louati, S. Bechikh, A. Louati, A. Aldaej, and L. Ben Said, “Joint design and compression of convolutional neural networks as a Bi-level optimization problem,” *Neural Comput Appl*, vol. 34, no. 17, pp. 15007–15029, Sep. 2022, doi: 10.1007/s00521-022-07331-0.

[32] J. Stuckner, B. Harder, and T. M. Smith, “Microstructure segmentation with deep learning encoders pre-trained on a large microscopy dataset,” *NPJ Comput Mater*, vol. 8, no. 1, Dec. 2022, doi: 10.1038/s41524-022-00878-5.

[33] S. Krishnapriya and Y. Karuna, “Pre-trained deep learning models for brain MRI image classification,” *Front Hum Neurosci*, vol. 17, 2023, doi: 10.3389/fnhum.2023.1150120.

[34] X. Ying, “An Overview of Overfitting and its Solutions,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Mar. 2019. doi: 10.1088/1742-6596/1168/2/022022.

[35] S. R. Ryan and M. E. Granger, “The importance of dissimilar redundancy for safety in future space vehicle design,” *Journal of Space Safety Engineering*, 2023, doi: 10.1016/j.jsse.2023.08.005.

7. Appendix

7.1. Appendix I

```
def create_model():

    """
    This function creates and returns a Sequential model.
    The model is structured as a series of Convolutional layers, followed
by BatchNormalization, MaxPooling,
    and Dropout layers to prevent overfitting. It finally has Dense layers
at the end to perform regression.
    """

    model = Sequential([
    #Input Layer
        layers.Conv2D(16, kernel_size=(3, 3), activation='relu',
input_shape=(128, 128, 1)),
        layers.BatchNormalization(),
        layers.Conv2D(16, kernel_size=(3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),

        #Second Stacked Block
```

```
        layers.Conv2D(32, kernel_size=(3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.Conv2D(32, kernel_size=(3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),

        #Third Stacked Block
        layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.Conv2D(64, kernel_size=(3, 3), activation='relu'),
        layers.BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),

        #Flatten layer to flatten the features for Dense layers
        layers.Flatten(),

        #Dense layers
        layers.Dense(128, activation='relu'),
        layers.BatchNormalization(),
        #Dropouts
        layers.Dropout(0.3),
        layers.Dense(64, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.3),
        layers.Dense(32, activation='relu'),
        layers.BatchNormalization(),
        layers.Dropout(0.3),

        #Output layer
        layers.Dense(3)
    ])

def load_and_process_data():

 """
    This function loads and processes the data from the specified folder
path.
    It loads the .npy files starting with the specified prefix and
combines them into a single DataFrame.
    It also loads and processes the images from the specified image path.
    """

# Load and combine data
combined_df = pd.DataFrame(columns=['Cell ID', 'Value 1', 'Value 2',
'Value 3'])
for filename in os.listdir(FOLDER_PATH):
    if filename.endswith('.npy') and filename.startswith(PRIORS_PREFIX):
        cell_id = filename[len(PRIORS_PREFIX):][:-4]
        npy_data = np.load(os.path.join(FOLDER_PATH, filename))
        if npy_data.ndim == 1:
            npy_data = npy_data.reshape((1, -1))
        df = pd.DataFrame(npy_data, columns=['Value 1', 'Value 2', 'Value
3'])
        df['Cell ID'] = cell_id
        combined_df = pd.concat([combined_df, df[['Cell ID', 'Value 1',
'Value 2', 'Value 3']]], ignore_index=True)
```

```
#Load, process, and flatten the images
flatten_list = [preprocess_img(os.path.join(IMG_PATH,
f"img{str(i).zfill(4)}.png")) for i in range(5000)]
flatten_arr = np.array(flatten_list)

return combined_df, flatten_arr


def preprocess_img(img_path):

    """
    This function reads an image from the specified path, resizes it to
128x128 pixels,
    normalizes it, and adds an additional dimension to it.
    """

    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
    img = cv2.resize(img, (128, 128))
    img = img.astype('float32') / 255.0
    img = np.expand_dims(img, axis=-1)
    return img

def root_mean_squared_error(y_true, y_pred):

"""
This function calculates the Root Mean Squared Error between the true and
predicted values.
"""

return K.sqrt(K.mean(K.square(y_pred - y_true)))


def r_squared(y_true, y_pred):

    """
    This function calculates the R squared value between the true and
predicted values.
    """

    SS_res = K.sum(K.square(y_true - y_pred))
    SS_tot = K.sum(K.square(y_true - K.mean(y_true)))
    return (1 - SS_res / (SS_tot + K.epsilon()))


def imgaug_data_generator(X, y, batch_size, augmentations):

    """
    This function is a generator that yields batches of images and labels
indefinitely.
    It applies the specified augmentations to the images in each batch.
    """

    while True:
        idx = np.random.permutation(X.shape[0])
        num_batches = int(np.ceil(X.shape[0] / batch_size))
```

```
        for i in range(num_batches):
            batch_idx = idx[i*batch_size:(i+1)*batch_size]
            X_batch = X[batch_idx]
            y_batch = y[batch_idx]

            # Perform augmentations
            X_batch_aug = augmentations(images=X_batch)
            yield X_batch_aug, y_batch


#Define the augmentations to be applied to the images
augmentations = iaa.Sequential([
    iaa.Affine(rotate=(-30, 30), mode="constant", cval=avg_pixel),
    iaa.Fliplr(0.5),
    iaa.Flipud(0.5),
    iaa.Affine(scale={"x": (0.9, 1.1), "y": (0.9, 1.1)}, mode="constant",
cval=avg_pixel),
    iaa.Multiply((0.9, 1.1)),
    iaa.GaussianBlur(sigma=(0, 0.5)),
])

#Compile the model
model.compile(optimizer=RMSprop(), loss='mean_squared_error',
          metrics=['mse', root_mean_squared_error, 'mae', 'mape',
r_squared])

#Define callbacks
checkpoint = ModelCheckpoint(MODEL_PATH, monitor='val_loss', verbose=30,
save_best_only=True, mode='min')
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5,
min_lr=0.001)

# Train the model using the data generator and applying the defined
augmentations
history = model.fit(
    imgaug_data_generator(X_train, y_train, batch_size=32,
augmentations=augmentations),
    validation_data=(X_val, y_val),
    steps_per_epoch=int(np.ceil(X_train.shape[0] / 32)),
    epochs=200,
    callbacks=[checkpoint, reduce_lr]
)
```

7.2. Appendix II

Table 4. Full structure, layers, output sized and block structures of the R - Net

R - Net		
Layer Name	**Output Size**	**Block Structure**
Conv1	128 x 128	3 x 3, 16
Batch Norm	128 x 128	-
Conv2	128 x 128	3 x 3, 16
Batch Norm	128 x 128	-
MaxPool	64 x 64	2 x 2
Conv3	64 x 64	3 x 3, 32
Batch Norm	64 x 64	-
Conv4	64 x 64	3 x 3, 32
Batch Norm	64 x 64	-
MaxPool	32 x 32	2 x 2
Conv5	32 x 32	3 x 3, 64
Batch Norm	32 x 32	-
Conv6	32 x 32	3 x 3, 64
Batch Norm	32 x 32	-
MaxPool	16 x 16	2 x 2
Dense1	128	-
Batch Norm	128	-
Dropout1	128	-
Dense2	64	-
Batch Norm	64	-
Dropout2	64	-
Dense3	32	-
Batch Norm	32	-
Dropout3	32	-
Output Layer	3	-

7.3. Appendix III

Table 5. MNIST Network used and trained before improving design towards the R - Net

Standard MNIST Convolution Network		
Layer Name	Output Size	Block Structure
Conv1	128 x 128	3 x 3, 32
MaxPool1	64 x 64	2 x 2
Conv2	64 x 64	3 x 3, 64
MaxPool2	32 x 32	2 x 2
Conv3	32 x 32	3 x 3, 128
MaxPool3	16 x 16	2 x 2
Flatten	-	-
Dense1	256	-
Dense2	128	-
Dense3	64	-
Dense4	32	-
Output Layer	3	-

7.4. Appendix IV

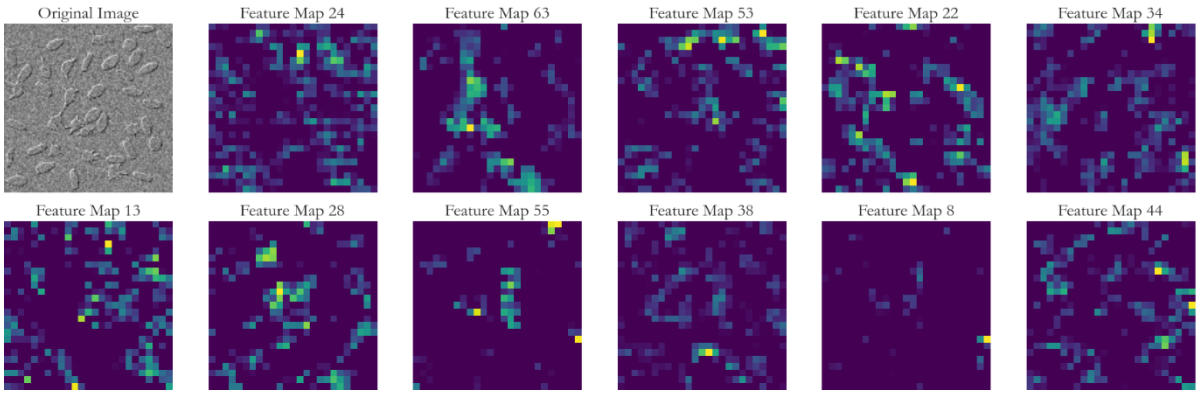

Figure 10. Display of the last convolution layers feature maps displaying points of peak attention focused on by R - Net (yellow - high, Purple - Low)