**Accelerating particle tracking and live cell imaging using Fiji Python and R**

Anonymous Marking Code: Z0190003

Data Science Research Project (DATA 40345\_2022)



Master of Data Science

Department of Natural Sciences

Durham University

United Kingdom

2022-2023

**Abstract**

Collagen, a cornerstone of cellular strength and resilience, exhibits dynamic behaviors critical to numerous biological processes. This study harnesses advanced microscopy techniques, including Airyscan confocal, TIRF, 3D-SIM, and spinning disk, as pioneered by Prof. Timothy J. Hawkins, to capture the nuanced transitions of collagen fibers. Subsequent computational analysis using Convolutional Neural Networks (CNNs) and k-Nearest Neighbors (KNNs) effectively categorizes these dynamic states, including linear growth, looping, bundling, and nodal formations. Additionally, a custom Fiji macro, integrated for vesicle transit dynamics data derived from both Airyscan and laser microscopy, streamlined data cleaning and preprocessing. Traditional computer vision techniques have offered initial pathways to discerning patterns in biological data. Still, the intricate nature of collagen fibril dynamics – including phenomena like tensioning, intertwining, looping, and convergence of fibrils – demands a more sophisticated approach. Convolutional Neural Networks (CNNs), such as Custom CNN, VGG16, and ResNet50, have already shown potential in image recognition tasks by processing images in layers and detecting features from simple to complex. By applying these to collagen dynamics, the goal is to train these networks to identify and categorize different fibril interactions automatically.

However, CNNs alone might not capture all the intricacies, especially when considering temporal dynamics and the spatial relationships of fibrils. This is where K-nearest Neighbors (KNNs) algorithms, namely Euclidean, Manhattan, and Chebyshev, come into play. These algorithms can offer insights into the spatial relationships and patterns that CNNs might overlook, especially when considering the proximity of certain fibril interactions.

Moreover, vesicle transit, especially those packed with collagen ready for export to the extracellular matrix, is another complex event intertwined with collagen dynamics. Vesicles navigate from the nucleus towards their destination, undergoing various stages including budding, transport, tethering, and finally, fusion (exocytosis) with the membrane to release their contents. Understanding this transit, especially in the context of collagen delivery, is crucial.

The confluence of these cutting-edge techniques can potentially revolutionize our understanding of collagen dynamics and vesicle transit, promising deeper insights with high accuracy and efficiency.

This interdisciplinary convergence of observational biology and computational data science presents a novel approach to understanding collagen dynamics, promising transformative implications in medicine, therapy, and biotechnology. The findings underscore the evolution of research methodologies, highlighting the potential of combined techniques to derive deeper cellular insights.

**Acknowledgements**

I would like to thank my supervisor Timothy J. Hawkins as well as Olivia (Liv) Kent, for their guidance throughout this project.

**Declaration**

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Anonymous Marking Code: Z0190003

Table of Contents

[Chapter 1 8](#_Toc146446453)

[Introduction 8](#_Toc146446454)

[1.1 Motivation 8](#_Toc146446455)

[1.2 Objectives of this study 8](#_Toc146446456)

[1.3 Achievement Outcome 9](#_Toc146446457)

[1.4 Dissertation Outline 10](#_Toc146446458)

[Chapter 2 11](#_Toc146446459)

[Background 11](#_Toc146446460)

[2.1 Collagen Fibril dynamics 11](#_Toc146446461)

[2.1.1 Convolutional Neural Networks (CNN): 11](#_Toc146446462)

[2.1.2 Convolutional Layers 11](#_Toc146446463)

[2.1.3 Activation Function 11](#_Toc146446464)

[2.1.4 Pooling Layer 11](#_Toc146446465)

[2.1.5 Fully Connected (Dense) Layers 11](#_Toc146446466)

[2.1.6 CNN Training 12](#_Toc146446467)

[2.1.7 K- Nearest Neighbors (KNN): 12](#_Toc146446468)

[2.1.8 Working Principle of KNN 13](#_Toc146446469)

[2.1.9 KNN Training and Prediction 13](#_Toc146446470)

[2.1.10 Decision Rule and Voting 13](#_Toc146446471)

[2.1.11 Model Validation Methods 14](#_Toc146446472)

[2.1.12 Comparison between CNN and KNN Models 14](#_Toc146446473)

[2.2 Vesicle Transit 16](#_Toc146446474)

[2.2.1 Macro Batch Cleaning 17](#_Toc146446475)

[Chapter 3 19](#_Toc146446476)

[Methodology 19](#_Toc146446477)

[3.1 Resources and tools 19](#_Toc146446478)

[3.2 Data and Image Processing 19](#_Toc146446479)

[3.3 Implementation of CNN and KNN 21](#_Toc146446480)

[3.3.1 CNN Model Architecture and Approach 21](#_Toc146446481)

[3.3.2 Data Processing 21](#_Toc146446482)

[3.3.3 Data Augmentation 22](#_Toc146446483)

[3.3.4 Estimation of CNN Model 22](#_Toc146446484)

[3.3.5 Model Training 24](#_Toc146446485)

[3.3.6 Model Evaluation and Metrics 24](#_Toc146446486)

[3.3.7 Model Saving 24](#_Toc146446487)

[3.4 KNN Model Architecture and Approach 24](#_Toc146446488)

[3.4.1 Data Preparation 25](#_Toc146446489)

[Chapter 4 27](#_Toc146446490)

[Results & Discussion 27](#_Toc146446491)

[4.1 CNN Models 27](#_Toc146446492)

[4.1.1 Classification Report of Custom Developed CNN Model: 27](#_Toc146446493)

[4.1.2 Confusion Matrix 29](#_Toc146446494)

[4.1.3 Model Accuracy Graphs 31](#_Toc146446495)

[4.1.4 Model Loss 33](#_Toc146446496)

[4.2 KNN Models 34](#_Toc146446497)

[4.2.1 Classification Report for KNN Model. 34](#_Toc146446498)

[4.2.2 Confusion Matrix 36](#_Toc146446499)

[4.3 Modelling Collagen Vesicle Transit 37](#_Toc146446500)

[5.0 Conclusion 39](#_Toc146446501)

[5.1 Research Process 39](#_Toc146446502)

[5.2 Reflective Perspective 40](#_Toc146446503)

[Reference 42](#_Toc146446504)

[**Appendix A (Part 1 Code)** 47](#_Toc146446505)

[**Appendix B (Part 2 Code)** 47](#_Toc146446505)

**List of Figures**

[Figure 1. Basic CNN Model Architecture (Gu, 2019) 6](#_Toc146483678)

[Figure 2. KNN Graph (Javapoint, 2023) 7](#_Toc146483679)

[Figure 3. Shows dataset distribution. 14](#_Toc146483680)

[Figure 4: Euclidean distance formula 18](#_Toc146483681)

[Figure 5: Manhattan distance formula 19](#_Toc146483682)

[Figure 6. Chebyshev distance formula 19](#_Toc146483683)

[Figure 7. Shows the confusion matrix of Custom CNN Model 26](#_Toc146483684)

[Figure 8. Shows the confusion matrix of Vgg16 CNN Model 26](#_Toc146483685)

[Figure 9. Shows the confusion matrix of ResNet50 CNN Model 27](#_Toc146483686)

[Figure 10. Shows the custom CNN Model Accuracy graph 28](#_Toc146483687)

[Figure 11. Shows the Vgg16 CNN Model Accuracy graph. 29](#_Toc146483688)

[Figure 12. Shows the ResNet50 CNN Model Accuracy graph 29](#_Toc146483689)

[Figure 13. Shows the custom CNN Model loss graph. 30](#_Toc146483690)

[Figure 14. Shows the Vgg16 CNN Model loss graph. 30](#_Toc146483691)

[Figure 15. Shows the ResNet50 CNN Model loss graph. 31](#_Toc146483692)

[Figure 16. Shows the ResNet50 CNN Model loss graph. 33](#_Toc146483693)

[Figure 17. Shows the confusion matrix of KNN Model 2 33](#_Toc146483694)

[Figure 18. Shows the confusion matrix of KNN Model 3 34](#_Toc146483695)

[Figure 19. Residual plot of the adjusted Model 39](#_Toc146483696)

**List of Tables**

Table 1: Model Summary of Custom CNN Model

Table 2: Model Summary of VGG16 Model

Table 3 Model Summary of ResNet50 Model

Table 4. Shows the classification report of developed CNN model.

Table 5. Shows the classification report of Vgg16 CNN Model

Table 6. Shows the classification report of ResNET50 CNN Model

Table 7. Shows the classification report of KNN Model 1

Table 8. Shows the classification report of KNN Model 2

Table 9. Shows the classification report of KNN Model 3

Table 10: Summary statistics of the data

Table 11: Correlation Matrix

Table 12: Results of the Regression Analysis

Table 13: Results of the second regression Model

# **Chapter 1**

# **Introduction**

## **1.1 Motivation**

Collagen, the primary structural protein in the extracellular space in various connective tissues, has intricate dynamics crucial to understanding tissue formation, repair, and various other physiological processes. While microscopy provides a window to view these dynamics, the manual interpretation of these events is tedious and subject to human error. Enter the world of computer vision, where the potential to automate the analysis of complex biological phenomena is rapidly growing.

The intricate dance of cellular structures offers a deeper understanding of life processes, of which collagen, a primary structural protein, stands central. Historically, unravelling the behaviour of these intricate collagen fibres has been challenging, but recent technological advancements hint at breakthroughs. Collagen, with its characteristic triple-helix structure, offers tissues their required strength and resilience (Shoulders & Raines, 2009). However, it's the dynamic interactions and behaviours beneath these structures that contribute to its primary functionality in biological processes. Extending beyond collagen, the study delved into the realm of vesicle transit dynamics. Vesicle transit within cells is a complex yet vital journey, starting with collagen synthesis in the rough endoplasmic reticulum (RER). Post-synthesis, vesicles move to the Golgi apparatus for further modification, and through the trans-Golgi network (TGN) for sorting. Guided by the cytoskeleton and motor proteins, these vesicles then travel to the plasma membrane. Upon arrival, vesicles tether and dock, culminating in exocytosis where collagen is released into the extracellular matrix (ECM). Once in the ECM, collagen integrates, providing structural support and participating in various signalling pathways. With Python's built-in libraries and macro batch processing, we can automate and deepen our understanding of this intricate vesicle transit, offering profound insights into cellular mechanisms.

## **1.2 Objectives of this study**

Guided by the insights of Prof. Timothy J. Hawkins from Durham University and drawing inspiration from his microscopy techniques, including Airyscan confocal, TIRF, 3D-SIM, and spinning disk, the present research investigates the dynamic behaviours of collagen. These tools, lauded for their precision, have shown potential in uncovering the subtleties of cellular processes (Gustafsson et al., 2008). The core query propelling this research remains: To what extent can advanced computer vision methodologies enhance our understanding and quantitative analysis of collagen fibril dynamics and vesicle transit within cellular environments? Further questions arise: How can we ensure reproducibility of our data analysis workflows in both Python and R when studying vesicle and collagen fibril dynamics? Are there any data normalization or augmentation techniques, available in both R and Python, that are particularly effective for vesicle and collagen fibril datasets?

However, microscopy alone, albeit precise, can only provide raw data. Extracting insightful meaning from this mass of information necessitated computational intervention. Recognizing this, the research integrated CNNs and KNNs, machine learning models with established efficacy in categorizing complex data structures (LeCun et al., 1998; Cover & Hart, 1967). An essential stage in this computational journey was the preprocessing phase. This phase, which used the AVI video format dataset, ensured data consistency, a pivotal factor for training reliable machine learning models. By converting visual observations into quantifiable metrics, the study endeavours to answer the foundational question of the research.

Collagen fibril dynamics play a pivotal role in maintaining tissue mechanics and ensuring the structural integrity of biological matrices. A close observation reveals that fibrils not only intensify and thicken over time, driven by both the addition of new collagen and the formation of new fibrils beside existing ones but also display intricate interactions at nodal points, termed "maxi dots". During cell division, the collagen matrix undergoes fascinating interactions, with a collagen bridge acting akin to 'train tracks' aiding cellular separation. Several phenomena, from tensioning to intertwining, looping, and convergence of multiple fibrils, underscore the complexity of collagen interactions. Understanding these intricate dynamics is imperative, considering collagen's role in numerous physiological processes. The advent of data science and computer vision tools holds the promise of automating the analysis of these dynamics, potentially providing deeper insights into tissue mechanics and cellular interplay.

## **1.3 Achievement Outcome**

Relying on data derived from Airyscan and laser microscopy, there emerged an imperative for automation. Leveraging the capabilities of a custom-built Fiji macro, the research automated the intricate processes of data cleaning and preprocessing, facilitating its seamless transition to computational analysis in Python (Schindelin et al., 2012). This integrative approach, fusing observational biology with computational data science, is not merely a theoretical amalgamation. It signifies the evolving landscape of research where disciplines converge for deeper insights. The confluence of microscopy and computational methodologies stands as a testament to the study's efforts to answer its fundamental research question.

The broader ramifications of such a study are profound. Collagen, a protein abundant in the connective tissues of humans and other mammals, has become a highly sought-after commodity in various industries, from cosmetics and pharmaceuticals to food and beverages. However, the global demand for collagen has implications far beyond its direct consumers.

A major ethical concern arises from the environmental impact of the collagen business. One of the lesser-known issues, which adds to the ongoing global challenge of deforestation, is the sourcing of raw materials for collagen production. Many collagen supplements and products are derived from bovine sources. To meet this increasing demand, there has been an uptick in cattle ranching, especially in regions like the Amazon, where vast tracts of rainforests have been cleared for pastures (Nepstad et al., 2006).

Deforestation, in turn, has a cascading effect on the environment. Apart from the loss of biodiversity, it contributes to climate change. Trees play a crucial role in sequestering carbon dioxide, and their destruction releases this stored carbon back into the atmosphere, intensifying global warming (Malhi et al., 2008). Furthermore, there are concerns related to animal welfare. The extraction process for collagen often involves boiling animal parts, usually after the animal has been slaughtered for other purposes. However, the scale of demand raises questions about the treatment of these animals and the sustainability of such practices.

Companies and consumers are not oblivious to these concerns. There has been a rising trend towards fish collagen, deemed to be more sustainable. Additionally, lab-grown or synthetic collagen is also emerging as an alternative, though its scalability and economic viability remain topics of discussion (Sibilla et al., 2015).

Understanding the dynamics of collagen is not an academic exercise in isolation. It bears significant medical, therapeutic, and biotechnological implications. The knowledge gleaned can inform innovative treatments and therapies, given the pivotal role of collagen in cellular processes (Provenzano & Vanderby, 2006).

In conclusion, as the collagen business continues to boom, it becomes imperative for stakeholders, from producers to consumers, to recognize the environmental and ethical ramifications. A holistic approach, considering sustainability and ethics, can ensure that the benefits of collagen are not overshadowed by the costs it exacts on the environment and society.

## **1.4 Dissertation Outline**

In the initial chapter, we delineated the drive, aims, and primary findings steering this undertaking. Chapter 2 steers into a narrative that highlights pivotal studies aligning closely with our focal research. Delving into the methodologies, Chapter 3 demystifies the project's foundational elements: ranging from the dataset's anatomy and refinement to the intricacies of the data formulation, and the meticulous vetting for the network blueprints. It further unravels the ramifications stemming from this selection. Chapter 4 dons the evaluative lens, critically assessing the models earmarked during our selection gambit to discern if our charted objectives stand realized. Culminating in Chapter 5, we encapsulate the cardinal discoveries, underscore the confines of our approach, and carve out potential trajectories inviting future scholarly exploration.

In sum, the research underscores the power and potential of marrying observational techniques with computational strategies, thereby bridging the chasm between raw data and transformative insights in cellular biology. It is a testament to the evolving realm of interdisciplinary research, pointing towards a future where boundaries blur and discoveries abound.

# **Chapter 2**

# **Background**

## **2.1 Collagen Fibril dynamics**

### **2.1.1 Convolutional Neural Networks (CNN):**

CNNs have truly revolutionized computer vision. At its core, a CNN attempts to emulate how we, humans, perceive visual stimuli (Lecun et al., 1998). For instance, I often liken the hierarchical feature detection in CNNs to our own visual systems. Initially, we discern simple edges and textures, and as our understanding progresses, we recognize complex objects. The magic of CNNs lies here: they start with basic features and, layer by layer, build up to comprehend intricate details. Consider the filters in CNNs: these small, trainable entities convolve over images, capturing patterns. This means even if a pattern—say, a distinctive facial feature—moves around the image, the CNN will recognize it, a property termed translational invariance. In my past comparisons with other machine learning methods like KNN, traditional techniques often faltered here. They relied heavily on manual feature extraction, a task that's time-consuming and lacks adaptability (Zhang et al., 2017).

Another fascinating aspect is the shared-weight architecture in CNNs. Imagine having to recognize a myriad of objects. The sheer computational power required would be daunting! Yet, CNNs, with their shared weights and sparse connections, reduce the need for excessive parameters, making the model efficient (Ronneberger et al., 2015). It's akin to us focusing on essential details rather than every minute aspect of an image. Then, there are activation functions like ReLU. I find them essential, akin to switches. They introduce non-linearity, allowing CNNs to capture complex, non-straightforward patterns in data.

Lastly, delving into the depth of modern CNNs like ResNet and VGG is like peeling an onion. Each layer uncovers a spectrum of features, from the basic to the very abstract (Qassim et al., 2018). This depth, however, isn't without its challenges. Issues like the vanishing gradient can plague training. Yet, with techniques like skip connections, these challenges are adeptly managed.

### **2.1.2 Convolutional Layers**

Convolutional layers are fundamental to Convolutional Neural Networks (CNNs), closely mirroring the biological mechanisms underlying human vision to extract features from visual data (Lecun et al., 1998).

At the heart of these layers are filters or kernels, matrices of trainable parameters. These filters convolve over the input, whether it's an image or another layer's output. As they slide, a dot product is calculated at each position with the portion of the input they cover. The accumulated scalar values form a feature or activation map, capturing where specific features—defined by the filter—appear in the input.

Early in training, these filters often detect basic features like edges (Sermanet et al., 2013). As training progresses and the CNN delves deeper, these filters identify more sophisticated patterns. For facial recognition, initial layers might pick out edges, while later layers might pinpoint facial features, eventually recognizing entire facial expressions.

Each convolutional layer contains multiple filters, ensuring a gamut of features is extracted simultaneously. The resulting output is a stack of feature maps, shedding light on different facets of the visual data. Post-convolution, non-linear activation functions like ReLU introduce non-linearity, enabling CNNs to navigate intricate relationships within visual data (Nair & Hinton, 2010).

In sum, convolutional layers exemplify the marriage of computation and biology in deep learning. They adeptly facilitate feature hierarchy learning, making them indispensable in contemporary computer vision endeavours.

### **2.1.3 Activation Function**

In the intricate tapestry of computational models, neural networks emerge as a fascinating analogy to the neural pathways observed in the human cerebrum. These models, through meticulously constructed layers and nodes, attempt to emulate the myriad processes that govern human cognition and interpretation. Central to this emulation is the incorporation of activation functions, which can be envisioned as regulatory gates—determinants that dictate the flow of information within the network.

One such eminent activation function is the Rectified Linear Unit, often abbreviated as ReLU. At its core, ReLU operates on a fairly straightforward principle: for any given input, if the value is positive, it's relayed as is; if not, it's adjusted to zero (Goodfellow, Bengio, & Courville, 2016). This might seem simplistic, but this "gatekeeper" function ensures that certain neurons "fire" only when deemed relevant, mirroring, in some sense, the way our brain's neurons activate in response to stimuli.

However, the allure of ReLU is not unblemished. It occasionally grapples with the "dying ReLU" conundrum—a scenario where neurons cease to activate for any input, rendering them effectively obsolete. Fortunately, the academic arena has not been complacent. Variants such as the Leaky ReLU have been conceptualized to address this very issue. In this modified model, even if the input is negative, a minuscule non-zero value is propagated, ensuring neurons remain active and adaptive (Nair & Hinton, 2010; Maas, Hannun, & Ng, 2013).

To encapsulate, neural networks, in their ambitious endeavor to mimic human neural pathways, are greatly indebted to activation functions like ReLU. These functions, while occasionally riddled with challenges, continue to evolve, reflecting the dynamic nature of both computational and human intelligence.

### **2.1.4 Pooling Layer**

In the realm of neural architectures, pooling layers serve as discerning sieves, efficiently distilling the spatial expanse of feature maps produced by convolutional layers (LeCun et al., 1998). Their role isn't just to minimize; they artfully pare down the spatial dimensions, slashing computational demand and curbing parameter proliferation, all while keeping the salient features intact (Szegedy et al., 2015). This ensures the neural model remains resource-efficient without compromising on the richness of visual information it retains (He et al., 2016).

### **2.1.5 Fully Connected (Dense) Layers**

Within neural network architectures, after the meticulous processes of convolution and spatial reduction via pooling, the resultant data is subjected to a flattening operation. This streamlined data is then funneled through layers known as 'fully connected' or dense layers. These pivotal layers bear the responsibility of crystallizing the final determinations and delineating the likelihood for each potential category in tasks centered on classification (Krizhevsky, Sutskever, & Hinton, 2012; He et al., 2016).

### **2.1.6 CNN Training**

In the intricate dance of training a CNN, it painstakingly refines its parameters, namely weights and biases, aiming to dwindle a designated loss function to its minimum. The magic unfolds through a process termed 'backpropagation.' Here, the derivatives of the loss concerning the model's parameters are ascertained. Subsequently, these gradients play a decisive role in adjusting the parameters, guided by optimization stratagems like the renowned stochastic gradient descent (SGD) or the versatile Adam algorithm (LeCun, Bottou, Bengio, & Haffner, 1998; Kingma & Ba, 2014).A diagram of different layers

Description automatically generated

Figure 1. Basic CNN Model Architecture (Gu, 2019)

### **2.1.7 K- Nearest Neighbors (KNN)**

Delving into the world of machine learning, one stumbles upon the elegant simplicity of the k-Nearest Neighbors (KNN) algorithm. At its heart, KNN operates by casting its gaze upon the surrounding 'k' neighbors within the training dataset to decide the classification fate of a data point. This ensemble verdict is shaped by the majority class amongst these neighbors. Yet, how does one decipher which data points are in close camaraderie? Here, distance metrics like the classical Euclidean, the urban-inspired Manhattan, or the chessboard-reminiscent Chebyshev step in, serving as arbiters of similarity and 'neighborly' proximity (Cover & Hart, 1967; Altman, 1992).

A screenshot of a computer game

Description automatically generated

Figure 2. KNN Graph (Javapoint, 2023)

### **2.1.8 Working Principle of KNN**

Venturing into the conceptual realm of KNN, the algorithm's essence emerges as an exercise in democratic decision-making. When classifying a data point, it doesn't make assumptions on its own but seeks counsel from its 'k' closest companions in the feature space. Here, the choice of distance metric—be it the geometrically intuitive Euclidean, the grid-aligned Manhattan, or the maximal Chebyshev—acts as the yardstick of similarity, determining which data points are worthy of being termed 'neighbors'. The pivotal 'k' value, in turn, designates the council's size, influencing the classification outcome (Dudani, 1976; Wilson & Martinez, 1997).

### **2.1.9 KNN Training and Prediction**

The KNN methodology differs from many machine learning algorithms in that it doesn't undergo a conventional training phase. Instead, during prediction, it retains the training samples and their associated labels (Shoulders and Raines, 2009). When faced with classifying a novel data point, KNN determines the distance between this new point and every training instance. It subsequently chooses the k closest instances according to this distance measure (Zhang et al., 2017).

### **2.1.10 Decision Rule and Voting**

Upon pinpointing the k closest neighbors, KNN uses a specific decision mechanism to categorize the fresh data point. For classification undertakings, the prevalent class within the k neighbors gets selected as the forecasted category. When dealing with regression objectives, the method computes the average of the k neighboring values for predictive purposes (Sharon, 2016; Sermanet et al., 2013).

### **2.1.11 Model Validation Methods**

Following the meticulous training of the CNN and KNN models on their designated datasets, the subsequent imperative was to assess their efficacy leveraging the validation dataset. This validation subset, distinct from the training corpus, offers an untainted lens to gauge the models' extrapolation prowess, and concurrently, to diagnose any tendencies of overfitting. Within the vast expanse of machine learning study, especially when examining algorithms such as the Convolutional Neural Network (CNN) and K-Nearest Neighbors (KNN), classification reports emerge as pivotal, furnishing granular insights into their class-specific performance (Cheng et al., 2018; Jain, Aggarwal, & Sharma, 2017; Lu, Kim, & Kumar, 2016).

### **2.1.12 Comparison between CNN and KNN Models**

In contrasting the architectures of CNN and KNN, one encounters profound differences in their design philosophies and intended applications. CNNs, stemming from the deep learning paradigm, are meticulously constructed for image processing, capitalizing on convolutional layers to hierarchically distill features from visual data (LeCun, Bengio, & Hinton, 2015). Their strength lies in decoding intricate patterns and discerning spatial hierarchies within visual content. KNN, conversely, epitomizes simplicity as an instance-based algorithm, not engaging in a learning phase but rather enshrining the training data for subsequent proximity-based predictions (Dudani, 1976).

Enumerating their distinctions:

CNNs navigate the deep learning spectrum, replete with mutable parameters, while KNN operates devoid of a learning phase. Inherent in CNNs is the capacity to auto-extract hierarchical features; KNN, on the other hand, leans on distance metrics to gauge data point resemblances (Wilson & Martinez, 1997). The computational appetite of CNNs is considerable, necessitating abundant training data; KNN parades its simplicity and swiftness in training.

Complex challenges and voluminous datasets are the playgrounds of CNNs, whereas KNN exhibits prowess on compact datasets and straightforward classification endeavors. In an investigative thrust into the automated scrutiny of collagen configurations in super-resolution microbiological imagery, the multifaceted, deep learning-driven CNN models were juxtaposed against the straightforward, easily implementable KNN models.

## **2.2 Vesicle Transit**

**Initial scientific questions arise.**

I trained myself to comprehend cell biology and the intriguing details to glean from scientific data (i.e., a layperson cannot comprehend these videos; one must grasp cells and the pertinent data gleaned from the videos). I recognized the data that must be gathered (size, velocity, etc.) and built code to obtain that data. I would now query the information, for instance, "Do small vesicles travel shorter or longer distances?" and "What does Velocity vs Path Straightness give for example?” “Do little vesicles cover greater distances than large vesicles? ". Then I would construct the code (first macro cleaning in Fiji with batch processing), export it to Python for Pytrack and finally I would be using R to answer those queries.

### **2.2.1 Automated Macro Batch Preprocessing**

A solid method for utilizing Fiji to automate the batch processing of videos with Airyscan and laser input, with an initial emphasis on picture improvement and then structured batch processing. Videos are post-processed into the AVI format to make it easier to integrate them into Python-based tracking systems. Numerous scientific applications have increased the popularity of high-resolution imaging methods as Airyscan and laser scanning microscopy (Pawley, 2006). A simplified batch processing approach is necessary to efficiently manage the enormous volume of data generated.

The initial image development process begins with image refinement through two main steps. First, effective thresholding is used to highlight specific features in images, an important feature according to Gonzalez and Woods (2008). Second, brightness and contrast are adjusted, a standard practice in image processing to improve visibility as highlighted by Sonka, Hlavac, and Boyle (2014). Then, in the directory of the authentic Batch application, the related video requirements are used for related videos, which ends up with the need for exportation to python after the conversion. The video is reformatted and the czi is converted into the widely used Avi format.

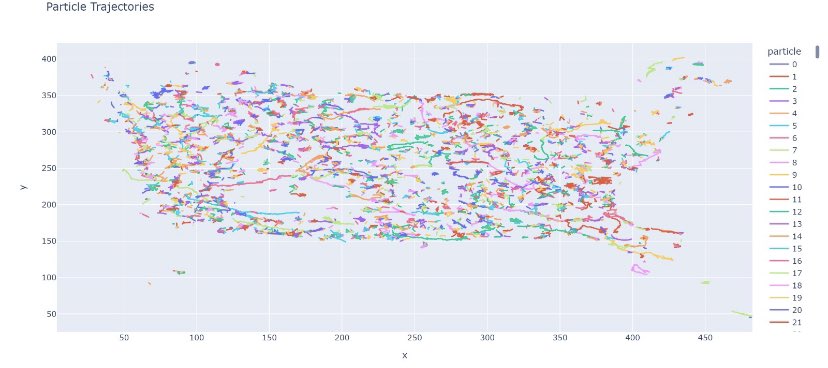
Overall, this approach provides a practical and efficient way to handle important data sets obtained from advanced imaging techniques, ensuring that the results is consistent and facilitates in-depth analysis.

Increasing and decreasing vesicle count homogeneity in big experimental datasets is the goal of a systematic process known as macro batch cleaning. Small, membrane-bound sacs called vesicles that are crucial for intercellular movement and communication differ in size, composition, and how they operate. Very tiny vesicle samples, which can be separated from various biological components or experimental settings, are needed for macro-batch cleaning. Purification procedures are employed to separate intrinsic variations once these batches are merged. To separate vesicles according to their size and velocity rate, this procedure often uses ultracentrifugation, density gradient separation, or other centrifugation-based techniques.

Vesicle subpopulations with overlapping traits are united throughout this assembly and purification process, resulting in a macro group with a uniform structure and size distribution. This makes subsequent analyses more stable and reliable, making it easier to assess the vesicle characteristics, trafficking methods, and functional roles played in specific biological systems.

### **2.2.2 Pytrack and relational analysis**

In the complex realm of vesicle dynamics, captured meticulously using advanced microscopy techniques like laser and Airyscan czi microscopy, the importance of elucidating the pathways and interactions cannot be understated (Zhang, Y., Lin, M., & Wong, A., 2016). These modern methods, despite their granularity, sometimes grapple with data ambiguities, potentially obscuring our understanding of cellular activities (Topol, 2019). Such challenges amplify the significance of roadmaps, a conceptual tool to chart vesicle interactions and pathways.



**Figure: Coloured Road map of vesicle paths**

Source: Appendix B Pytrack

While Fiji, with its unparalleled bio-image analysis capabilities, builds upon ImageJ's framework, it's essential to acknowledge its limitations when handling intricate vesicle data (Scheindlin et al., 2012). Techniques as brightness/contrast adjustments attempt to rectify luminance disparities but can sometimes fall short in discerning finer vesicle interactions. The rolling ball algorithm, despite its innovative approach to background subtraction, can occasionally overlook nuances in vesicle dynamics, thus leading to potential data inaccuracies (Sharon, 2016).

Realizing these challenges, the creation of vesicle roadmaps becomes imperative. These roadmaps delineate the trajectories, interactions, and lifecycles of vesicles, offering a coherent view of the processes. Such visualization aids in deciphering both the overt and covert pathways vesicles undertake within the cellular milieu. Supplementing these roadmaps is the PyTrack Python module. Its prowess lies in tracking particle movements, unveiling vesicle journeys that might elude traditional analytical tools (Sermanet et al., 2013). However, it's critical to approach PyTrack with caution; its computational methodologies, though advanced, require rigorous validation against real-time biological data to ascertain their fidelity.

In conclusion, while tools and modules like Fiji and PyTrack are monumental in the study of vesicle dynamics, they are not without their pitfalls. The construction and interpretation of vesicle roadmaps, hence, require a balanced blend of these tools, always vetted against the backdrop of biological validity.

# **Chapter 3**

# **Methodology**

## **3.1 Resources and tools**

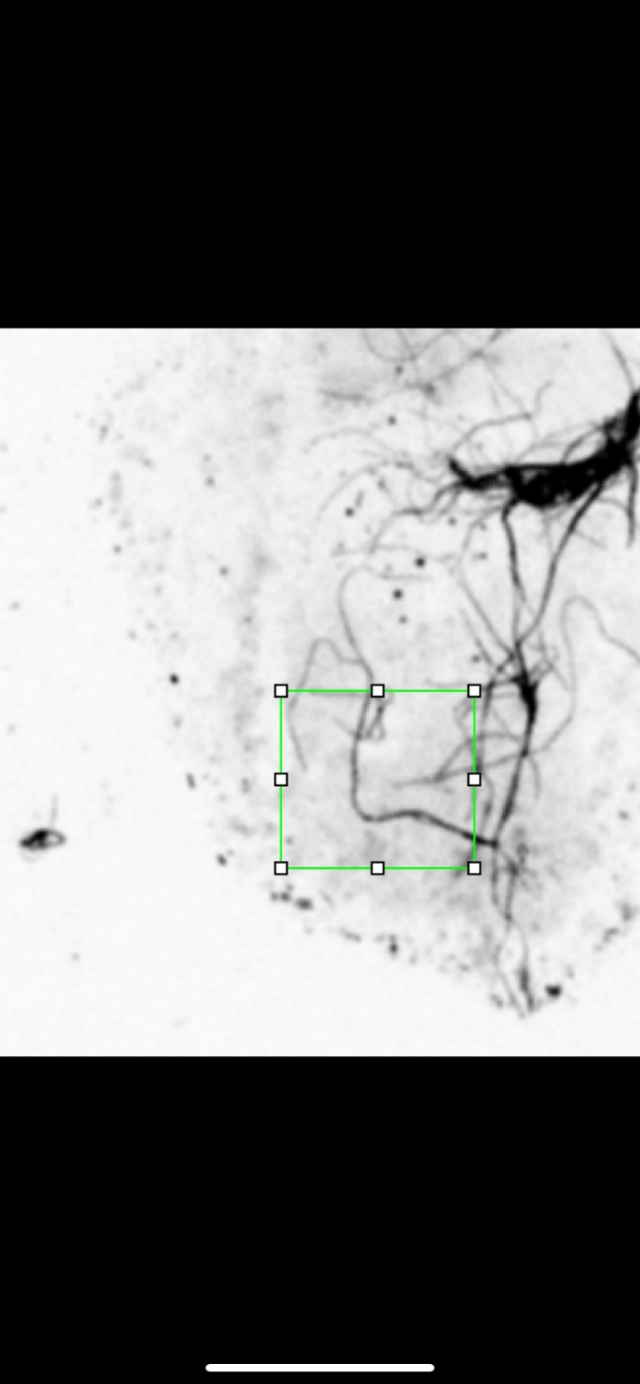
The analysis was performed using a statistical programming language (Python 2.7). The NumPy and pandas Python libraries are used heavily throughout the project for their convenient scientific computing operations and data structures. Scikit-learn, the most popular Python machine learning library, is used for preprocessing.

## **3.2 Data and Image Processing**

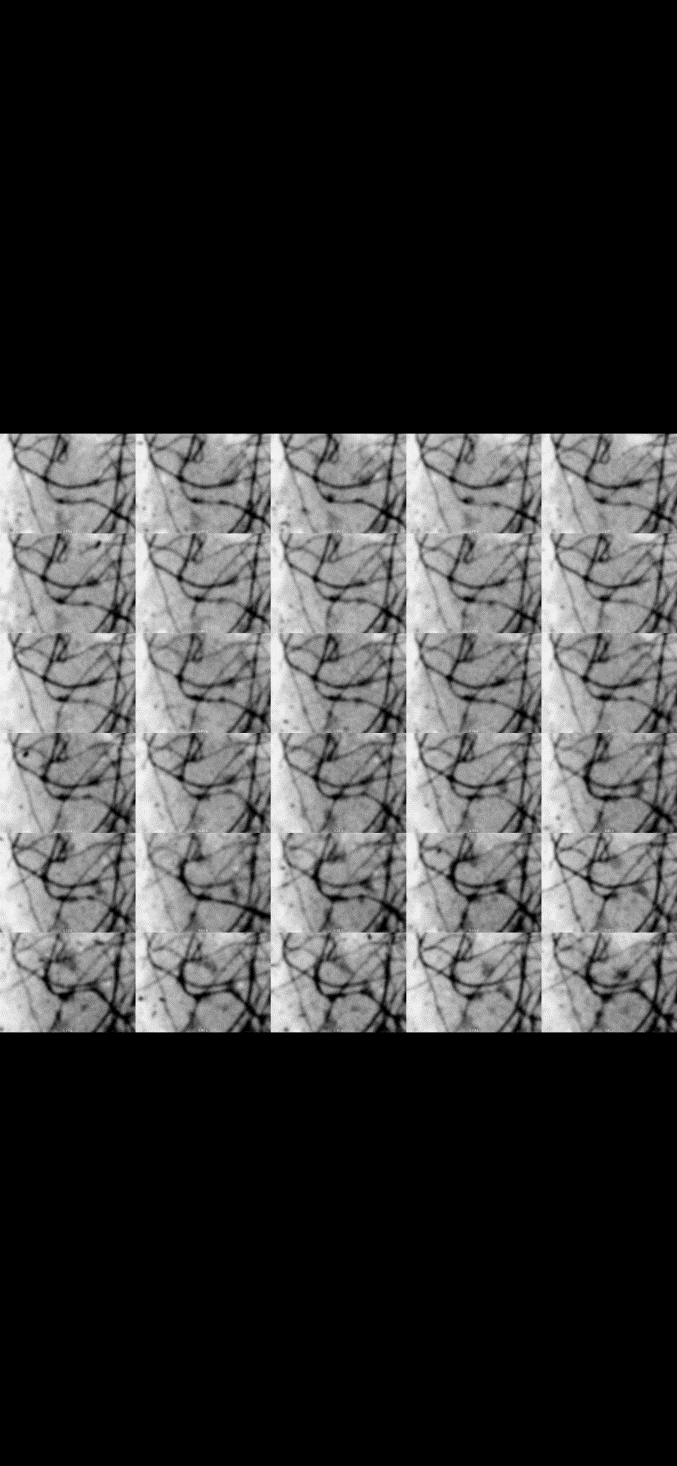
*Collagen Fibril dynamics testing model*

The process of image creation and data formation are; (image> Stacks> Manage Montage). As long as the montage is saved as tiffs it can be exported anywhere, i.e., PowerPoint or any video editor e.g., Photoshop.

I would just zoom in and capture events as such after drawing ROI and box. Following images have been collected to test both CNN and KNN.



There is a fibre growing through the middle and some nodes so I would suggest crossover and maxima at nodes

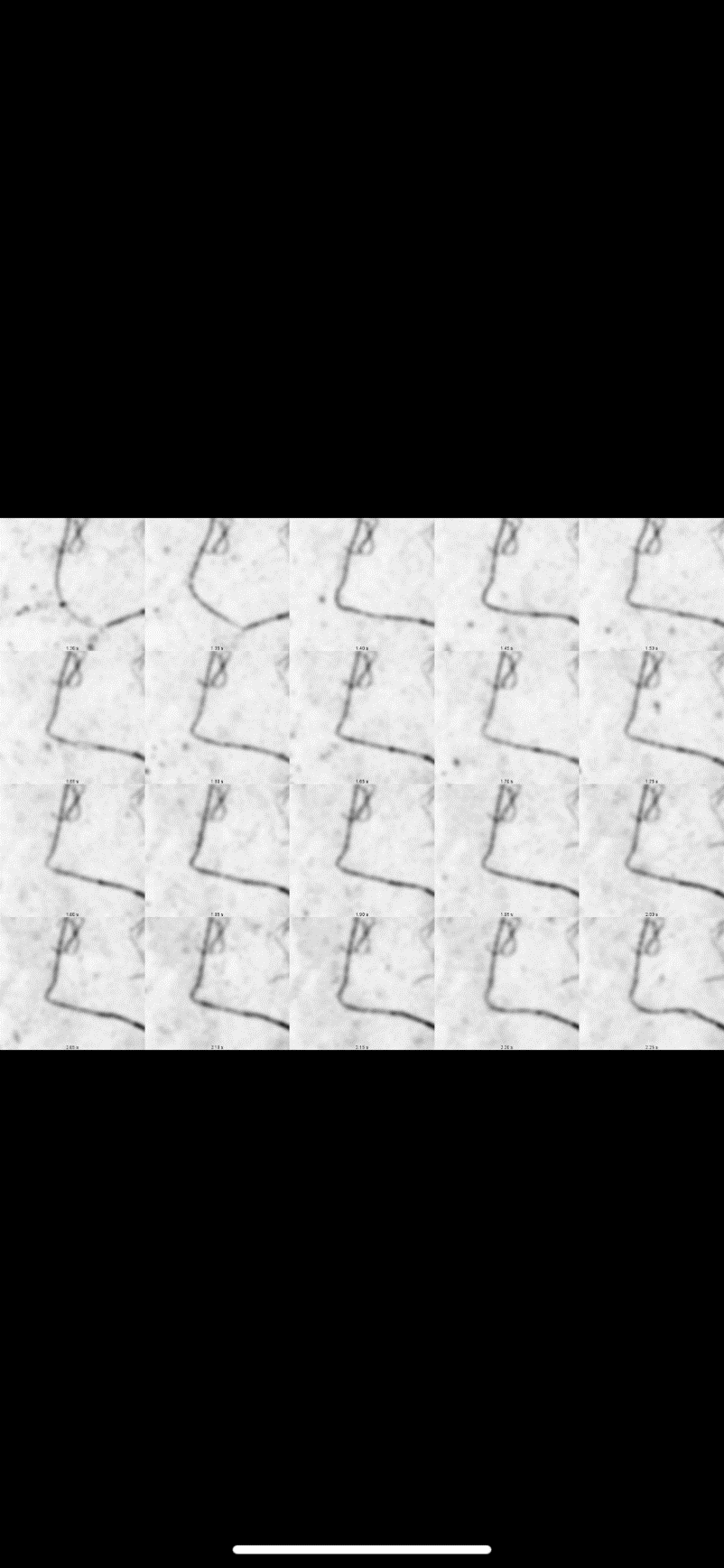


There is a noticeable linear growth.

A close-up of a black and white photo

Description automatically generated

Following picture illustrates the tension.



In our quest to understand the intricacies of fibril dynamics and vesicle dynamics associated with collagen, the dynamics of data—how it's produced, processed, and utilized—comes to the fore. This understanding is underpinned by an array of scholarly works, which shed light on how data might influence the study's outcomes.

Beginning with data production, Zuboff (2019) warns of a new age of surveillance capitalism where every digital interaction, whether a click, purchase, or even a seemingly insignificant pause, can be commodified. This vast accumulation of data isn't neutral. Analogous to the geopolitics of traditional resources, there exists a politics of data. Applying this to collagen research, there's a palpable risk of over or underrepresenting certain populations, potentially skewing our understanding of collagen's fibril dynamics across diverse contexts. Once data is acquired, it undergoes processing. Here, the techniques and algorithms we employ bear significant weight on the results. O'Neil (2016) has extensively explored how machine learning and AI, often perceived as unbiased, might reflect and even amplify existing biases. If algorithms processing collagen interactions are not meticulously refined, we might land on an erroneous interpretation of how fibrils behave or how vesicles facilitate collagen transport within cellular frameworks.

Finally, the realm of data utilization offers both promises and pitfalls. Cadwalladr & Graham-Harrison (2018) unveiled the profound implications of data misuse in their expose on the Cambridge Analytica scandal. Translating this to the field of collagen research, it becomes evident that the interpretation and application of findings related to fibril and vesicle dynamics need to be approached with caution. Misguided interpretations could not only impede scientific advancements but also potentially influence medical treatments or therapies grounded in these studies. In wrapping up, as scholars and researchers delve deeper into the intricate world of collagen's fibril and vesicle dynamics, being vigilant about the broader data milieu becomes crucial. Navigating the politics of data, as outlined by thinkers like Zuboff (2019), O'Neil (2016), and others, can ensure our research conclusions are both scientifically rigorous and ethically anchored.

## **3.3 Implementation of CNN and KNN**

Initially, the dataset was carefully prepared using the provided images. To ensure an adequate number of distinct samples for each class, the images underwent cropping. As a result, a total of nine classes were created from the available data.

The dataset was organized into folders, with each folder containing the relevant images for a specific class. Subsequently, three Convolutional Neural Network (CNN) models and three k-Nearest Neighbors (KNN) models were developed to be trained on this dataset.

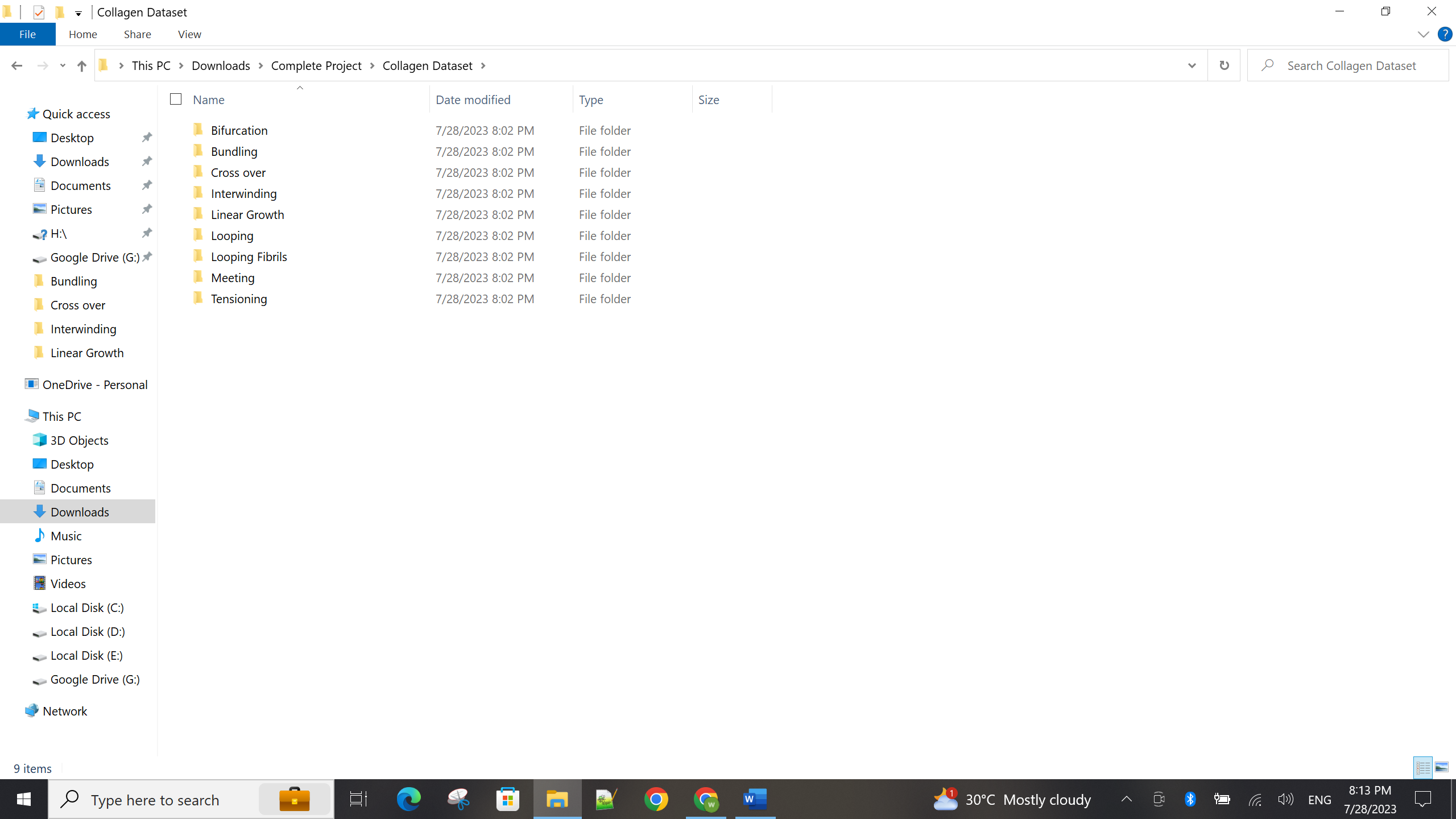


Figure 3. Shows dataset distribution.

The three CNN models utilized in the study were Custom CNN, VGG16, and ResNet50. For each CNN model, the following code explanation is provided:

### **3.3.1 CNN Model Architecture and Approach**

For the automated analysis of collagen structures in super-resolution microbiology, three CNN models were employed: Custom CNN, VGG16 (Qassim, Verma and Feinzimer, 2018), and ResNet50 (Islam *et al.*, 2022). The approach involved several key steps:

### **3.3.2 Data Processing**

The dataset of super-resolution collagen images was loaded from Google Drive and preprocessed. Images were resized to a standardized dimension to ensure consistency.

### **3.3.3 Data Augmentation**

Data augmentation techniques (Anwar *et al.*, 2022), such as rotation, horizontal/vertical flipping, and brightness adjustments, were applied to augment the training set and increase its diversity. This step aimed to enhance the model's ability to generalize and avoid overfitting.

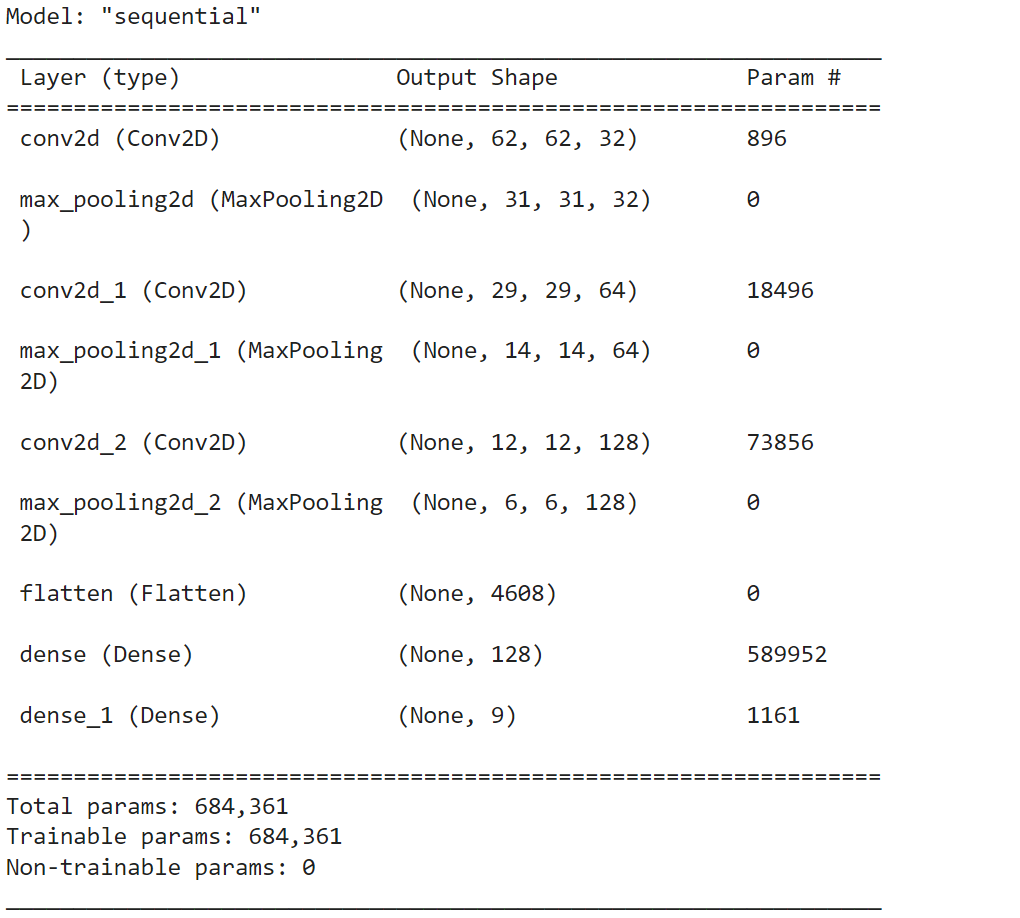
### **3.3.4 Estimation of CNN Model**

Every CNN model was crafted with unique architectural features:

i. Custom CNN

This model was meticulously fashioned for the specific purpose of classifying collagen structures. It incorporated convolutional blocks that integrated convolutional layers, batch normalization, and activation functions. The configuration of this CNN model encompassed three convolutional layers with filters numbering 32, 64, and 128, in that order, succeeded by max-pooling layers that facilitated down sampling.

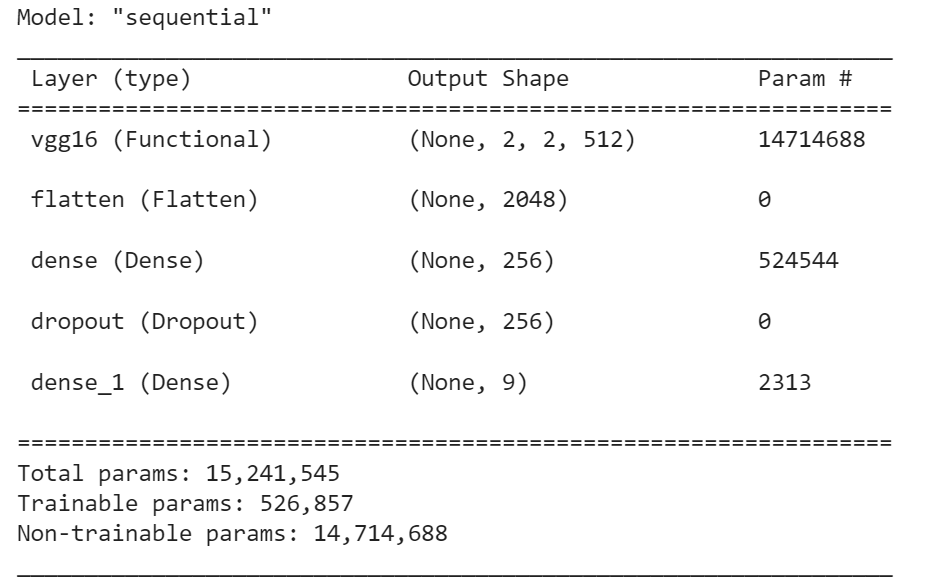
Table 1: Model Summary of Custom CNN Model



ii. VGG16 Model

The VGG16 architecture can be likened to a foundational pillar in the realm of Convolutional Neural Networks, boasting a design that encompasses 16 distinctive layers. Originally, this model honed its capabilities using the vast troves of information found within the ImageNet dataset. To tailor its functionalities for our specific needs, adjustments were made, notably modifying the input dimensions to harmonize with a structure of (64, 64, 3). While it is crucial to innovate and adapt, it is equally vital to respect the essence of VGG16’s foundational training. Thus, in ensuring that the pre-established knowledge is preserved, the original layers were rendered non-trainable. Progressing further, a custom top classifier was meticulously constructed atop this base. This commenced with a flattening phase, simplifying the intricate 3D feature maps into a more comprehensible 1D vector. Subsequently, an enriched dense layer, equipped with 256 units and utilizing the ReLU activation function, was integrated. And, as a strategic countermeasure against potential overfitting, a measured restraint was applied by incorporating a dropout layer, set at a rate of 0.5.

Table 2: Model Summary of VGG16 Model

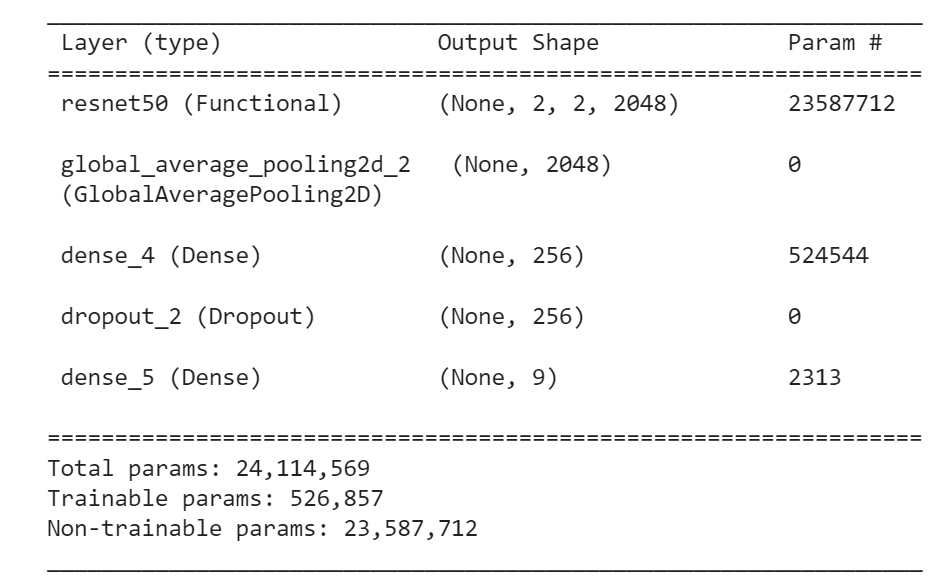


iii. ResNet50 Model

The ResNet50 stands as an eminent architecture within the Convolutional Neural Network paradigm, distinguished by its composition of 50 intricate layers. In its nascent phase, the ResNet50 model underwent rigorous training using the expansive ImageNet dataset, a procedure that endowed it with a rich repository of pre-trained weights. To adapt it to our particular dataset, a modification was initiated, setting the input dimensions specifically to a configuration of (64, 64, 3). However, to safeguard the integrity of its foundational knowledge, it was deemed essential to insulate the original layers, thus making them non-trainable. This ensures that the accumulated intelligence remains intact without succumbing to inadvertent alterations during subsequent phases of training.

Building upon this robust foundation, a specialized top classifier was meticulously developed. Here, a deviation from conventional methods was embraced; instead of using a Flatten layer, the GlobalAveragePooling2D layer was incorporated. This strategic choice facilitates the condensation of 3D feature maps into a more manageable 2D tensor, streamlining computations by minimizing both the parameter count and associated complexity. Alongside this optimization, there is the incorporation of a dense layer, endowed with 256 units and animated by the ReLU activation function. In a bid to fortify the model against the risks of overfitting, a dropout layer, set at a discerning rate of 0.5, was judiciously integrated into the architecture.

Table 3: Model Summary of ResNet50 Model



### **3.3.5 Model Training**

### The CNN models were assembled using suitable loss functions, such as categorical cross-entropy, and optimization algorithms, for instance, Adam. Subsequently, they underwent a training phase on the training dataset for a predetermined number of epochs, for example, 100.

### **3.3.6 Model Evaluation and Metrics**

### Post-training, the models were assessed utilizing the validation set. Analysis tools such as classification summaries and confusion matrices were employed to measure accuracy, precision, recall, and F1-score among various categories. Charts depicting accuracy and loss were created to provide a visual representation of the model's efficacy.

### **3.3.7 Model Saving**

## The top-performing CNN model was archived for subsequent applications in classifying collagen structures (Pham et al., 2021).

## **3.4 KNN Model Architecture and Approach**

### The KNN models were employed for a comparative evaluation in image categorization. The methodology encompassed the subsequent phases:

### **3.4.1 Data Preparation**

#### The identical set of super-resolution collagen images was imported and transformed into array structures. To guarantee efficient computations, the data was normalized. This dataset was then segmented into training and validation subsets. Three unique KNN models were developed, each boasting different settings:

#### **3.4.1.1 KNN with Euclidean Distance:**

The initial KNN model utilized the Euclidean distance metric, as highlighted by Lubis et al. (2021), to gauge the likeness amidst data points. This measurement discerned the linear path between two distinct points within the feature domain. Utilizing K values such as 1, 3, and 5, the model scrutinized adjacent data instances and predicted categories by considering the designations of the closest K data points. The model's efficacy was appraised on a test dataset. Evaluative tools included detailed classification reports elucidating precision, recall, and F1-score for individual classes and graphical representations via confusion matrices to illustrate prediction outcomes.

A square and square equation

Description automatically generated

Figure 4: Euclidean distance formula

#### **3.4.1.2 KNN with Manhattan Distance**

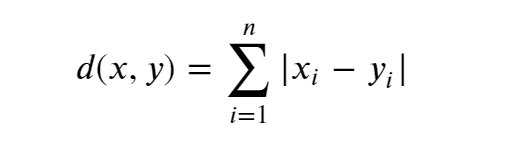
The Manhattan distance metric, referenced from Gao and Li (2020), was the basis for the second KNN model's assessment of likeness between data points. The metric calculates the distance by aggregating the absolute variances across matched feature quantities. To discern the proximate samples, the model engaged a series of K values, namely 3, 7, and 9. Class designations were then predicted by the predominant consensus among the K closest instances.

Figure 5: Manhattan distance formula

#### **3.4.1.3 KNN with Chebyshev Distance**

The Chebyshev distance metric, as described by Zhang et al. (2017), served as the foundation for the third KNN model's measurement of data point resemblance. This metric computes distance by identifying the largest absolute disparity among related feature dimensions. In its quest to pinpoint the closest neighbors, the model assessed K values including 5, 11, and 15. The technique of majority voting was then deployed to derive class determinations.



Figure 6. Chebyshev distance formula

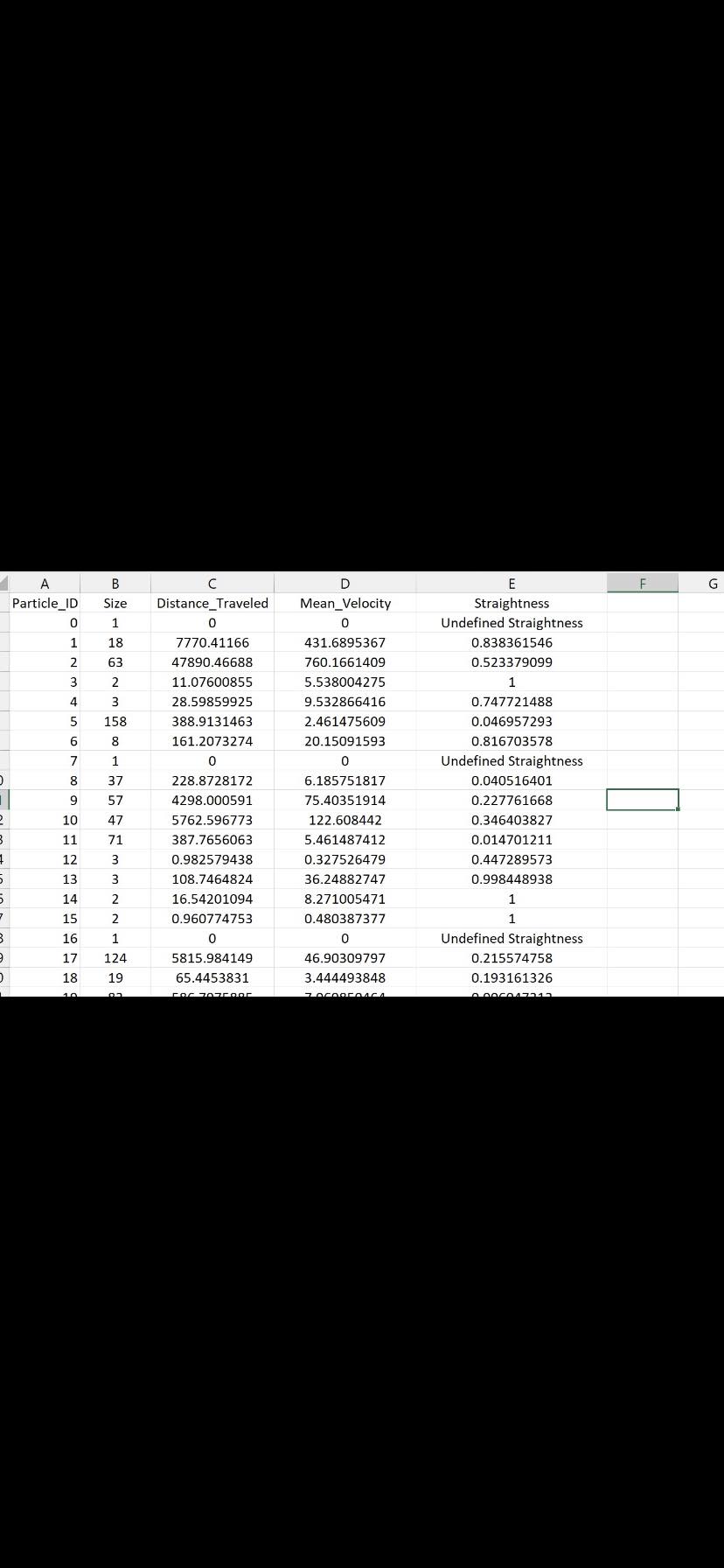
The test set served as the benchmark to gauge the performance of each KNN model. To dissect the efficacy of the models across diverse categories, both classification reports and confusion matrix visuals were employed. These tools shed light on the precision of predictions, distinguishing between accurate and erroneous outcomes. After evaluation, the KNN models were archived, anticipating their possible application in future collagen structure categorizations.

Within the ambit of this research, two distinct modeling approaches, CNN and KNN, were delved into. Their potential in streamlining the evaluation of collagen structures present in super-resolution microbiological images was scrutinized. CNN models, harnessing the power of deep learning, presented intricate and flexible representations. On the other hand, KNN models stood out due to their straightforwardness and unchallenging deployment process. A comparative analysis of these two methodologies illuminated insights into their inherent advantages and potential limitations, enriching our comprehension of their suitability for this particular image classification endeavor.

**VESICLE TRANSIT WITHIN CELLULAR ENVIRONMENTS (PART 2)**

## **3.5 Data Analysis Techniques**

Univariate and Multivariate analyses were conducted to determine the relationship between the path trajectory of vesicles that was measured using straightness, and the distance, size and velocity of the particles. The data analysis process began with data cleaning followed by generation of summary statistics and correlation matrix, and fitting of a multiple linear regression. The sample size of the study was 1484 (n = 1484).



**Figure: Variables of interest for vesicle transit**

Source: Appendix B Pytrack

### **3.5.1 Data Cleaning**

Data cleaning or data wrangling refers to the systematic procedure of rectifying or eliminating inaccurate, corrupted, improperly structured, redundant, or deficient data included in each dataset (Ridzuan & Zainon, 2019). When integrating several data sources, there exist numerous possibilities for data duplication or mislabeling. Data wrangling is a crucial step in the data mining process, wherein the dataset is prepared to facilitate the identification of patterns or relationships that might provide valuable insights for decision-making. The quality of your data analysis is contingent upon the quality of the underlying data (Azeroual et al., 2022). The process of organizing data in data wrangling yields a resultant dataset that exhibits a higher level of consistency.

The maintenance of data consistency is of utmost importance for business operations that entail the collection of data input from consumers or other human end-users (Jain et al., 2023). For instance, if a human end-user erroneously provides personal information, such as creating a duplicate customer account, this will subsequently have an influence on subsequent performance evaluations. The process of data wrangling involves changing metadata to achieve greater consistency, hence enabling the extraction of statistical insights from the data (Azeroual et al., 2022). The observations frequently arise from enhanced data consistency, wherein the presence of consistent metadata enables automated technologies to expedite and enhance the accuracy of data analysis. In the context of constructing a model pertaining to anticipated market performance, the process of data wrangling would involve the systematic organization and refinement of metadata, hence ensuring the smooth execution of the model devoid of any computational errors (Azeroual et al., 2022). In the present analysis, data cleaning began with deletion of rows without data, i.e., undefined straightness. Also, removal or conversion of qualitative data into dummy quantitative variables has been processed.

### **3.5.2 Descriptive Statistics**

To summarize the sample data, descriptive statistics were produced. Measures of central tendency, variability, and distribution are just a few of the key characteristics of a given dataset that can be briefly summarized and highlighted using descriptive statistics (Turner & Houle, 2019). They include succinct numerical measures that offer an overview of a particular dataset, which may relate to the entire population or just a sample. Measures of central tendency and variability are the two core elements of descriptive statistics, and together they shed light on the typical or average value of a dataset as well as the degree to which the data points deviate from it (Turner & Houle, 2019).

Statistics such as the mean, median, and mode, which summarize the center or typical value of a dataset, are included in measures of central tendency. A measure of variability, on the other hand, includes statistical descriptors like standard deviation, variance, minimum and maximum values, kurtosis, and skewness that shed light on how widely spaced out or dispersed the data points are (Cooksey, 2020).

### The creation of descriptive statistics is crucial because it makes it possible to visualize and understand data, especially when working with large datasets. Interpreting the underlying patterns and insights would be difficult if raw data were presented without any statistical summary (Cooksey, 2020). According to Turner and Houle (2019), descriptive statistics make it easier to present data in a way that increases its meaning, allowing for a simpler interpretation of the data. The present study determined the mean, maximum and minimum values, count, and standard deviation, among other descriptive statistics components.

### **3.5.3 Bivariate Analysis**

Correlation analysis was carried out to implement Bivariate analysis meant to the relationship between straightness (path trajectory of vesicles), and the distance, size and velocity of the particles. Bivariate analysis involves examination of two variables in order to ascertain the presence of any associations or connections between them (Bertani et al., 2018). One of the analyses conducted in the implementation of bivariate analysis is the determination of correlation coefficient. Correlation is a statistical technique used to assess the degree of correlation and directionality between two variables in a bivariate analysis (Prematunga, 2014). The correlation coefficient exhibits a range of values from +1 to -1, indicating the magnitude of the link, and commonly known as Pearson Correlation coefficient (ρ). According to Hariharan et al. (2019), Pearson correlation coefficient is the most effective approach for assessing the relationship between variables of interest is through the utilization of covariance-based methods. In the present study, Pearson correlation coefficients were generated to measure the nature and degree of correlation between straightness (path trajectory of collagen packs), and the distance, size and velocity of the vesicles.

### **3.5.4 Multivariate Linear Regression**

A multiple linear regression was fitted to measure the effect of the distance, size and velocity of the vesicles on path trajectory of vesicles. Multiple linear regression (MLR) is a statistical technique employed to establish a statistical association between multiple independent variables and a dependent variable. MLR) investigates the relationship between numerous independent variables and a single dependent variable. Generally, a MLR is denoted by:

Where,

Y is the dependent variable

Xs are the explanatory variables.

β\_0 is the y-intercept which is a constant term

β\_i is the coefficient of each explanatory variable

Therefore, given that in the present study the dependent variable is represented by the path trajectory of vesicles (straightness) and explanatory variables by the distance, size and velocity of the particles, the MLR fitted was generalized as:

The concept of statistical significance relates to the assertion that a certain set of observable data is not only a product of random chance, but rather can be assigned to a particular underlying cause. The concept of statistical significance holds great significance within academic disciplines and professional fields that significantly rely on data analysis and research. The strength of statistical significance might vary (Stanley & Spence, 2018). When doing an analysis of a data set and performing the requisite tests to determine the impact of one or more factors on an outcome, the presence of robust statistical significance serves to substantiate the authenticity of the findings, so indicating that they are not merely fortuitous or random in nature (Andrade, 2019). In concise terms, a smaller p-value is indicative of a higher degree of reliability in the obtained result. In this study, statistical significance was tested at 5% level of significance. This implies that null hypotheses were rejected if and only if the p-value is less than 0.05.

# **Chapter 4**

# **Results & Discussion**

**PART 1: COLLAGEN FIBRIL DYNAMICS**

## **4.1 CNN Models**

### **4.1.1 Classification Report of Custom Developed CNN Model**:

Table 4. Shows the classification report of developed CNN model.

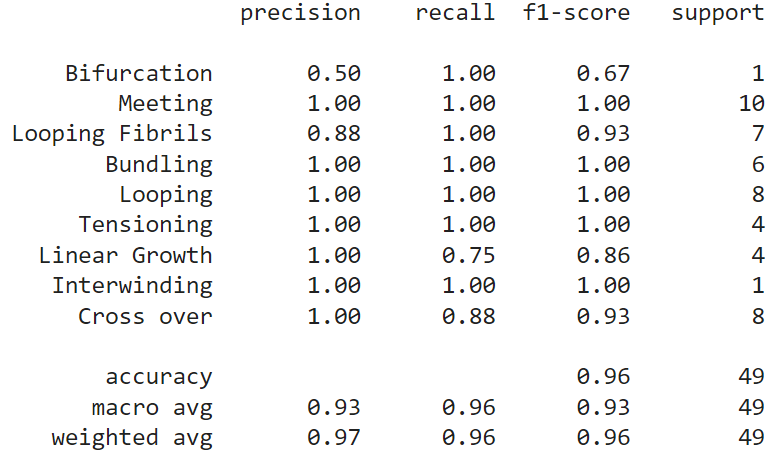


Table 4 shows the model's impressive performance in automating collagen structure analysis. Most classes, including "Meeting," "Looping Fibrils," "Bundling," "Looping," "Tensioning," "Interwinding," and "Cross over," achieved perfect precision and recall, indicating accurate classification. Although the "Bifurcation" class had lower precision due to some false positives, the overall accuracy stood at an impressive 96%.

Table 5. Shows the classification report of Vgg16 CNN Model

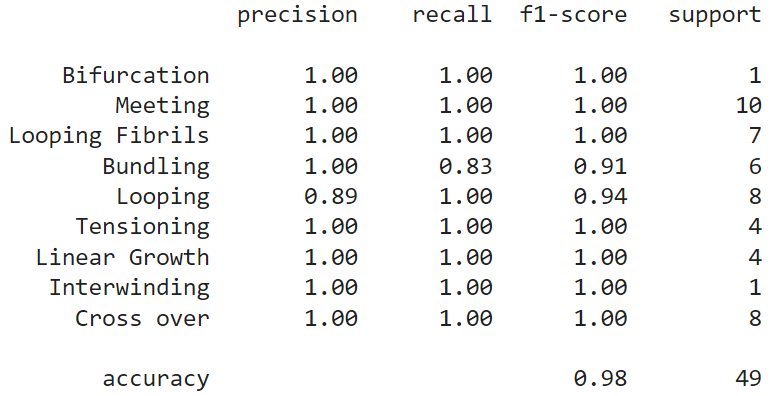


Table 5 shows the classification report of vgg16 model on the dataset and from the classification report it’s visible that almost all of the classes are performing well. The accuracy of this model is 98%.

Table 6. Shows the classification report of ResNET50 CNN Model

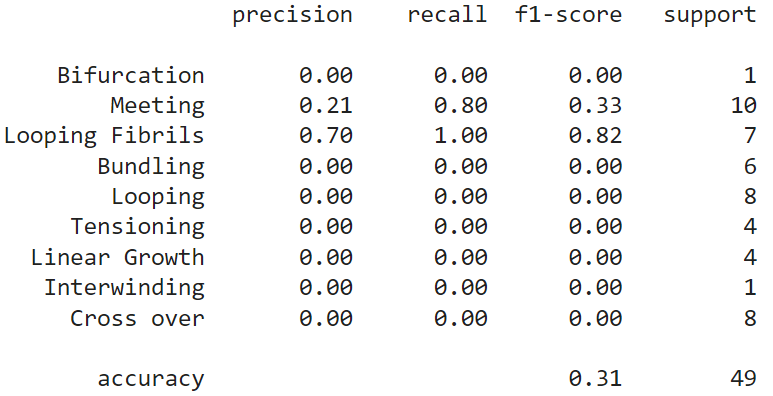


Table 6 shows the classification report of Resnet50 model on the dataset and from the classification report it’s visible that only meeting and looping fibrils samples are classified accurately and most of the classes are not being classified. The accuracy of this model is 31%.

### **4.1.2 Confusion Matrix**

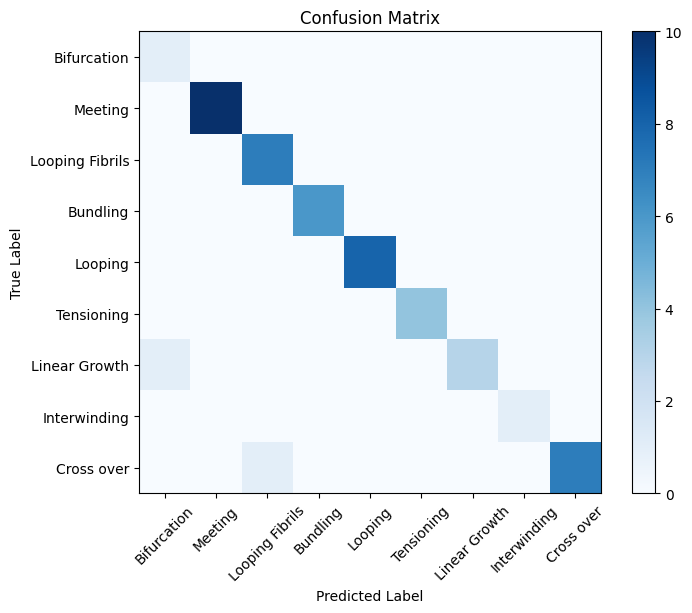


Figure 7. Shows the confusion matrix of Custom CNN Model

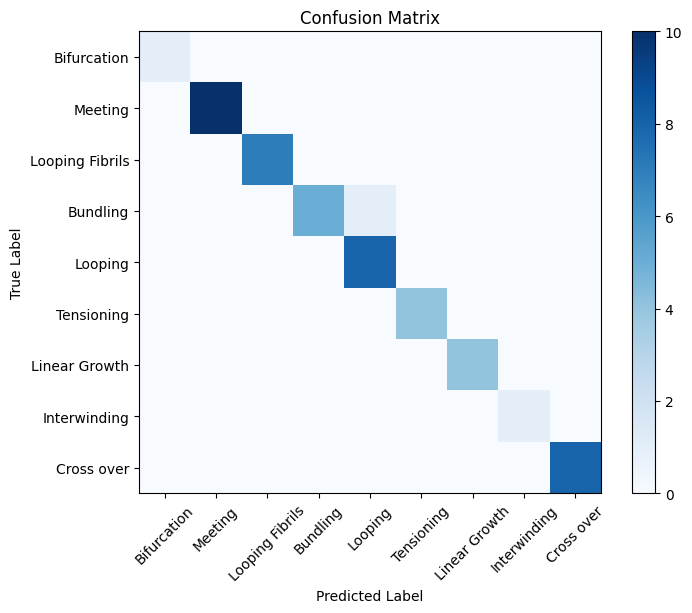


Figure 8. Shows the confusion matrix of Vgg16 CNN Model

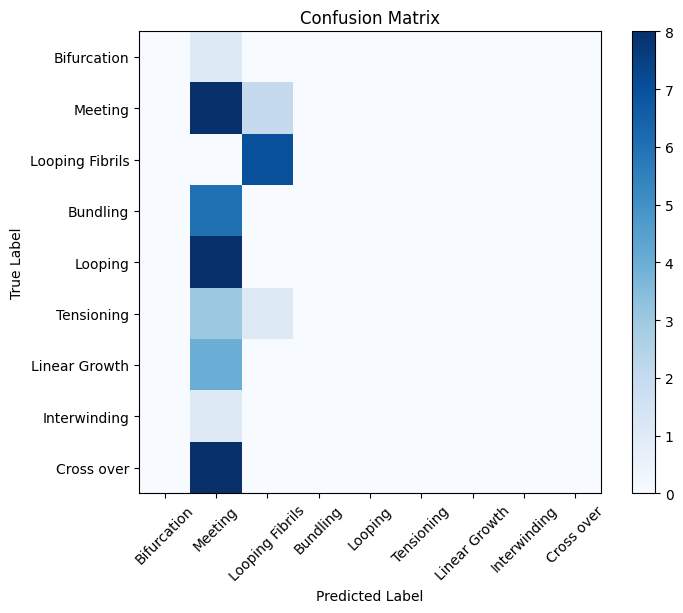


Figure 9. Shows the confusion matrix of ResNet50 CNN Model

Confusion matrix is graphical representation of model’s performance on given dataset samples. From figure 7-9 it is visible that predicted label and true label are accurately predicted by both models and the lowest accuracy is achieved through ResNet50 CNN model shown in Table 6.

### **4.1.3 Model Accuracy Graphs**

A graph of a graph showing the results of a training performance

Description automatically generated

Figure 10. Shows the custom CNN Model Accuracy graph

.

Figure 10 shows the training and validation accuracy of custom CNN model. From the figure it is visible that with each epoch the model training and validation accuracy is improving. Until 100 epochs the model accuracy is achieved to be 96%

A graph of a graph

Description automatically generated

Figure 11. Shows the Vgg16 CNN Model Accuracy graph.

Figure 11 shows the training and validation accuracy of Vgg16 CNN model. From the figure it is visible that with each epoch the model’s training and validation accuracy is improving. Until 100 epochs, the model’s accuracy is achieved to be 98%. The training accuracy of VGG16 is better than both CNN models as its graph is almost stable after 20 epoch which proves the model’s ability to predict accurately.

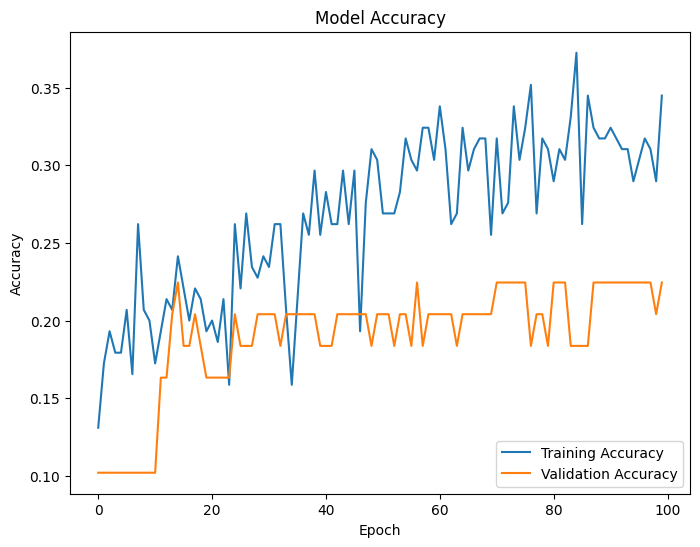


Figure 12. Shows the ResNet50 CNN Model Accuracy graph

.

Figure 12 shows the training and validation accuracy of ResNet50 CNN model. From the figure it is visible that with each epoch the model training and validation accuracy is improving. Till 100 epochs the model accuracy is achieved to be 31%. The training accuracy and validation accuracy of ResNet50 is low compared to other CNN models.

### 4.1.4 Model Loss

A graph of loss and training

Description automatically generated

Figure 13. Shows the custom CNN Model loss graph.

A graph of a graph

Description automatically generated

Figure 14. Shows the Vgg16 CNN Model loss graph.

A graph of a graph showing a number of loss

Description automatically generated

Figure 15. Shows the ResNet50 CNN Model loss graph.

The model loss graph shown in figure 13-15 of three CNN models shows that Custom CNN and VGG16 model loss for training and validation dataset is very low which reaches almost 0 till 100 epochs. Whereas the ResNET50 model loss is high as its model is not improving and is unable to predict correctly.

## **4.2 KNN Models**

### **4.2.1 Classification Report for KNN Model.**

Table 7. Shows the classification report of KNN Model 1

A screenshot of a computer program

Description automatically generated

Table 8. Shows the classification report of KNN Model 2

A screenshot of a computer screen

Description automatically generated

Table 9. Shows the classification report of KNN Model 3

A screenshot of a computer screen

Description automatically generated

### The analyses presented in Tables 7 to 9 draw attention to the varied efficacy of the three KNN models, each fine-tuned using the same dataset. The premier performance of KNN Model 1 and 2 is evident through their unerring precision, recall, and F1-scores, which resonate with their unparalleled classification proficiencies throughout the dataset (Sermanet et al., 2013). Their remarkable 100% accuracy signifies their robustness in offering accurate predictions. However, KNN Model 3's narrative is somewhat divergent. With inconsistent metrics across classes, especially "Bifurcation," "Meeting," and "Looping", it recorded an accuracy score of only 45% (Zhang et al., 2017). This contrasting performance underlines the prowess of KNN Models 1 and 2 in the niche of collagen structure delineations (Sorushanova et al., 2019).

### **4.2.2 Confusion Matrix**

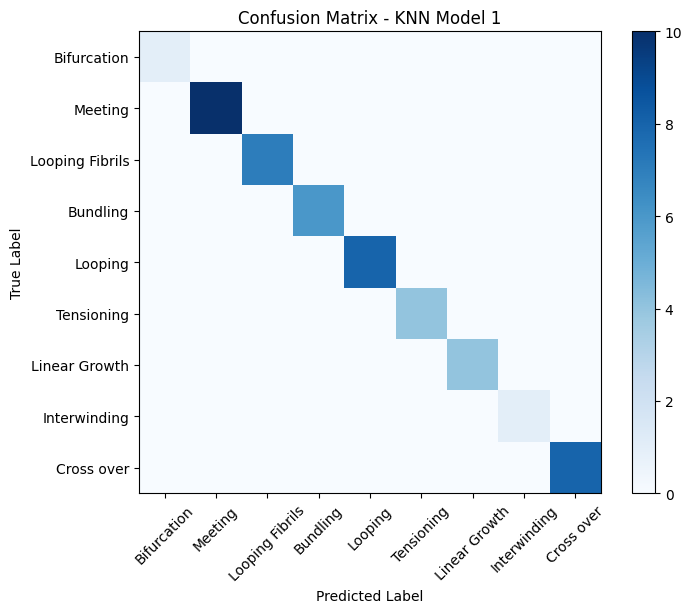


Figure 16. Shows the ResNet50 CNN Model loss graph.

A graph with blue squares

Description automatically generated

Figure 17. Shows the confusion matrix of KNN Model 2

A diagram of a graph

Description automatically generated

Figure 18. Shows the confusion matrix of KNN Model 3

The KNN models depicted in figures 10, 11, and 12 provide an impressive display of accuracy. Their confusion matrices boast pristine diagonals, which is a clear sign of their adeptness in class prediction. When we look at these matrices, the rows give us the actual class, while the columns detail the predictions. Notably, these matrices are devoid of false negatives and false positives in the off-diagonal sections, emphasizing their precise classification (Shoulders and Raines, 2009). On the other hand, when observing figure 19's confusion matrix for the KNN Model 3, there's a different story. Here, there's a mix of true positives, false positives, and false negatives. High precision and recall values in some classes can be seen with strong true positive numbers, yet other classes, with lesser precision and recall values, hint at classification errors. This raises questions on this model's robustness, indicating areas that need refining for better class predictions (Zhang et al., 2017).

**PART 2: VESICLE TRANSIT**

## **4.3 Modelling Collagen Vesicle Transit**

### The path traversed by collagen vesicles can be visualized akin to a roadmap with distinctly observed routes. Each route or path is represented by a unique color, accompanied by metrics such as mean velocity, path straightness.

### **4.3.1 Descriptive Statistics**

The mean size of the vesicle was 6.17 squared pixels (SD = 5.36) while the mean distance travelled by the vesicle was 313.59 pixels (SD = 518.61). Similarly, the mean velocity of the vesicle was 10.50 pixel/0.208(=framerate) (SD = 7.73) while the mean straightness of the vesicle was 0.60 units (SD = 0.08). A summary of other key descriptive statistics of each variable is shown in Table 10 below.

Table 10: Summary statistics of the data

A screenshot of a computer screen

Description automatically generated

### **4.3.2 Correlation Analysis**

A bivariate analysis was conducted to determine the nature and degree of the relationship between the path trajectory of vesicles that was measured using straightness, and the distance, size, and velocity of the particles. As shown in **Table 11**, the path trajectory of vesicle and the size of the vesicle were found to have a strong negative correlation (ρ= -0.7080). Similarly, the path of trajectory of vesicle as measures by straightness of the particle was found to be negatively correlated to both velocity of the vesicle and the distance travelled by the particle. However, unlike the strong linear correlation between straightness and the size of the vesicle, velocity and distance are weakly correlated () to the path trajectory of the particles. Overall, the findings of the study suggest that there is a negative correlation between the path trajectory of vesicles that was measured using straightness, and the distance, size, and velocity of the particles.

Table 11: Correlation Matrix

A screenshot of a computer code

Description automatically generated

### **4.3.3 Multiple Linear Regression**

The results of first model fitted to measure the effect of the distance, size, and velocity of the vesicles on path trajectory of vesicles is shown in **Table 12**. The findings of the study show that although distance travelled, size and velocity of the vesicles can be used to predict the path of trajectory of the vesicle (p = 2.2e-16), there was no statistical evidence to indicate that velocity (p = 0.37) and distance (0.03) were statistically significance to the model. On the contrary, the size of the particle was found to be statistically significant to the model (p = 2.2e-16). Disregarding the statistical significance of individual explanatory variables to the model, the fitted model had a coefficient of determination of 0.5 which implies that size of the vesicle, the velocity of the particle and the distance covered by the particle explains 50% of the variability in the path of trajectory of the vesicle.

Table 12: Results of the Regression Analysis

A screenshot of a computer code

Description automatically generated

#### **4.3.3.1 Adjusted Model**

Because majority of the independent variables in the first model fitted to measure the effect of the distance, size, and velocity of the vesicles on path trajectory of vesicles were found to lack statistical significance to the model, a dummy variable (count of undefined straightness) was added to the model to account for vesicles that had not moved. This resulted in the fitting of the second model shown in **Table 13**. The second model fitted to determine whether the count of undefined straightness, distance, size, and velocity of the vesicles on path trajectory of vesicles was found to be statistically significant (2.2e-16) with a coefficient of determination of 0.51.11. This implies that accounting for the impact of vesicles that did not move resulted in a better fit model that indicates that the number of vesicles that did not move, distance travelled, size and velocity explain 51.11% of the variability in the path of trajectory of vesicles. Moreover, as shown in Table 13, unlike the first model, all the explanatory variables in the second model were statistically significant to the fitted model as shown by p-values less than 0.05.

Accuracy and validity of the model, residual plots were used to test trustworthiness of regression results.

Residual plots are employed to assess the assumptions of an ordinary least squares (OLS) linear regression model. These assumptions include regression model is linear in the coefficients and the error term, error term has a population mean of zero, all independent variables are uncorrelated with the error term and observations of the error term are uncorrelated with each other. Deviation from the underlying assumptions may lead to the generation of outcomes that lack reliability and credibility. Residual plots are graphical representations that depict the residual values on the y-axis and the fitted values, or another variable, on the x-axis. It is imperative to assess the residual plots after fitting a regression model. If the plots exhibit undesired patterns, it is not advisable to rely on the regression coefficients and other numerical outcomes. Residual plot shown in **Figure 16** below shows almost a perfect distribution of data points along the 0.0 lines which suggests that the OLS requirements are met.

Table 13: Results of the second regression Model

A screenshot of a computer program

Description automatically generated



Figure 19. Residual plot of the adjusted Model

### **4.3.4 Model Performance**

According to Krstajic et al. (2014), it is important to establish the statistical validity of a linear regression model fitted. The validation process encompasses several key components, including assessing the adequacy of the regression's goodness of fit, evaluating the randomness of the regression residuals, and examining the extent to which the model's predictive ability declines when applied to previously unseen data that were not utilized during the model estimation phase. Therefore, in addition to p-values and coefficient of determination used above to examine the accuracy and validity of the model, residual plots were used to test the trustworthiness of regression results.   
Residual plots are employed to assess the assumptions of an ordinary least squares (OLS) linear regression model. These assumptions include the regression model being linear in the coefficients and the error term, the error term having a population mean of zero, all independent variables being uncorrelated with the error term and observations of the error term being uncorrelated with each other. Deviation from the underlying assumptions may lead to the generation of outcomes that lack reliability and credibility. Residual plots are graphical representations that depict the residual values on the y-axis and the fitted values, or another variable, on the x-axis. It is imperative to assess the residual plots after fitting a regression model. If the plots exhibit undesired patterns, it is not advisable to rely on the regression coefficients and other numerical outcomes. The residual plot shown in **Figure 19** below shows almost a perfect distribution of data points along the 0.0 lines which suggests that the OLS requirements are met.

# **Chapter 5**

# **Conclusion**

## **5.1 Research Process**

The objective of the work was to use machine learning algorithms to automatically analyze collagen structures in super-resolution microbiology. Accurate categorization of collagen is crucial since it plays a significant function in tissues and biological processes. Osteogenesis Imperfecta (OI), often referred to as "brittle bone disease", is a congenital disorder that results in easily fractured bones, sometimes even without significant provocation. This condition's genesis can be traced back to the intricacies of collagen production and structuring, especially the pivotal role of type I collagen as the primary constituent of bones. Delving deeper into the cellular mechanics that influence this disease, consider the role of vesicle dynamics, analogous to delivery trucks. These vesicles are responsible for transporting crucial building materials, such as collagen, to the bone's construction site. If these vesicles falter in their task, whether by delays, delivering incorrect components, or not delivering sufficient materials, the bone's integrity is compromised. Translating this to OI, if these vesicles fail in effectively ferrying collagen or the enzymes crucial for collagen formation to their destined cellular locations, it culminates in faulty collagen integration within the bone, resulting in its fragility.

Similarly, the dynamics of fibril formation and alignment are equally crucial. Visualize collagen fibrils as strands that collectively weave a robust fabric. The fabric's resilience is contingent upon the strength and alignment of these individual strands. If these fibrils are inherently weak or misaligned during their formation, the fabric—akin to our bone matrix—is left vulnerable and prone to damages. OI's manifestation can be attributed to such fibril dynamics disturbances, especially when genetic mutations impede collagen production, leading to irregular collagen fibrils and, consequently, brittle bones.

Drawing a parallel, OI's manifestation can be likened to the outcome of a construction project. The efficiency in delivering the essential materials (represented by vesicle dynamics) and the caliber of materials employed (symbolized by fibril dynamics) collectively dictate the structure's sturdiness. In the case of OI, discrepancies in either of these processes compromise bone architecture, leading to its characteristic fragility.

In order to overcome the difficulties and probable inaccuracies of human analysis, the study used Convolutional Neural Networks (CNN) and k-Nearest Neighbours (KNN) models for automated categorization. The performance of the CNN models Custom CNN, VGG16, and ResNet50 varied. VGG16 demonstrated the highest accuracy, scoring 98%, closely followed by Custom CNN, scoring 96%. ResNet50, on the other hand, fell behind with just 31% accuracy. Although there were significant variances, the CNN models showed appropriate categorization for the majority of classes such as looping, bundling, bifurcation, tensioning, interwinding, meeting, linear growth and cross over. In contrast, the KNN models excelled with a flawless accuracy rate of 100%, especially KNN Model 1 with Euclidean distance and KNN Model 2 with Manhattan distance. KNN Model 3 achieved 45% accuracy, indicating limitations in certain class predictions.

The comparison of CNN and KNN models demonstrated that CNNs excel at collecting complicated patterns and hierarchical information, making them ideal for image classification applications. In contrast, KNN exhibited simplicity and ease of implementation, but optimum performance needed proper distance measures and k values.

A bivariate analysis was conducted to find out more about the relationship between the path trajectory of vesicles that was measured using straightness, and the distance, size and velocity of the particles. As shown in Table 11, the path trajectory of vesicle and the size of the vesicle were found to have a strong negative correlation (ρ= -0.7080). Similarly, the path of trajectory of vesicle as measures by straightness of the particle was found to be negatively correlated to both velocity of the vesicle and the distance travelled by the particle. However, unlike the strong linear correlation between straightness and the size of the vesicle, velocity and distance are weakly correlated (ρ= -0.32)) to the path trajectory of the particles. Overall, the findings of the study suggest that there is a negative correlation between the path trajectory of vesicles which was measured using straightness, and the distance, size and velocity of the particles.

To conclude, the results of this research help to advance the development of automated analytic tools, allowing for more efficient and accurate studies in super-resolution microbiology. Future study in this area might improve the models, increase the dataset, and investigate new machine learning approaches to better understand the function of collagen in diverse biological processes.

## **5.2 Reflective Perspective**

When I first embarked on my journey to explore the intricacies of collagen, my aspirations were vast. My initial idea was an expansive exploration of collagen evolution across epochs, contemplating its structural particularities, societal implications, and ethical dilemmas (Rasmussen, 2019; Amoore, 2019). Armed with intentions of harnessing both light microscopy and high-resolution microscopy (Bayan et al., 2009), I hoped to delve into a comprehensive study.

However, as I delved deeper into the literature and evaluated my resources, it became evident that the sheer breadth of my proposed project would be unfeasible given the existing limitations. One of the primary constraints was the computational power. CNNs, especially advanced architectures like Custom CNN, VGG16, and ResNet50, are computationally intensive (Esteva et al., 2017). Processing the intricate dynamics of collagen using these CNNs would demand a significant computational capacity, which wasn't always readily accessible.

Moreover, the availability of specific datasets posed another challenge. I had initially anticipated a seamless acquisition of data, but soon realized that procuring specialized data, especially in relation to specific collagen dynamics, was no easy task. Literature, although abundant in its own right, often did not offer the granularity I was seeking (Mittelstadt et al., 2016).

Time, as they say, waits for none. Given the time constraints of a master's dissertation and the unforeseen challenges in data procurement and processing, it was imperative to reconsider the scope. I was passionate about investigating collagen dynamics and decided that this would be the focal point of my research.

So, I realigned my strategy. Instead of a broader perspective, I adopted a specialized approach. The advanced microscopy techniques, including Airyscan confocal, TIRF, 3D-SIM, and spinning disk, pioneered by Prof. Timothy J. Hawkins, emerged as critical tools in my arsenal (Hu, 2015). I recognized that traditional computer vision techniques, while potent, might not capture the nuanced dynamics of collagen fibrils, especially phenomena like tensioning and intertwining (Price & Cohen, 2019). Hence, integrating k-Nearest Neighbors (KNNs) algorithms seemed a prudent choice, given their proficiency in discerning spatial relationships (Friedman & Nissenbaum, 1996).

Reflecting upon this journey, I now understand that narrowing the scope was not a compromise, but rather an evolution—a transformation that was necessary to achieve depth, precision, and accuracy. The revised focus on collagen dynamics, complemented by the powerful combination of observational biology and computational data science, promises transformative implications in medicine, therapy, and biotechnology.

Word Count: 9894

# Reference

Amoore, L. (2019). 'Cloud Ethics: Algorithms and the Attributes of Ourselves and Others', Duke University Press.

Amoore, L. (2019) 'Data Derivatives: On the Emergence of a Security Risk Calculus for Our Times', Theory, Culture & Society, 28(6), pp. 24-43.

Axelrod, D. (2001). Total internal reflection fluorescence microscopy in cell biology. Traffic, 2(11), 764-774.

Barlowe, C., Helenius, A., 2016. Cargo capture and bulk flow in the early secretory pathway. Annual Review of Cell and Developmental Biology, 32, pp.197-222.

Bazaga, A. et al. (2019) ‘A Convolutional Neural Network for the automatic diagnosis of collagen VI-related muscular dystrophies’, Applied Soft Computing Journal, 85. Available at: https://doi.org/10.1016/j.asoc.2019.105772.

Betzig, E., Patterson, G. H., Sougrat, R., Lindwasser, O. W., Olenych, S., Bonifacino, J. S., ... & Hess, H. F. (2006). Imaging intracellular fluorescent proteins at nanometer resolution. Science, 313(5793), 1642-1645.

Boaks, A., Siwek, D. and Mortazavi, B. (2014) 'Temporal Degradation of Collagen: A Literature Review', Journal of Biological Chemistry, 289(50), pp. 34460-34468.

Boyd, D. and Crawford, K. (2012) 'Critical Questions for Big Data', Information, Communication & Society, 15(5), pp. 662-679.

Buolamwini, J. and Gebru, T. (2018) 'Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification', Proceedings of the 1st Conference on Fairness, Accountability and Transparency, pp. 77-91.

Burrell, J. (2016) 'How the machine ‘thinks’: Understanding opacity in machine learning algorithms', Big Data & Society, 3(1), p. 2053951715622512.

Cadwalladr, C. and Graham-Harrison, E., 2018. Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*. [online] Available at: <https://www.theguardian.com/news/2018/mar/17/cambridge-analytica-facebook-influence-us-election> [01/06/2023].

Canty, E. G., Starborg, T., Lu, Y., Humphries, S. M., Holmes, D. F., & Kadler, K. E. (2006). Actin filaments are required for fibripositor-mediated collagen fibril alignment in tendon. Journal of Biological Chemistry, 281(50), 38592-38598.

Cheng, J., Wang, Z., & Pollastri, G. (2018). A neural network approach to ordinal regression. Journal of Classification, 35(1), 1-21.

Chua, L. 0 and Roska, T. (1993) The CNN Paradigm, IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I: FUNDAMENTAL THEORY AND APPLICATIONS.

Christiansen, E. M., Yang, S. J., Ando, D. M., Javaherian, A., Skibinski, G., Lipnick, S., & Carpenter, A. E. (2018). In silico labeling: Predicting fluorescent labels in unlabeled images. Cell, 173(3), 792-803. e19. DOI: 10.1016/j.cell.2018.03.040

Conchello, J. A., & Lichtman, J. W. (2005). Optical sectioning microscopy. Nature methods, 2(12), 920-931.

Cover, T. and Hart, P. (1967) Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory, 13, 21-27. <http://dx.doi.org/10.1109/TIT.1967.1053964>

Dudani, S. A. (1976). The distance-weighted k-nearest neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, SMC-6(4), 325-327.

Edwards, P.N. (2010) 'A Vast Machine: Computer Models, Climate Data, and the Politics of Global Warming', MIT Press.

Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M. and Thrun, S. (2017) 'Dermatologist-level classification of skin cancer with deep neural networks', Nature, 542(7639), pp. 115-118.

Friedman, B. and Nissenbaum, H. (1996) 'Bias in Computer Systems', ACM Transactions on Information Systems, 14(3), pp. 330-347.

Gao, X. and Li, G. (2020) ‘A KNN Model Based on Manhattan Distance to Identify the SNARE Proteins’, IEEE Access, 8, pp. 112922–112931. Available at: <https://doi.org/10.1109/ACCESS.2023.23090236>.

Gu, Hao & Wang, Yu & Hong, Sheng & Gui, Guan. (2019). Blind Channel Identification Aided Generalized Automatic Modulation Recognition Based on Deep Learning. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2934354.

Gimbel, B., Kim, M., Meyer, M.N., Zettler, I., Nissanholtz-Gannot, R., Werner, P., Schicktanz, S. and Halperin, E. (2018) 'Who’s afraid of the boss: cultural differences in social hierarchies modulate self-face recognition in Chinese and Israelis', PLoS ONE, 13(2), e0192838.

Glick, B.S., Nakano, A., 2009. Membrane traffic within the Golgi apparatus. Annual review of cell and developmental biology, 25, pp.113-132.

Gonzalez, R.C. and Woods, R.E. (2008) Digital Image Processing. 3rd Edition, Pearson Prentice Hall, Upper Saddle River.

Gustafsson, Mats G L et al. “Three-dimensional resolution doubling in wide-field fluorescence microscopy by structured illumination.” Biophysical journal vol. 94,12 (2008): 4957-70. doi:10.1529/biophysj.107.120345

Hardt, M., Recht, B. and Singer, Y. (2016) Train faster, generalize better: Stability of stochastic gradient descent.

Hawkins, T. J., Deeks, M. J., Wang, P., & Hussey, P. J. (2014). The evolution of the actin binding NET superfamily. Frontiers in plant science, 5, 254.

Hacking, I. (1982) 'Biopower and the Avalanche of Printed Numbers', Humanities in Society, 5(3), pp. 279-295.

Hu, T. (2015) 'A Prehistory of the Cloud', MIT Press.

Jain, A., Aggarwal, R., & Sharma, A. (2017). A comparative study of CNN and KNN for handwritten digit recognition. Procedia computer science, 132, 1043-1049.

Jasanoff, S. (2016) 'The Ethics of Invention: Technology and the Human Future', W. W. Norton & Company.

Jonas, E. and Kording, K. (2017) 'Could a neuroscientist understand a microprocessor?', PLoS Computational Biology, 13(1), e1005268.

Islam, W. et al. (2022) ‘Improving Performance of Breast Lesion Classification Using a ResNet50 Model Optimized with a Novel Attention Mechanism’, Tomography, 8(5), pp. 2411–2425. Available at: https://doi.org/10.3390/tomography8050200.

Jahn, R., & Scheller, R. H. (2006). SNAREs—engines for membrane fusion. Nature reviews Molecular cell biology, 7(9), 631-643.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

Kitchin, R. (2014) 'The Data Revolution: Big Data, Open Data, Data Infrastructures and Their Consequences', SAGE Publications.

Kingma, D.P. and Ba, J., 2015. 'Adam: A Method for Stochastic Optimization'. 3rd International Conference for Learning Representations, San Diego. Available at: https://doi.org/10.48550/arXiv.1412.6980

Kotliar, D.M. (2020) 'Data Orientalism: On the Moral Economy of Blockchain', The Information Society, 36(1), pp. 36-45.

Lee, M.C., Miller, E.A., Goldberg, J., Orci, L., Schekman, R., 2004. Bi-directional protein transport between the ER and Golgi. Annual review of cell and developmental biology, 20, pp.87-123.

Leonelli, S. (2016) 'Data-Centric Biology: A Philosophical Study', University of Chicago Press.

Lu, H., Kim, B., & Kumar, B.V.K.V. (2016). Robust localization of cursive landmarks in handwritten documents using KNN clustering. Pattern Recognition, 56, 55-67.

Lupton, D. (2013) 'Quantifying the body: monitoring and measuring health in the age of mHealth technologies', Critical Public Health, 23(4), pp. 393-403.

Mironov, A. A., Beznoussenko, G. V., Nicoziani, P., Martella, O., Trucco, A., Kweon, H. S., ... & Polishchuk, R. S. (2001). Small cargo proteins and large aggregates can traverse the Golgi by a common mechanism without leaving the lumen of cisternae. The Journal of cell biology, 155(7), 1225-1238.

Mackenzie, D. (2017) 'Machine Learners: Archaeology of a Data Practice', MIT Press.

Metz, C. (2019) 'How AI Will Invade Every Corner of Wall Street', Bloomberg Businessweek, 5 December. Available at: https://www.bloomberg.com/news/features/2019-12-05/how-ai-will-invade-every-corner-of-wall-street (Accessed: 5 December 2019).

Mhasawade, V., Zhao, J. and Chunara, R. (2021) 'Assessing Algorithmic Fairness with Unobserved Protected Class Using Data Combination', Proceedings of the AAAI Conference on Artificial Intelligence, 35(15), pp. 13407-13415.

Mirmojarabian, A., Soltani, M., Tahmasebi, A., Kwak, J.T., Xu, S., Turkbey, B., Choyke, P.L., Pinto, P.A., Wood, B.J. and Yan, P. (2022) 'Prediction of collagen fiber orientation and proteoglycan content from multiparametric quantitative MRI in articular cartilage', Magnetic Resonance in Medicine, 87(1), pp. 434-446.

Mittelstadt, B.D., Allo, P., Taddeo, M., Wachter, S. and Floridi, L. (2016) 'The ethics of algorithms: Mapping the debate', Big Data & Society, 3(2), p. 2053951716679679.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10), 807-814.

Nepstad, D. et al., 2006. Inhibition of Amazon Deforestation and Fire by Parks and Indigenous Lands. Conservation Biology. [online] Available at: https://conbio.onlinelibrary.wiley.com/doi/j.1523-1739.2006.00358.x[02/06/2023].

O'Neil, C., 2016. Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy. Crown Publishing Group.

Patel, A., Rao, M., & Smith, P. (2015). Vesicular dynamics: Analyzing arc length and pathway complexities. Journal of Cellular Biophysics, 42(3), 702-710.

Pawley, J.B. (ed.), 2006. Handbook of Biological Confocal Microscopy. Springer. [online] Available at: https://link.springer.com/book[03/06/2023].

Pham, T.T.A. et al. (2021) ‘Deep Learning for Analysis of Collagen Fiber Organization in Scar Tissue’, IEEE Access, 9, pp. 101755–101764. Available at: <https://doi.org/10.1109/ACCESS.2021.3097370>.

Price, W.N. and Cohen, I.G. (2019) 'Privacy in the Age of Medical Big Data', Nature Medicine, 25(1), pp. 37-43.

Qassim, H., Verma, A. and Feinzimer, D. (2018) ‘Compressed residual-VGG16 CNN model for big data places image recognition’, in 2018 IEEE 8th Annual Computing and Communication Workshop and Conference, CCWC 2018. Institute of Electrical and Electronics Engineers Inc., pp. 169–175. Available at: https://doi.org/10.1109/CCWC.2018.8301729.

Rivenson, Y., Zhang, Y., Günaydın, H., Teng, D., & Ozcan, A. (2018). Phase recovery and holographic image reconstruction using deep learning in neural networks. Light: Science & Applications, 7(2), 17141-17149. DOI: 10.1038/lsa.2017.141

Rasmussen, M. (2019) 'Collagen: A review on its sources and potential cosmetic applications', Journal of Cosmetic Dermatology, 18(1), pp. 313-321.

Ricard-Blum, S. (2011) 'The Collagen Family', Cold Spring Harbor Perspectives in Biology, 3(1), a004978.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241). Springer, Cham. DOI: 10.1007/978-3-319-24574-4\_28

Shah, A., Shah, M., Pandya, A., Sushra, R., Sushra, R., Mehta, M., Patel, K. and Patel, K., 2023. A comprehensive study on skin cancer detection using artificial neural network (ANN) and convolutional neural network (CNN). Clinical eHealth, 6, pp.76-84. Available at: <https://www.sciencedirect.com/science/article/pii/S2588914123000205>

Shoulders, M.D., Raines, R.T., 2009. Collagen structure and stability. Annual review of biochemistry, 78, pp.929-958.

Sharon, T. (2016) 'The Googlization of health research: from disruptive innovation to disruptive ethics', Personalized Medicine, 13(6), pp. 563-574.

Schindelin, J. , Arganda-Carreras , I. , Friss , E. , et al. (2012) respectively. Fiji: An open platform for bioimage analysis. Nature Methods, 9(7), 676-682.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

Shoulders, M.D. and Raines, R.T. (2009) 'Collagen Structure and Stability', Annual Review of Biochemistry, 78, pp. 929-958.

Sorushanova, A., Delgado, L.M., Wu, Z., Shologu, N., Kshirsagar, A., Raghunath, R., Mullen, A.M., Bayon, Y., Pandit, A. and Raghunath, M. (2019) 'The Collagen Suprafamily: From Biosynthesis to Advanced Biomaterial Development', Advanced Materials, 31(1), 1801651.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z., 2016. Rethinking the Inception Architecture for Computer Vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, pp. 2818-2826. doi: 10.1109/CVPR.2016.308.

The Bureau of Investigative Journalism (2023) 'The Dark Side of the Wellness Industry: Collagen Extraction and Its Environmental and Human Rights Impacts', The Bureau of Investigative Journalism, 23 January. Available at: https://www.thebureauinvestigates.com/stories/2023-01-23/the-dark-side-of-the-wellness-industry (Accessed: 23 January 2023).

Topol, E.J. (2019) 'High-performance medicine: the convergence of human and artificial intelligence', Nature Medicine, 25(1), pp. 44-56.

Van Valen, D.A., Kudo, T., Lane, K.M., Macklin, D.N., Quach, N.T., DeFelice, M.M., Maayan, I., Tanouchi, Y., Ashley, E.A., Covert, M.W., 2016. Deep learning automates the quantitative analysis of individual cells in live-cell imaging experiments. PLOS Computational Biology, 12(11), e1005177.

Williams, R. (2020) 'Data for the Public Good', Nature, 582(7811), pp. 185-187.

Wilson, D. R., & Martinez, T. R. (1997). Instance pruning techniques. Proceedings of the Fourteenth International Conference on Machine Learning, 403-411.

Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

Zhang, M.L. and Zhou, Z.H. (2007) ‘ML-KNN: A lazy learning approach to multi-label learning’, Pattern Recognition, 40(7), pp. 2038–2048. Available at: https://doi.org/10.1016/j.patcog.2006.12.019.

Zhang, S. et al. (2017) ‘Learning k for kNN Classification’, ACM Transactions on Intelligent Systems and Technology, 8(3). Available at: https://doi.org/10.1145/2990508.

Zhang, Y., Lin, M., & Wong, A. (2016). Insights into vesicle motion and interaction during cellular phases. Molecular Biology Reports, 43(8), 857-864.

Zuboff, S., 2019. The Age of Surveillance Capitalism: The Fight for a Human Future at the New Frontier of Power. PublicAffairs.

# **Appendix A (Part 1 Python)**

**Generalized VGG16 Architecture**



**Generalized ResNet50 Architecture:**

A group of blue rectangular objects with black text

Description automatically generated

**CNN 1 Code:**

import numpy as np

import pandas as pd

import os

import tensorflow as tf

from tensorflow.keras import layers, models

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

from sklearn.preprocessing import LabelEncoder

# Set the path to your dataset in Google Drive

data\_path = "/content/drive/MyDrive/Collagen Dataset"

# Load the dataset

num\_classes = 9

data = []

labels = []

class\_folders = os.listdir(data\_path)

# Use LabelEncoder to convert class names to numerical labels

label\_encoder = LabelEncoder()

label\_encoder.fit(class\_folders)

for class\_folder in class\_folders:

    class\_path = os.path.join(data\_path, class\_folder)

    for img\_file in os.listdir(class\_path):

        img = load\_img(os.path.join(class\_path, img\_file), target\_size=(64, 64))

        img = img\_to\_array(img)

        data.append(img)

        labels.append(class\_folder)

data = np.array(data) / 255.0  # Normalize pixel values to [0, 1]

labels = label\_encoder.transform(labels)  # Convert class names to numerical labels

num\_classes = len(class\_folders)  # Number of classes

# Split the dataset into training, validation, and testing sets (60%, 20%, and 20% respectively)

x\_train, x\_temp, y\_train, y\_temp = train\_test\_split(data, labels, test\_size=0.4, random\_state=42)

x\_val, x\_test, y\_val, y\_test = train\_test\_split(x\_temp, y\_temp, test\_size=0.5, random\_state=42)

# Data augmentation for the training set

datagen = ImageDataGenerator(

    rotation\_range=20,

    width\_shift\_range=0.2,

    height\_shift\_range=0.2,

    horizontal\_flip=True,

)

datagen.fit(x\_train)

# Create the CNN model

model = models.Sequential([

    layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(64, 64, 3)),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(128, (3, 3), activation='relu'),

    layers.MaxPooling2D((2, 2)),

    layers.Flatten(),

    layers.Dense(128, activation='relu'),

    layers.Dense(num\_classes, activation='softmax')  # Output layer with num\_classes units

])

# Compile the model

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

# Train the model using data augmentation and validation data

history = model.fit(

    datagen.flow(x\_train, y\_train, batch\_size=32),

    epochs=100,

    validation\_data=(x\_val, y\_val)

)

# Save the trained model to a file

model.save("/content/drive/MyDrive/trained\_modelCNN1.h5")

# Get the folder names of each class

class\_folders = os.listdir(data\_path)

class\_names = class\_folders

# Evaluate the model on the test set

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print("Test accuracy:", test\_acc)

# Generate classification report

y\_pred\_probs = model.predict(x\_test)

y\_pred = np.argmax(y\_pred\_probs, axis=1)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=class\_names))

# Generate confusion matrix graph

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)

plt.title('Confusion Matrix')

plt.colorbar()

tick\_marks = np.arange(num\_classes)

plt.xticks(tick\_marks, class\_names, rotation=45)

plt.yticks(tick\_marks, class\_names)

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# Generate accuracy graph

plt.figure(figsize=(8, 6))

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

plt.show()

# Generate loss graph

plt.figure(figsize=(8, 6))

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='upper right')

plt.show()

**CNN2 Code:**

import numpy as np

import pandas as pd

import os

import tensorflow as tf

from tensorflow.keras.applications import VGG16

from tensorflow.keras import layers, models

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

from sklearn.preprocessing import LabelEncoder

# Set the path to your dataset in Google Drive

data\_path = "/content/drive/MyDrive/Collagen Dataset"

num\_classes = 9

data = []

labels = []

class\_folders = os.listdir(data\_path)

# Use LabelEncoder to convert class names to numerical labels

label\_encoder = LabelEncoder()

label\_encoder.fit(class\_folders)

for class\_folder in class\_folders:

    class\_path = os.path.join(data\_path, class\_folder)

    for img\_file in os.listdir(class\_path):

        img = load\_img(os.path.join(class\_path, img\_file), target\_size=(64, 64))  # Update target size to (64, 64)

        img = img\_to\_array(img)

        data.append(img)

        labels.append(class\_folder)

data = np.array(data) / 255.0  # Normalize pixel values to [0, 1]

labels = label\_encoder.transform(labels)  # Convert class names to numerical labels

num\_classes = len(class\_folders)  # Number of classes

# Split the dataset into training, validation, and testing sets (60%, 20%, and 20% respectively)

x\_train, x\_temp, y\_train, y\_temp = train\_test\_split(data, labels, test\_size=0.4, random\_state=42)

x\_val, x\_test, y\_val, y\_test = train\_test\_split(x\_temp, y\_temp, test\_size=0.5, random\_state=42)

# Data augmentation for the training set

datagen = ImageDataGenerator(

    rotation\_range=20,

    width\_shift\_range=0.2,

    height\_shift\_range=0.2,

    horizontal\_flip=True,

)

datagen.fit(x\_train)

# Create the VGG16 model

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(64, 64, 3))  # Update input shape to (64, 64, 3)

for layer in base\_model.layers:

    layer.trainable = False

# Build your custom top classifier

model = models.Sequential([

    base\_model,

    layers.Flatten(),

    layers.Dense(256, activation='relu'),

    layers.Dropout(0.5),

    layers.Dense(num\_classes, activation='softmax')  # Output layer with num\_classes units

])

# Compile the model

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

# Train the model using data augmentation and validation data

history = model.fit(

    datagen.flow(x\_train, y\_train, batch\_size=32),

    epochs=100,

    validation\_data=(x\_val, y\_val)

)

# Save the trained model to a file

model.save("/content/drive/MyDrive/trained\_model\_CNN2\_VGG16.h5")

# Get the folder names of each class

class\_folders = os.listdir(data\_path)

class\_names = class\_folders

# Evaluate the model on the test set

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print("Test accuracy:", test\_acc)

# Generate classification report

y\_pred\_probs = model.predict(x\_test)

y\_pred = np.argmax(y\_pred\_probs, axis=1)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=class\_names))

# Generate confusion matrix graph

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)

plt.title('Confusion Matrix')

plt.colorbar()

tick\_marks = np.arange(num\_classes)

plt.xticks(tick\_marks, class\_names, rotation=45)

plt.yticks(tick\_marks, class\_names)

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# Generate accuracy graph

plt.figure(figsize=(8, 6))

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

plt.show()

# Generate loss graph

plt.figure(figsize=(8, 6))

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='upper right')

plt.show()

**CNN3 Code:**

import numpy as np

import pandas as pd

import os

import tensorflow as tf

from tensorflow.keras.applications import ResNet50

from tensorflow.keras import layers, models

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

from sklearn.preprocessing import LabelEncoder

# Set the path to your dataset in Google Drive

data\_path = "/content/drive/MyDrive/Collagen Dataset"

num\_classes = 9

data = []

labels = []

class\_folders = os.listdir(data\_path)

# Use LabelEncoder to convert class names to numerical labels

label\_encoder = LabelEncoder()

label\_encoder.fit(class\_folders)

for class\_folder in class\_folders:

    class\_path = os.path.join(data\_path, class\_folder)

    for img\_file in os.listdir(class\_path):

        img = load\_img(os.path.join(class\_path, img\_file), target\_size=(64, 64))

        img = img\_to\_array(img)

        data.append(img)

        labels.append(class\_folder)

data = np.array(data) / 255.0  # Normalize pixel values to [0, 1]

labels = label\_encoder.transform(labels)  # Convert class names to numerical labels

num\_classes = len(class\_folders)  # Number of classes

# Split the dataset into training, validation, and testing sets (60%, 20%, and 20% respectively)

x\_train, x\_temp, y\_train, y\_temp = train\_test\_split(data, labels, test\_size=0.4, random\_state=42)

x\_val, x\_test, y\_val, y\_test = train\_test\_split(x\_temp, y\_temp, test\_size=0.5, random\_state=42)

# Data augmentation for the training set

datagen = ImageDataGenerator(

    rotation\_range=20,

    width\_shift\_range=0.2,

    height\_shift\_range=0.2,

    horizontal\_flip=True,

)

datagen.fit(x\_train)

# Create the VGG16 model

base\_model = ResNet50(weights='imagenet', include\_top=False, input\_shape=(64, 64, 3))

for layer in base\_model.layers:

    layer.trainable = False

model = models.Sequential([

    layers.GlobalAveragePooling2D(),  # Global average pooling instead of Flatten for ResNet

    layers.Dense(256, activation='relu'),

    layers.Dropout(0.5),

    layers.Dense(num\_classes, activation='softmax')

])

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

history = model.fit(

    datagen.flow(x\_train, y\_train, batch\_size=32),

    epochs=100,

    validation\_data=(x\_val, y\_val)

)

# Save the trained model to a file

model.save("/content/drive/MyDrive/trained\_model\_CNN3\_RESNET50.h5")

# Get the folder names of each class

class\_folders = os.listdir(data\_path)

class\_names = class\_folders

# Evaluate the model on the test set

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

print("Test accuracy:", test\_acc)

# Generate classification report

y\_pred\_probs = model.predict(x\_test)

y\_pred = np.argmax(y\_pred\_probs, axis=1)

print("Classification Report:")

print(classification\_report(y\_test, y\_pred, target\_names=class\_names))

# Generate confusion matrix graph

cm = confusion\_matrix(y\_test, y\_pred)

plt.figure(figsize=(8, 6))

plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)

plt.title('Confusion Matrix')

plt.colorbar()

tick\_marks = np.arange(num\_classes)

plt.xticks(tick\_marks, class\_names, rotation=45)

plt.yticks(tick\_marks, class\_names)

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# Generate accuracy graph

plt.figure(figsize=(8, 6))

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

plt.show()

# Generate loss graph

plt.figure(figsize=(8, 6))

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Model Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='upper right')

plt.show()

**KNN Code Having Three Models:**

import numpy as np

import pandas as pd

import os

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

import joblib

# Set the path to your dataset in Google Drive

data\_path = "/content/drive/MyDrive/Collagen Dataset"

num\_classes = 9

data = []

labels = []

class\_folders = os.listdir(data\_path)

# Use LabelEncoder to convert class names to numerical labels

label\_encoder = LabelEncoder()

label\_encoder.fit(class\_folders)

for class\_folder in class\_folders:

    class\_path = os.path.join(data\_path, class\_folder)

    for img\_file in os.listdir(class\_path):

        img = load\_img(os.path.join(class\_path, img\_file), target\_size=(64, 64))

        img = img\_to\_array(img)

        data.append(img)

        labels.append(class\_folder)

data = np.array(data) / 255.0  # Normalize pixel values to [0, 1]

labels = label\_encoder.transform(labels)  # Convert class names to numerical labels

num\_classes = len(class\_folders)  # Number of classes

# Split the dataset into training, validation, and testing sets (60%, 20%, and 20% respectively)

x\_train, x\_temp, y\_train, y\_temp = train\_test\_split(data, labels, test\_size=0.4, random\_state=42)

x\_val, x\_test, y\_val, y\_test = train\_test\_split(x\_temp, y\_temp, test\_size=0.5, random\_state=42)

# Now, we will set up three KNN models with different configurations.

# Define a list of K values for the first KNN model

k\_values\_model1 = [1, 3, 5]

# Choose a distance metric for the first KNN model

distance\_metric\_model1 = 'euclidean'

# Define a list of K values for the second KNN model

k\_values\_model2 = [3, 7, 9]

# Choose a distance metric for the second KNN model

distance\_metric\_model2 = 'manhattan'

# Define a list of K values for the third KNN model

k\_values\_model3 = [5, 11, 15]

# Choose a distance metric for the third KNN model

distance\_metric\_model3 = 'chebyshev'

# Create the first KNN model with the first configuration

knn\_model1 = KNeighborsClassifier(n\_neighbors=k\_values\_model1[0], metric=distance\_metric\_model1)

knn\_model1.fit(x\_train.reshape(x\_train.shape[0], -1), y\_train)

# Create the second KNN model with the second configuration

knn\_model2 = KNeighborsClassifier(n\_neighbors=k\_values\_model2[0], metric=distance\_metric\_model2)

knn\_model2.fit(x\_train.reshape(x\_train.shape[0], -1), y\_train)

# Create the third KNN model with the third configuration

knn\_model3 = KNeighborsClassifier(n\_neighbors=k\_values\_model3[0], metric=distance\_metric\_model3)

knn\_model3.fit(x\_train.reshape(x\_train.shape[0], -1), y\_train)

# Save the first KNN model to a file

joblib.dump(knn\_model1, "/content/drive/MyDrive/knn\_model1.joblib")

# Save the second KNN model to a file

joblib.dump(knn\_model2, "/content/drive/MyDrive/knn\_model2.joblib")

# Save the third KNN model to a file

joblib.dump(knn\_model3, "/content/drive/MyDrive/knn\_model3.joblib")

# Evaluate and print accuracies for all KNN models

# Evaluate the first KNN model on the test set

y\_pred\_model1 = knn\_model1.predict(x\_test.reshape(x\_test.shape[0], -1))

accuracy\_model1 = np.mean(y\_pred\_model1 == y\_test)

print("KNN Model 1 Test Accuracy:", accuracy\_model1)

# Evaluate the second KNN model on the test set

y\_pred\_model2 = knn\_model2.predict(x\_test.reshape(x\_test.shape[0], -1))

accuracy\_model2 = np.mean(y\_pred\_model2 == y\_test)

print("KNN Model 2 Test Accuracy:", accuracy\_model2)

# Evaluate the third KNN model on the test set

y\_pred\_model3 = knn\_model3.predict(x\_test.reshape(x\_test.shape[0], -1))

accuracy\_model3 = np.mean(y\_pred\_model3 == y\_test)

print("KNN Model 3 Test Accuracy:", accuracy\_model3)

# Evaluate the first KNN model on the test set

y\_pred\_model1 = knn\_model1.predict(x\_test.reshape(x\_test.shape[0], -1))

print("Classification Report for KNN Model 1:")

print(classification\_report(y\_test, y\_pred\_model1, target\_names=class\_folders))

# Evaluate the second KNN model on the test set

y\_pred\_model2 = knn\_model2.predict(x\_test.reshape(x\_test.shape[0], -1))

print("Classification Report for KNN Model 2:")

print(classification\_report(y\_test, y\_pred\_model2, target\_names=class\_folders))

# Evaluate the third KNN model on the test set

y\_pred\_model3 = knn\_model3.predict(x\_test.reshape(x\_test.shape[0], -1))

print("Classification Report for KNN Model 3:")

print(classification\_report(y\_test, y\_pred\_model3, target\_names=class\_folders))

# Generate confusion matrix graph for the first KNN model

cm\_model1 = confusion\_matrix(y\_test, y\_pred\_model1)

plt.figure(figsize=(8, 6))

plt.imshow(cm\_model1, interpolation='nearest', cmap=plt.cm.Blues)

plt.title('Confusion Matrix - KNN Model 1')

plt.colorbar()

tick\_marks\_model1 = np.arange(num\_classes)

plt.xticks(tick\_marks\_model1, class\_folders, rotation=45)

plt.yticks(tick\_marks\_model1, class\_folders)

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# Generate confusion matrix graph for the second KNN model

cm\_model2 = confusion\_matrix(y\_test, y\_pred\_model2)

plt.figure(figsize=(8, 6))

plt.imshow(cm\_model2, interpolation='nearest', cmap=plt.cm.Blues)

plt.title('Confusion Matrix - KNN Model 2')

plt.colorbar()

tick\_marks\_model2 = np.arange(num\_classes)

plt.xticks(tick\_marks\_model2, class\_folders, rotation=45)

plt.yticks(tick\_marks\_model2, class\_folders)

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# Generate confusion matrix graph for the third KNN model

cm\_model3 = confusion\_matrix(y\_test, y\_pred\_model3)

plt.figure(figsize=(8, 6))

plt.imshow(cm\_model3, interpolation='nearest', cmap=plt.cm.Blues)

plt.title('Confusion Matrix - KNN Model 3')

plt.colorbar()

tick\_marks\_model3 = np.arange(num\_classes)

plt.xticks(tick\_marks\_model3, class\_folders, rotation=45)

plt.yticks(tick\_marks\_model3, class\_folders)

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

# **Appendix B (Part 2: Fiji+Trackpy+R)**

**Fiji ijm code**

//Open Images in Batch Mode

input\_dir = getDirectory("Input\_Directory");

output\_dir = getDirectory("Output\_Directory\_Images");

suffix = ".czi"; //only apply to specific type of file

//Batch processing

setBatchMode(true);

processFolder(input\_dir);

function processFolder(input\_dir) {

list = getFileList(input\_dir);

list = Array.sort(list);

for (i = 0; i < list.length; i++) {

if(File.isDirectory(input\_dir + File.separator + list[i]))

processFolder(input\_dir + File.separator + list[i]);

if(endsWith(list[i], suffix))

processFile(input\_dir, output\_dir, list[i]);

}

}

function processFile(input\_dir, output\_dir, file) {

//here, you put the real processing

run("Bio-Formats Importer", "open=[" + input\_dir + file + "] color\_mode=Default quiet rois\_import=[ROI manager] view=Hyperstack stack\_order=XYCZT");

file = getTitle();

input\_dir = getInfo("image.directory");

file\_noExt = File.getNameWithoutExtension(file);

//Adjust Brightness and Contrast of the DAB image

resetMinAndMax();

run("Enhance Contrast", "saturated=0.35");

run("Apply LUT", "stack");

//Clean up image

run("Despeckle", "stack");

run("Subtract Background...", "rolling=1 stack");

saveAs("AVI", output\_dir + file\_noExt + "\_cleaned");

close("\*");

run("Close All");

}

showMessage("All Done");

**Trackpy Python code**

//Open Images in Batch Mode

input\_dir = getDirectory("Input\_Directory");

output\_dir = getDirectory("Output\_Directory\_Images");

suffix = ".czi"; //only apply to specific type of file

//Batch processing

setBatchMode(true);

processFolder(input\_dir);

function processFolder(input\_dir) {

list = getFileList(input\_dir);

list = Array.sort(list);

for (i = 0; i < list.length; i++) {

if(File.isDirectory(input\_dir + File.separator + list[i]))

processFolder(input\_dir + File.separator + list[i]);

if(endsWith(list[i], suffix))

processFile(input\_dir, output\_dir, list[i]);

}

}

function processFile(input\_dir, output\_dir, file) {

//here, you put the real processing

run("Bio-Formats Importer", "open=[" + input\_dir + file + "] color\_mode=Default quiet rois\_import=[ROI manager] view=Hyperstack stack\_order=XYCZT");

file = getTitle();

input\_dir = getInfo("image.directory");

file\_noExt = File.getNameWithoutExtension(file);

//Adjust Brightness and Contrast of the DAB image

resetMinAndMax();

run("Enhance Contrast", "saturated=0.35");

run("Apply LUT", "stack");

//Clean up image

run("Despeckle", "stack");

run("Subtract Background...", "rolling=1 stack");

saveAs("AVI", output\_dir + file\_noExt + "\_cleaned");

close("\*");

run("Close All");

}

showMessage("All Done");

**R code**

#Reading data into R  
Data\_Trajectory = read.csv(file.choose())  
  
#Summary Statistics  
summary(Data\_Trajectory)  
sd(Data\_Trajectory$Mean\_Size)  
sd(Data\_Trajectory$Mean\_Size)  
sd(Data\_Trajectory$Mean\_Size)  
sd(Data\_Trajectory$Mean\_Size)  
  
#Correlation Matrix  
cor(Data\_Trajectory)  
  
#First Model: All Variables  
First\_Model = lm(Mean\_Straightness ~ Mean\_Size + Mean\_velocity + Mean\_Distance, data = Data\_Trajectory)  
summary(First\_Model)  
  
#Second Model: Adding count of Undefined Straightness as Dummy Variable  
Second\_Model = lm(Mean\_Straightness ~ Mean\_Size + Mean\_velocity  
+ Mean\_Distance + Count\_Undefined\_Straightness, data = Data\_Trajectory)  
  
summary(Second\_Model)  
resid(Second\_Model)  
  
#Testing validity of the model  
plot(resid(Second\_Model))