U-Net CNN: Myotube Detection in vitro

Jared Andrés Silva Villa, Paola Félix Torres, Herbert Eduardo Euroza Hernández, and Alma Paulina González Sandoval

Instituto Tecnológico y de Estudios Superiores de Monterrey
Guadalajara Jal., México

Keywords: Myotube · Image segmentation · Convolutional Neural Networks (CNN) · U-Net model · Image pre-processing · Data augmentation · Neural network training · Supervised learning · Deep learning · Object detection

Abstract. The task of image segmentation has been rightfully dominated by convolutional neural networks (CNN) with diverse architectures. Diverse fields that deal with the biomedical are such fields that need this kind of tool. Different CNN architectures have been put in place but one stands out in any biomedical research, that being the U-Net architecture. This paper explores the use of such models and architectures for the purposes of creating a tool to aid in myotube research, more specifically how myotube formation is affected by different alcohol concentrations in skeletal muscle samples taken from female volunteers.

1 Introduction

The development of artificial intelligence models for image segmentation, computer vision or object detection has shown great promise in various fields. This report focuses on a specific challenge; the segmentation of myotubes in cellular images.

Myotubes are an essential intermediate structure in muscle development, formed during the differentiation of muscle cells. They are created when precursor cells, known as myoblasts, align and fuse in a process called myotubeogenesis. This fusion results in elongated, multinucleated cells that serve as precursors of mature muscle fibers. These mature fibers are ultimately responsible for muscle contraction and movement in the body.

Segmenting myotubes in cellular images is an important task in biology research. High quality segmentation can facilitate the quantitative analysis of myotube formation, morphology and density, providing valuable insights into muscle related diseases and potential risks. In this case, high-quality segmentation is critical as it ensures that the scientist leading the research will not encounter confusion or bias during segmentation, as might occur with manual segmentation.

The objective of this project is to develop and evaluate AI models capable of accurately identifying and segmenting myotubes in cellular images. These models aim to assist researches and automating a traditionally manual and time

consuming process, improving the efficiency and accuracy of their analyses. By leveraging advanced techniques in image processing and AI, this work seeks to contribute to the growing intersection of computational tools and biological research.

2 Methodology

For the purposes of this project, the following methodology was put in place in order to achieve the set objectives; these being to generate and train a machine learning model that can confidently detect myotubes for the purpose of facilitating object detection and removing observer bias during research. This way the investigation regarding how different levels of alcohol affects myotube formation in skeletal muscle can be done swiftly.

Firstly, the photos and data provided were examined at face value to give an initial approach to the problem at hand. After looking at the images and discussing different viewpoints on what should be considered a myotube, the process of image segmentation was realized. The team used a software to segment images by hand, providing a JSON file with all the annotations pertaining to supposed myotubes, all images segmented were from myotubes not exposed to any alcohol concentration. With the help of our collaborators, meetings were scheduled bi-weekly after a month had passed to discuss progress updates and ask relevant questions that would help resolve the doubts about the images provided, and therefore, improve the development of the model for object detection.

The development of the model is simple in concept. The images are segmented by hand, those images and their annotations are loaded into the model, it is trained on the images we chose and the model gives a prediction on further images to confirm its effectiveness. The problem in these kinds of situations resides in the amount of data. Considering the nature of the problem at hand, and how the images were provided, the team was able to gather a small sample of images that were deemed as viable for the model to learn on. To this capacity, the method of data augmentation was used to alleviate this issue. Data augmentation refers to generating more data from the existing data by different means, it consists of making adjustments to the images (as in rotation, flipping, changing brightness levels, adjusting contrast or saturation) to make more copies out of a single image. Within this project only rotation of the images was used. Having this in mind, having to segment the rest of the images that have been changed would be short of a hassle, in that regard a script was developed to rotate the annotation files making this process more streamlined and efficient, effectively reducing the workload needed to have a dataset of decent size. [5]

Why is this a problem anyway? Shouldn't just a few images be enough for the model to be effective. The answer to that question is yes, with a caveat. Machine learning models, not limited to neural networks and its variants, will have very promising results even on smaller datasets. The issue here is the lack of generalization of the model. Models trained on small datasets often display great metrics when validating existing data but struggle to make anything of new

data. This is the problem of over-fitting, and that's the reason behind trying to scale the dataset as much as possible. A model not capable of generalization is not an effective model, since the development of such models is made with the idea of making predictions on new data, not existing data. [5] With this in mind, we established a dataset of 600 images, where 80% was used in training and the remaining 20% was used in validation. The validation portion of the dataset offers insight in how the model would perform when new data is entered. After training is done and results are given, these are shown to our collaborators in the progress updates to ensure the model is on the right track.

The word "model" has been thrown around a lot in the previous sections and the question follows: Which model was used? The answer to that would be a neural network, more specifically a convolutional neural network or CNN for short. CNN's are used primarily in the field of image classification or object detection due to their effectiveness in the field. This model is inspired on how the animals process images. This attempt at recreating how biological neurons process external information is the basis of the CNN model. As the acronym and name imply, a CNN is made of convolutions which extract learnable weights and biases within a package of different components such as the convolution layers, the activation functions, pooling layers, and fully connected layers. [4]

Convolution layers essentially serve as the feature extractors, such as edges, texture, gradients and color if the image is not gray-scaled. These layers are made of filters called kernels which operate in a size of $n \times m \times d$. Then a dot product is computed across the kernel and the input several times depending on the architecture of the network, allowing the model to learn kernels that get activated by the features mentioned previously, like shown in Figure 1.

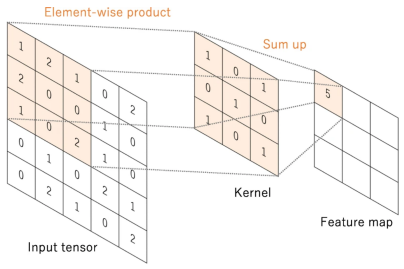

Fig. 1. A convolution layer of 3x3 computates a dot product to get the corresponding feature map

This, in short, is the process of convolution. [4] Training a CNN model in this regard is to identify the kernels that work best for the task at hand. Kernels are the only parameters that are self taught by the model itself while other parameters such as the size and number of kernels are called hyperparameters since they are defined before training forming the architecture of the network. [5] Once an output is formed from the process of convolution, the final step in

this type of layer is to run said output through an activation function. This is to more accurately depict the type of data we encounter in the real world, that being non-linear. The following are the most common within the field:

2.1 Sigmoid

Taking a real number x and depicting a range of values between 0 and 1. Large negative values are placed closer to 0 and large positive values closer to 1.

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

2.2 Tan Hyperbolic

Similar to the sigmoid function, it depicts an x value across a range of values between -1 and 1.

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2}$$

2.3 Rectified Linear Unit (ReLU for short)

Takes a real number x and converts it into 0 if x is negative.

$$f(x) = \max(0, x) \tag{3}$$

The first two functions had more widespread use since they are the mathematical representations of how neurons work. [5] However the last function is now the standard for these types of networks since it takes less computational run time and is significantly faster than the first two, also offering similar results. [4]

Pooling layers in essence reduces the dimensionality of a given feature map to decrease the number of learnable parameters in the models and serves as a way to introduce variance into the model, generalizing it. [2] Pooling results in having the feature maps in different pieces that then get replaced by a specific value according to the pooling used, either max pooling or average pooling. [4] Max pooling refers to the extraction of the maximum value within a patch. The downsampling is done according to set parameters and a pooling filter of 2x2 is the most commonly used as shown in the Figure 2. Average pooling is done by extracting the average of all elements within it, effectively transforming a n x m feature map into a 1x1. This is only applied before the fully connected layers in practice. [5]

Fully connected layer works as traditional artificial neural networks in the sense that each neuron has connections into it from the inputs gathered and these all have different weights associated with them. The output is then calculated by the sum of all weights with their corresponding values, and is followed by an activation function, in this case ReLU. [4] In the CNN, the final convolution or

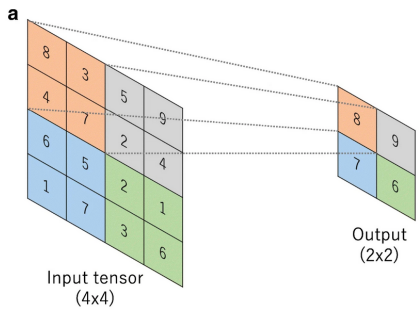

Fig. 2. Max pooling uses patches, in this example a 2x2, and gets the maximum value of the patch and passes it to a 2x2 where all the maximum inputs are stored

pooling is flattened, where the final output is transformed into a one dimensional vector of numbers that are use as the values previously discussed. [5]

These are the principles behind CNN models, which were the ones used during the development of this project. More specifically, a U-Net CNN model was used to complete the set objectives. U-Net refers to the architecture of the CNN model itself having a particular order of different layers like the ones discussed above. The U-net architecture proposed a model by which to distinguish every pixel, where the input is encoded and decoded to produce a segmentation mask with the same resolution to that of the original image. [2] This architecture consists of two paths: contraction and expansion. In the contraction phase features of the images are extracted and the feature maps are limited in size. The expansion phase performs up conversion to enable the recuperation of the full feature map. [1] In terms of layers, the contracting phase consists of repeating a two 3x3 convolution, both activated with a ReLU and a 2x2 max pooling filter with a stride of 2 for downsampling. The expansion phase consists of an upsampling of a feature map thrown into a 2x2 convolution, a concatenation of the feature map of limited size, and two 3x3 convolutions each activated by a ReLu once more and finally a ending layer of a 1x1 convolution, with a total of 23 layers; as seen in Figure 3. [3] In essence, the architecture dowscales the dimensions of the image and the feature map and then upscales it to form a segmentation mask of the same resolutions. This way the segmented mask can be overlapped with the original image for further validation.

These are the tools used for the development of the project. In short, the team segmented a sample of images by hand, having an image file with the images themselves and a JSON with all the annotations of myotubes encountered. These images were later flipped and rotated to artificially create a bigger data set. These images and their corresponding annotations were fed to the CNN model with the U-Net architecture. The final step in the process was to add a way to produce a sort of time-lapse with the images provided. The images were divided into several folders, each with a specific date entry. Each folder corresponded to a 3 hour interval of the images taken on the biological samples. The team sought

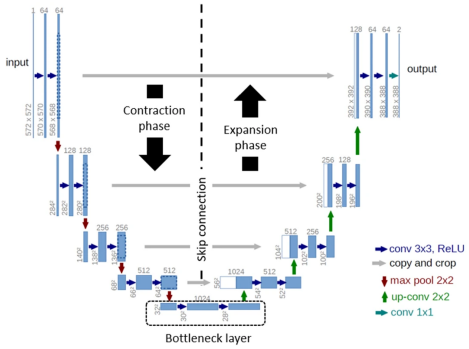

Fig. 3. Diagram of the U-net Architecture, as the name suggest it is formed in a U shaped where each stage is represented by each half. Conctracting phase at the left and expansion phase at the right

out to produce a script which could show the development of myotubes within a given plate using the predicted mask of the already trained model.

3 Experiments

In this section, we describe the experiments conducted to develop and optimize the Convolutional Neural Network (CNN) model for segmenting myotube structures. Our approach involved several stages, starting with image pre-processing to prepare the data for model input, followed by the application of data augmentation techniques to enrich the dataset. We then proceeded with the training of the CNN using the U-Net architecture, exploring various configurations and strategies to optimize its performance.

In our initial experiments with filters, we explored various pre-processing techniques to analyze and enhance the features of our images. These included binary thresholding, where pixels were classified as 0 or 255 based on intensity (see Figure 4); erosion, which shrinks object boundaries to remove small noise (see Figure 5); dilation, which expands object boundaries to fill gaps; and morphological opening, a combination of erosion and dilation to smooth regions and remove noise. Although these techniques proved effective in refining specific features, we ultimately opted for grayscale because of its balance between preserving detail and simplicity in processing.

To ensure that the images were suitable for processing with our Convolutional Neural Network (CNN), we resized the original images from their large resolution of 2592x1944 pixels to 256x256 pixels. This reduction in size was necessary to reduce computational overhead and ensure that the network could process the images more efficiently. Simultaneously, the associated JSON files containing the annotations for the images were updated to reflect the new dimensions, ensuring that all regions of interest were correctly aligned with the resized images.

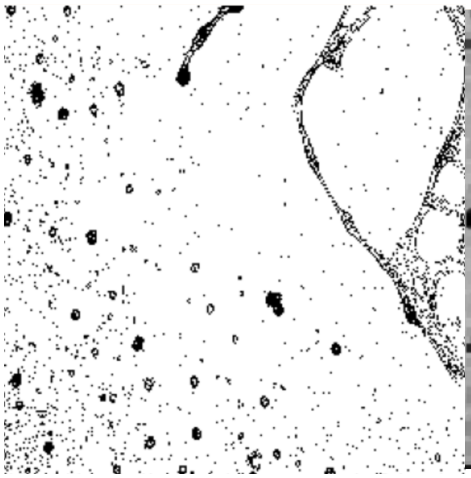

Fig. 4. Binary Thresholding

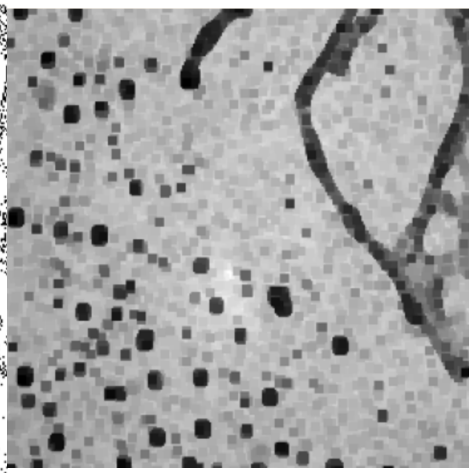

Fig. 5. Erosion

Upon resizing the images to the target dimensions, we applied data augmentation to further enrich the dataset. This process involved four distinct transformations: horizontal flipping and rotations of 90°, 180°, and 270°. Consequently, the original data set of 117 labeled images was expanded to a total of 585 images. With the augmented and resized dataset ready, we proceeded to verify the correctness of the transformations by using binary masks. These masks allowed us to confirm that both the images and their corresponding JSON annotations had been accurately transformed. In the following, we present an example showing a rotated and resized image along with its corresponding binary mask, which was generated from the JSON annotations. This visual comparison ensures the integrity of the transformation process (see Figure 6).

Regarding the training of the CNN, we focused on optimizing different components of the U-Net model to achieve the best performance in segmenting myotube structures from medical images. We experimented with several key elements, such as the choice of optimizer, the loss function, activation functions, and the number of epochs used during training.

For the optimizer, we selected Adam, known for its adaptive learning rate and efficiency in training deep learning models. We chose binary cross-entropy as the loss function, which is appropriate for binary segmentation tasks, where the goal is to distinguish between the presence or absence of a specific structure in the images. The model used ReLU activations in the convolutional layers to introduce non-linearity and enable the network to learn more complex patterns. For the output layer, we employed a sigmoid activation to produce a binary mask, predicting the presence or absence of myotubes.

We also experimented with different batch sizes and epochs, carefully monitoring the training process by evaluating the model's performance through accuracy and loss metrics at each epoch. We tested various epoch configurations,

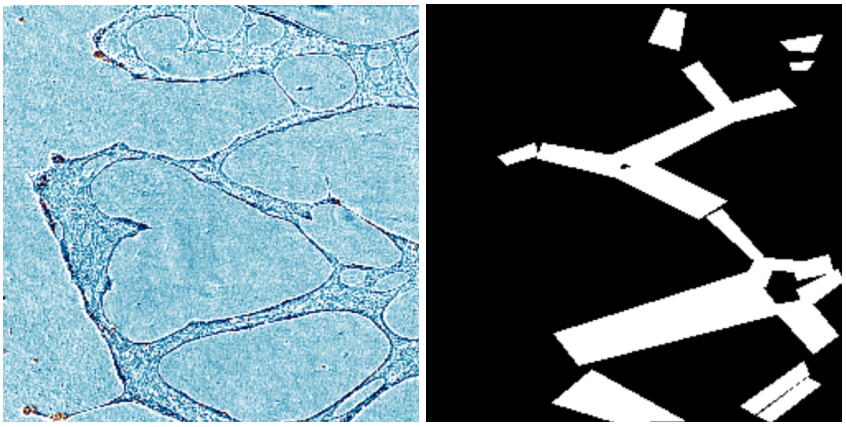

Fig. 6. Comparison of a rotated and resized image (left) with its corresponding binary mask (right), demonstrating the correct transformation of both the image and its JSON annotations.

including 50, 100, 200, and 300, and gradually increased the number of images we segmented during the training. This allowed us to observe how the model performed as it learned from a growing dataset.

In addition to experimenting with various model configurations, we also explored different approaches to labeling the images for segmentation. Initially, we labeled the images using polygons that closely followed the natural curve of the myotube. This approach aimed to capture the precise edges of the myotube structures. The results from this first model showed that the segmentation was quite accurate along the edges, with the boundaries closely matching the actual shape of the myotubes. However, the overall accuracy of this model was 0.70, indicating that while it was effective in detecting the myotube borders, it struggled with other aspects of the segmentation task.

To improve the model's performance, we decided to experiment with a second labeling strategy, this time using rectangular shapes for the segmentation. Our reasoning behind this adjustment was that by including the broader areas around the myotubes, we might help the model better distinguish the boundaries and learn more effectively. This approach provided a different perspective on the myotube structures, potentially giving the model a clearer context for segmentation.

By implementing this second approach with rectangular labeling, we hoped to improve the model's ability to capture the myotube edges and include the surrounding areas in the segmentation. While we did not immediately observe a drastic improvement in accuracy, this approach helped the model better generalize to different shapes and sizes of myotubes, ultimately contributing to a more robust performance in the final configuration.

4 Results

This section presents the results obtained from the evaluation of the U-net model for image segmentation. Below we will show two different configurations of the model, the first configuration shows a suboptimal performance, while the second configuration will show the most optimized possible, varying the number of epochs, number of images and other training configurations. Results, metrics and visual examples of the segmentation performed are detailed.

4.1 Evaluation criteria

To evaluate the performance of the model, the following metrics were used.

- **Dice coefficient**: Measures the similarity between predicted and true segmentation. Values close to 1 indicate a high level of accuracy.
- **IoU (Intersection over Union)**: Proportion between the intersection area and the total unit area between the predicted and true segmentation.
- **Accuracy**: Proportion of correctly classified pixels.

4.2 Evaluated configurations

Both configurations were tested with 80 percent training and 20 percent validation.

Configuration 1 (Suboptimal Performance) This configuration was trained with 200 epochs, without data augmentation, with a total of 117 images and a batch size of 8. The results were as follows:

Metric	**Value**
Dice coefficient	0.75
IoU (Intersection over Union)	0.60
Accuracy	0.85

Table 1. Metrics for Configuration 1 (Suboptimal Performance)

Configuration 2 (Optimal Performance) The second configuration was trained with 300 epochs, using data augmentation that includes rotations (flip, 90, 180, and 270) reaching a total of 585 images.

Metric	Value
Dice coefficient	0.90
IoU (Intersection over Union)	0.80
Accuracy	0.92

Table 2. Metrics for Configuration 2 (Optimal Performance)

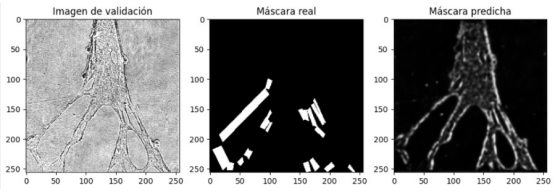

Fig. 7. Segmentation generated by Configuration 1 (Suboptimal Performance)

4.3 Visual comparison and observations

Below are examples of segmentation generated by both configurations.

We can see that configuration 2 significantly outperformed configuration 1. We can observe an area of poor segmentation in 7, where it is clearly seen that the CNN did not learn correctly, while in 8, a segmentation very similar to the real mask is observed.

4.4 Practical application example: Generation of time lapses

Next, we will show a practical example of our model, generating a timelapse using the predictions of our optimized model. The original images and a completely different set of images that the model was trained on were segmented and combined into a video, demonstrating how the model can be applied to continuous monitoring tasks.

- **See the video 1 on YouTube:** Click here to watch the video.
- **See the video 2 on YouTube:** Click here to watch the video.

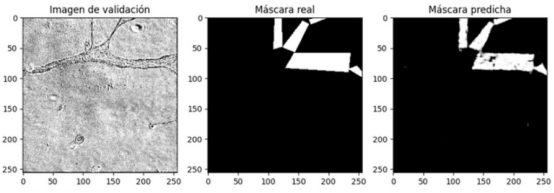

Fig. 8. Segmentation generated by Configuration 2 (Optimal Performance)

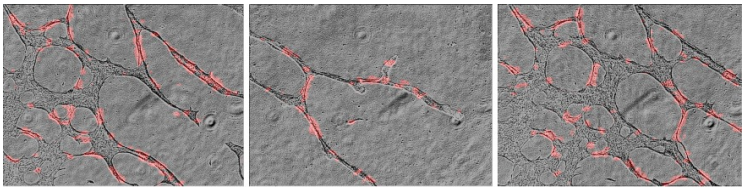

Fig. 9. Segmentation generated by Configuration 2 and final model(Optimal Performance)

5 Conclusion

In this study, the performance of a **U-Net model** for image segmentation can be reflected using two different training configurations. The results showed that Configuration 2, which incorporates data augmentation and a higher number of epochs, is significantly superior to the first configuration in terms of quality and accuracy when segmenting.

The metrics used highlight the importance of a solid training process to achieve the best possible results. Visual comparisons reinforce these findings, showing solid differences between one configuration and another.

We can conclude that this type of deep learning-based model not only helps to optimize the image segmentation process automatically, but also significantly reduces human error and the time required to perform the segmentation manually. These results are particularly attractive for practical applications, such as medical analysis, where fast and reliable segmentation is required to make sound decisions.

Future research could focus on expanding the dataset, exploring alternative network architectures, and automatic myotubes counting.

References

1. Intisar Rizwan I Haque and Jeremiah Neubert. Deep learning approaches to biomedical image segmentation. *Informatics in Medicine Unlocked*, 18:100297, 2020.
2. Narinder Singh Punn and Sonali Agarwal. Modality specific u-net variants for biomedical image segmentation: a survey. *Artificial Intelligence Review*, 55(7):5845–5889, 2022.
3. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
4. DR Sarvamangala and Raghavendra V Kulkarni. Convolutional neural networks in medical image understanding: a survey. *Evolutionary intelligence*, 15(1):1–22, 2022.
5. Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9:611–629, 2018.