Deep Representation Learning of Mitochondrial Dynamics

by

Rachel Mattson

(Under the Direction of Shannon Quinn)

Abstract

Certain diseases, and the success of their treatment, are reflected in the structure of mitochondria within human cells. Automating the characterization of mitochondrial states may accelerate trials to find life-saving drugs. This study explores the use of deep learning tools in preliminary characterization of cell states. We attempt to model morphological changes over the time, using a Convolutional Neural Network (CNN) to embed frames, then comparing several established methods for aggregating time information across frames. We train this deep model with a classification task to emphasize the presence of mitochondrial fission and fusion in the representation. After obtaining this representation, we embed the video in a low dimensional space to show a cohesive progression through the course of a video. This reveals structure in the frames that link to mitochondrial events, showing promise for detecting behavior shifts.

Index words: [Biomedical Imaging, Deep Learning, Video Modeling, Time Series Analysis, Mitochondrial Morphology]

Deep Representation Learning of Mitochondrial Dynamics

by

Rachel Mattson

B.A., University of Georgia, 2022

A Thesis Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the
Degree.

Master of Science

Athens, Georgia

2023

©2023
Rachel Mattson
All Rights Reserved

Deep Representation Learning of Mitochondrial Dynamics

by

Rachel Mattson

Major Professor:	Shannon Quinn
Committee:	Fred Maier
	Fred Quinn

Electronic Version Approved:

Ron Walcott
Dean of the Graduate School
The University of Georgia
August 2023

Dedication

To Hild, my cat. Let's hope this degree keeps us stocked with treats and toys for many years.

Acknowledgments

Much appreciation for the QRG lab and my labmates - I'm excited to see how each of your futures progress. We will have to do a podcast reunion to catch up on the rabbit-holes we've committed to. I'm especially grateful to Dr. Quinn and Marcus, whose kind words in a time of illness had a much greater impact on me than you guys would know. Thanks to my committee for supporting this research project. I would like to thank Dr. Maier for helping me navigate the MSAI program, vouching for me, and for the tireless help he extends to our entire cohort.

Sending love to my friends, roommates and siblings who supported me through the highs and lows of the last few years. I couldn't do it without you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Literature Review

1.1 Introduction

1.1.1 Motivation

The goal of this study is to model broad structural changes of mitochondria, with the hope of eventually using similar methods on the specific structural changes that result from pathogen invasion of the human cell. Such a model would be useful to assess the health of the cell as a whole. It could also help identify states of disease. Previous work shows that deep learning tools such as convolutional neural networks (CNNs) and long short term memory (LSTM) are capable representing broad mitochondrial states. These investigations primarily focus on learning still frames of mitochondria to classify current states. Unlike some of their more static counterparts, mitochondria are quite dynamic over the course of a single life-cycle. Thus, modeling mitochondria well requires modeling mitochondrial changes over time. This has broader implications in offering clinicians new ways to assess and treat diseases.

1.1.2 Objective

Our aim is to provide a temporal analysis of mitochondrial states using basic deep learning architectures such as CNNs and LSTMs. We contend that a bare minimum temporal analysis should be able to 1) show an interpretable cellular progression from states at the beginning to end of the video and 2) learn for short-term time dependencies between nearby frames.

In this study, we work with videos of mitochondria that are tagged with fluorescent proteins. The cells housing these mitochondria are placed in several different environments that provoke different dynamics over the course of the cell's life (discussed further in section 2.1.1). We focus on capturing short-term mitochondrial behavior; the changes that take place in a tiny fraction of the cellular life-cycle. We break videos of mitochondrial dynamics down into many small segments, model these segments, then show how they develop over the course of the video. This representation yields sections of the video that share similar behavior. This preliminary model may be useful in detecting shifts in the data, such as cell collapse. Optimistically, this method may eventually help detect stages of pathogenic invasion.

1.2 Biological Significance

Mitochondria play a crucial role in several cellular processes. While their most famous function is to provide "immediate" energy to cellular processes, they also have a complex role within cellular regulation as a whole. They are closely linked to the health of the cell and mediate cell death when called to do so. They interact with bacterial pathogens—for example, *Mycobacterium tuberculosis*, the bacteria responsible for causing the disease tuberculosis—and they respond to such invasions with specific changes in morphology [16], [26].

In their natural state, mitochondria stretch across the eukaryotic cell in a complex network of tubular protein structures. These bodies grow, merge, shrink, and divide in order to deliver their functionality to any part of the cell in need.

As this organelle has a strong link to many diseases, modeling it could empower wide scale research studies in areas such as drug discovery. Better models of mitochondria could also also benefit research into other organelles with similar dynamics. As it stands, manual analysis of mitochondrial dynamics is a time intensive task that requires trained experts. Automation of some of this analysis could open the door for bigger, broader studies of mitochondria.

1.2.1 Roles

Mitochondria's foremost role is to make energy available immediately and locally within a cell [9]. In a healthy cell, mitochondria are spread out to support ATP-dependent events, i.e. protein functions such as pumps or enzymes. They also support other local processes like calcium regulation, which is crucial in muscle tissue [13]. The fluctuating structure of mitochondria makes for a multifaceted temporal modeling problem.

Another wide role that mitochondria play lies in propagating information across the cell. They can even trigger cell death (apoptosis) if there is sufficient need; this pathway sparks extreme internal fragmentation of the mitochondria and communication of a "death signal" to the rest of the cell [19]. Apoptosis can be a marker of disease. If the body is fighting a disease, "shutting down" one cell may be the best choice for the health of surrounding cells. For instance, apoptosis might prevent pathogens from replicating within the relative safety of a cell. This mechanism protects nearby tissue from the spread of disease.

1.2.2 Disease

The response of mitochondria to disease is more complex than the broad cell states explored so far in the literature (section 1.3). Modeling the nuances of disease progression necessitates some integration of temporal information.

Mitochondria have evolved to respond to cellular dysfunction. They facilitate a lot of cross-talk in a cell regarding events like inflammation and autophagy [5]. This connection to the immune system makes them crucial in fighting disease. Defects in mitochondrial functions have been linked to various respiratory diseases, cancer, metabolic disorders, and neurodegenerative disorders such as Alzheimer's, Parkinson's, and Huntington's disease [9].

Several infectious diseases not only impact mitochondria, but exploit them. A studied example of this is the pathogen *M. tuberculosis*, which invades cells and embeds itself within mitochondrial systems. While the bacteria often lie dormant, they can also activate and engage in a complex interaction with mitochondria and the cell. The first stage of this interaction, where the bacteria seek to survive, includes altering the form and function of the organelle to provide nutrients to the invader [23]. The second stage involves more cell-wide influence, as the invader seeks to propagate and expand to new cells, without being detected by the immune system.

Mitochondria have great influence in deciding whether a cell is still viable [19]. This has significant consequences for the immune system. A disease that can manipulate mitochondrial regulation can preserve the life of its host cell. There is evidence that M. tuberculosis does just this: inhibiting mitochondrial pathways that signal for cell disposal [30]. Thus, not only is mitochondrial structure an indicator of cell health during invasion, but restoring mitochondrial health would be a key way to repel *M. tuberculosis* [3].

The complexity of *M. tuberculosis* invasion requires a global view of cell events to identify the disease and the success of the cell's response. Not only do they display a bimodal progression, but they can cause specific morphological changes that serve as clues to the disease's advancement. Models that only capture broad cell states may not be useful to clinicians seeking to fight disease. Our integration of temporal information is the first step towards a more nuanced model.

1.2.3 Dynamics

Mitochondria are some of the most complex organelles in the human cell. They have their own DNA, ribosomes, and a host of specialized proteins. While other organelles, like nuclei have a rather fixed form and location, mitochondria spread throughout an entire cell, propagating wherever they are needed. Mitochondria are best conceived as a network, with an interconnected protein structure sometimes branching, forming rings, and extending into terminal nodes [6]. This network is in constant flux.

Proteins in a mitochondria's membrane signal for fission or fusion with other mitochondria. Fission and fusion often determine which portions of the organelles are supported, and thus are engaged in a constant dialogue over cell-wide health.

Our model tracks visual changes in our 2D slice of tagged mitochondrial proteins. The most salient in our data occur from the process of fusing and fissioning mitochondria. Nanoscale repairs and lysosome-mediated processes are much more difficult to identify at the scale of our videos. Our data does not distinguish whether protein disappears from view due to movement, division, or destruction. Thus visible changes due to fusion and fission become the bedrock upon which our model is built.

These processes can become imbalanced, leading to extreme cell states. We focus on cells that have been artificially altered to induce imbalances in fusion and fission. The hyperfused state of mitochondria occurs when there is too little fission, and the hyperfissioned (fragmented) state occurs when there too much fission.

1.2.4 Mitochondrial Structure and Imaging

Modeling mitochondria from confocal microscope imaging is a challenging task.

An important aspect of modeling mitochondria is that significant parts of the mitochondrial structure are not visible under light microscopy. While proteins in the folds of mitochondria are easily tagged, the microtubule scaffolding of the cristae is less opaque. A 2021 study explored 3D electron microscopy of mitochondria and estimated that "42% of cristae structure surface exhibits tubular structure that are not recognizable in light microscopy [33]. These structures are critical to temporal characterization because much of the motion of mitochondrial protein is controlled by non-visible structure. Mitochondria move on tubular tracks, which are only discernible in the paths these organelles trace over time. These tubular structures and their resulting motion are only one part of mitochondrial dynamics, but are not negligible.

However, electron microscopy data is unsuitable for our purposes. We intend to extend mitochondrial models to temporal features of mitochondrial dynamics. Electron microscopy is not capable of taking videos, at least not at an acceptable frame rate. It requires a fixed cell, and it cannot be extended to live cell imaging. Thus, we must work with light microscopy knowing that it is missing critical structural information.

Beside dealing with hidden structure, this study is limited to 2D frames that show a horizontal slice of mitochondrial form. A whole dimension of movement is lost to us. This introduces ambiguity into tracking the structural changes—for example, when tagged protein appears in frame, it is unknown whether that is a result of protein growth, or some rotation or translation of existing protein from a different "slice" of the cell.

As such, we must choose a flexible model that can account for these unseen dynamics. Handcrafting features from visible information is risky; we seek to use deep methods to add more robust features to this body of work.

1.3 Mitochondria Modeling

1.3.1 Image Processing and Machine Learning

Spatial

Foundational work on mitochondrial modeling seeks to identify what the eye can see; the branching, endpoints, and isolated nodes in a mitochondrial network. Work like [24] uses image analysis techniques to cast branch points to a set of vectors. They then took measurements on the length and volume of branches. [35] extended this work into a software package, providing a tool to measure network features and identify relationships. They also measure the number of these features. For instance, they measure the average length between branches and count the number of toruses in the mitochondrial network. Note that these works used Z-stack imaging, which retains some 3-dimensional data and is useful for determining the true form of the mitochondrial network. This allows the formation of a holistic mitochondrial skeleton. The reliability of this tactic is effected by preprocessing and imaging artifacts. Different, brittle thresholds can cause erroneous merging or tearing of the learned mitochondrial skeleton.

These handcrafted features, like average branching length or number of rings, are good predictors of cellular state. Prior work[8] shows that a random forest classifier can identify these cell states, just from measurements of segmented mitochondrial networks. Our previous work shows that successful state classification is possible from a graph representation of the mitochondrial network [27]. Both these works build intermediate representations of morphology and classify these representations with some success.

Temporal

Zahedi et al[38] present a pipeline that extends model capabilities to video data. In order to classify cell states compute intermediate features such as the length or radius of a mitochondrial body. However, they also compute pixel intensity differences over time to gather temporal features like direction and speed. Styling temporal features like this has advantages: calculating the magnitude and direction of motion preserves key information from the videos. Notably, the authors also provide a measure of "texture". They then feed this information into a suite of simple machine learning classifiers, capturing hyperfusion, hyperfission, and apoptosis well. The advantages of calculating the flow of pixel intensity between frames is clear; the model is able to represent some of the fusion and fission behavior, rather than only the extreme results of these processes gone wrong.

This paradigm is taken a step further in Mitometer, which uses morphological measurements and changes in pixel intensity to add each individual mitochondrion to a rigid internal representation [20]. They cast mitochondrial dynamics into a set of tracks—the paths of mitochondrial bodies over time. The main difficulty of their work is how to determine when two mitochondria merge or split. Cell state classifications on top of this work yield similar accuracy as previous works [21], [38].

Issues with Classic Machine Learning Methods

These classic methods have some benefits. Most notably, they are relatively interpretable. A biologist knows what to do with features like a low branching number, presumably. Deep representations are not always so interpretable; they often build robust yet unintuitive features. Thus it is worth mentioning a few drawbacks of the classic methods, which deep learning can help overcome:

1) There may be some useful information lost when going from images to a set of measurements or a graph representation. A Deep Model is given all this information at train time and can learn what to disregard.

2) Handcrafted features can be brittle, sensitive to slight changes in the data. Deep features tend to be more robust to variations.

3) we need more than coarse state classification—we need a robust representation. Powerful classifiers like Random Forests do not build a representation

of the data that does anything other than classify. We want a representation that we can query in other ways, which deep learning provides.

1.3.2 Deep Methods

Deep methods have proven to be excellent at classifying still frames of mitochondria into general morphological categories. These works incorporate temporal domain by extending inference to more than one still frame. For instance, our previous work summarizes a video of mitochondrial dynamics as the first and last frames of the video [21]. Convolutional Neural Networks (CNNs) are shown to be capable of classifying these into broad morphological categories. However, this mostly captures the effect of laboratory stimuli - for instance, mitochondria that are induced to hypofission eventually turn into a tight mass. "Before" and "after" captures very little of the mitochondrial dynamics.

Similarly, [15] extends a one frame CNN classifications to multiple frames. They model a video by inferring the category for each still frame, then classifying the video with the majority label of all the frames. They achieve 98% accuracy on a binary "normal" or "abnormal" classification task, which are promising results for an anomaly detection application.

Both these methods show that CNN's are perfectly adequate for representing still mitochondrial morphology. Yet, they do not account for the temporal changes. For instance, taking only two frames is a severe summarization of the information available in the video, and will surely not generalize to more diverse clinical scenarios. Using frame labels to vote on the video label provides a stronger summary, but does not incorporate past temporal information to inform the next frame's classification.

We set out to provide a model that models still frames and the changes that occur between them.

1.4 Deep Learning for Video Methods

1.5 Supervised Methods

Video learning draws on models from image analysis and time series analysis to capture relationships in both space and time.

Researchers have extended CNNs to include 3D convolutions that process local information in space and time through the same kernel [4], [17]. By passing a 3D filter over a stack of images, they produce features that aggregate information from the space and time domains. Tran et al showed that changing some or all layers of a residual network to 3D convolutions can improve accuracy on action recognition benchmarks [34]. However, a convolutional filter's ability to compose non-local relationships across longer sequences is limited by the filter's size. At deeper layers of the model, abstract relationships may be learned across longer dependencies, but this is a dark art rather than a guarantee by the architecture.

In order to learn longer dependencies, research has turned to sequence modeling techniques used in NLP. Architectures like Recurrent Neural Networks (RNNs) seek to propagate signals across sequences [28]. In this paradigm, time-series units, or "tokens", are iteratively passed through the same RNN module, which maintains a stream of information influenced by each past state. However, this architecture also struggles with maintaining strong long-term dependencies, not because of filter constraints, but because it is difficult to push changes back along its learning gradient without collapse. In response to this problem, specialized gated RNNs were developed, such as the Long Short Term Memory module (LSTM) [12]. By combating gradient loss, this architecture achieves success at learning longer dependencies than the vanilla RNN. These do not claim to model truly "long" dependencies—they still are inclined to learn local relationships.

A natural strategy to video modeling is to use CNNs to generate the input sequence to a recurrent neural network. LSTMs accept a variety of inputs, such as patches of an image or percepts from pretrained 2d CNNs. One tactic is to simply flatten 2D image convolutions into individual vectors that form the sequence. Another tactic is taken by ConvNet, which maintains the input's dimensionality within the recurrent module, using convolutional operators to pass information between iterations [31].

The supervised component of our work is focused on learning short-term behavior: namely the "action" of fusion or fission. As such, LSTMs present a reasonable solution for learning temporal patterns. In our work, we train a CNN to output percepts of individual frames from 2D convolutions, then feed this sequence of outputs into an LSTM. This composite CNN-LSTM is trained from scratch—no part of the system is frozen or pretrained.

1.5.1 Inductive bias

Inductive bias refers to the collection of assumptions a model makes in order to guide its learning. Selecting the most useful inductive biases will allow the model to learn useful patterns in the data. When working with small datasets, it is especially important to select models with a good inductive bias because the model will have less opportunity to build good assumptions about the data from scratch. One reason Convolutional Neural Networks are so successful at image analysis is their design is biased towards learning shape and texture in image data. When trained to infer from textural features, CNNs can become quite reliant on texture as a predictor. Given the "high-frequency" morphology of mitochondrial dynamics, this preference towards texture is desirable in our model. CNNs have the additional advantage of building translational invariance through the use of pooling layers in the models. The cells in our data exist in different locations in frame, with no bearing on their internal state. Thus translational invariance is highly desirable.

LSTMs contain other inductive biases that are appropriate for our modeling goal. Naturally, their sequential processing of input is appropriate for a time series problem. LSTMs cannot look back to access information from old time points, so they are forced to push this information onto a constantly maintained hidden state. They have a recency bias, where timestamps more recently seen have a greater impact on future predictions. The overall effect is to boost the learning of short-term dependencies.

In addition to the LSTM model, we explore two simpler methods of aggregating the temporal information over a handfuls of frames. First, we try flattening frame outputs into a single vector. Secondly, we take the mean of each frame output. These are more low-tech solutions than an LSTM, but they give us a baseline to compare against. If they are able to learn, then the heavier parametrization of an LSTM may be unnecessary.

1.6 Unsupervised Methods

1.6.1 Dimensionality Reduction

Unsupervised methods like clustering seek to find meaningful groupings in the underlying structure of datasets. In order to learn reliable clusters, the data needs to be transformed into the appropriate representational form. The first step towards this in our study is learning the deep embeddings of our videos, the second is to reduce the dimensionality of this embedding.

A host of clustering methods were developed to cluster low dimensional data and they tend to be less reliable at increased dimensions. "The curse of dimensionality" is a catch-all phrase for issues which arise in computations at a high enough dimension. In these spaces, many metrics begin to fail. For example, the euclidean measure of distance is affected by dimensionality; as you add components to vector inputs, each component has less impact on the whole. At high enough dimensions, differences in components become negligible to the output, and the euclidean metric becomes less useful. As a result, common clustering algorithms built on similar metrics, like k-means and DBSCAN, will lose their efficacy.

Dimensionality reduction refers to the practice of projecting high dimensional data on a less complex space. Typically, these algorithms exploit some underlying structure in the data to map it to meaningful axes, such as a set of axes that explain the most variance in the dataset. With reducing the representational space of the dataset, choices must be made about which information is preserved. Typically, there is a trade-off between preserving global vs local structures - one can prioritize the greater landscape of the dataset, or prioritize the local neighborhoods that often show meaningful relationships between data points.

UMAP

UMAP is a popular tool due to its ability to balance this trade-off [22]. UMAP works by constructing a graph where connections are made from each datapoint to other points that fall within a certain radius of the datapoint. As the radius extends outwards, such edges are weighted to be "less likely" connections. (Prioritizing connections within the tight radius has the double benefit of devaluing connections that may by touched by the curse of dimensionality). This construction tends to heavily push topological information onto sparsely connected datapoints - simple forms in the graph like 'lines' or 'triangles' preserve a lot of the important information. This reduces the computational difficulty of the latter step of UMAP, which is to find an optimal match from the complex graph to a simple graph.

This simple graph is then cast back into a lower dimensional projection, with the data ripe for analysis.

Sparse PCA

Sparse PCA [18] is a variant of Principal Component Analysis that attempts to find axes containing the most variance in high dimensional data. Sparse PCA finds these axes by first selecting a subset of variables that describe each vector. As a result, the low dimensional projections are more representative of key relationships in the data, and less likely to be composed of many smaller relationships. This is meant to yield more interpretable axes that map to real world meanings.

Overall, both UMAP and Sparce PCA exploit "sparsity". UMAP deals purely with graph structure, and reflects a balance of local and global features. PCA more focused on global structure. Sparse PCA captures less variance in the reduced axes than traditional PCA and this is helpful if we want an axis that has some has some interpretation of fusion and fission.

Chapter 2

Experiments

2.1 Methods

2.1.1 Data Collection

The dataset consists of confocal microscopy video footage of epithelial HeLa cells collected for [6]. The videos span 24 hours: roughly the length of one life cycle for these cells.

Mitochondria were fluorescently tagged with the DsRed2- Mito-7 protein and filmed with a Nikon A1R confocal microscope. Thus, the data tracks intensity of the fluorescent signal in a 512x512 frame as it changes over time [6]. The footage collected one frame every 10 seconds, resulting in at least 20,000 frames per video. Each video contains multiple HeLa cells.

The cellular samples were artificially induced into three distinct morphologies. The first set of samples was exposed to Listeriolysin O (LLO), a pore-forming toxin which induces mitochondrial fragmentation. The second set of samples was exposed to mitochondrial division inhibitor 1 (mdivi), which induces mitochondrial hyperfusion. A final set of samples remain unaltered to serve as a control group. Examples of these three classes are shown in Figure 2.1.

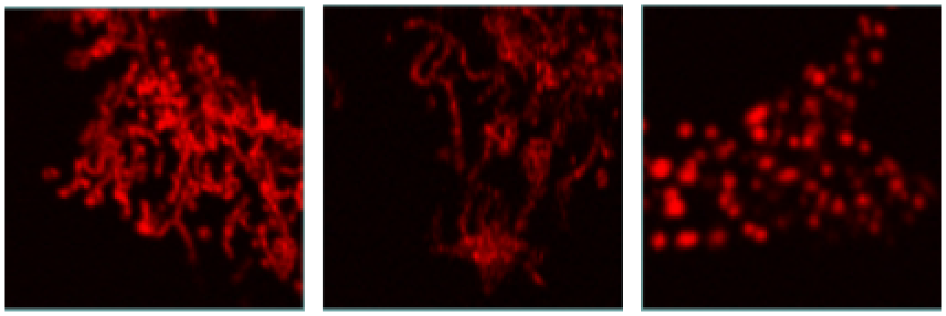
Figure 2.1: video frames from mitochondria in 3 different states, from left to right: normal, hyperfused, and hyperfragmented

Exposure to LLO and Mdivi creates extreme imbalances in mitochondrial fission and fusion processes, respectively, within affected cells. While affected classes display fragmentation and hyperfusion, all three classes can share aspects of fission and fusion at some time point.

2.1.2 Preprocessing

As each video contains multiple cells, we segment videos into sections containing one cell to obtain a set of videos tracking a single cell's development. The methodology is the same as in previous works on this data [6], [11], [27]. This consists of seeding each video with a segmentation mask generated by ITK-SNAP [37], then tracking each cell's progression over the video through iterative dilation, thresholding, and contour detection.

As development is fairly slow, the number of frames is downsampled from 20,000 to 200. Before, differences in the video were about 10 seconds, and after, they are about 7 minutes apart. This downsampling is used in previous OrNet work, leaving this study open to comparison with models. Some videos are truncated to shorter time frames due to loss of signal. For example, a tracked cell leaves the frame.

We then used median normalization to minimize the effect of light fluctuation over the frames over a video. This results in 114 videos of single HeLa cells. There are 54 instances of LLO, 31 of Mdivi, and 29 of healthy control.

2.1.3 Dataset construction

Our dataset contains a small number of videos, but a plethora of information along the temporal axis. Each video was decomposed into sets of 5 frames, spanning 0.25-0.4% of each video. The CNN training aimed to capture the difference between hypofission and hyperfission, rather than the categories LLO and Mdivi. These classes have overlapping structure; there is some fission in Mdivi, and some fusion in LLO. It would be preferable to only have datapoints with the respective imbalance implied by each class. Since each LLO video typically takes longer to develop an imbalance of fission, the first 50 frames of each LLO video were omitted from the training set.

The control group is not trained on, and serves as a true control to test the final model on unseen behavior. This allows us to reflect on what attributes the model actually uses to distinguish between the two classes.

In order to train the model, the dataset was split into three partitions: the train, validation, and test sets. These classes are balanced with each having slightly more LLO videos in each partition to reflect its majority in our dataset. The validation and test partitions were chosen to make sure the videos reflect the diversity of the training set. In the data, Mdivi tends to have a similar effect, whereas in LLO, some cells are more greatly effected by the toxin. There is additional diversity in the placement of cells, with some being large and centered, some smaller but still entirely visible, and some leaving the edge of the frame. Furthermore, some cells fade entirely from view as the cell presumably dies. In 2.3, we see this reflected in the test. It contains 3 similar Mdivi videos, three LLO videos with a reflection of the toxin, and two LLO videos that stay pretty static. It mostly contains medium sized, entirely visible cells, but also contains a larger cell, an occluded cell, and a cell that fades nearly to no signal.

To summarize, the dataset construction is as follows: First the train/validation/test split was performed on the set of videos. Second, the respective videos were split into chunks consisting of 5 frames, omitting the beginning of LLO videos. Last, to prevent the model from seeing a batch with samples all from the same video, the chunks were randomly shuffled one time upon instantiation of the training dataset.

These were implemented using PyTorch and scikit-learn.

2.1.4 Models

The model architecture used can be broken into three components: a CNN, an aggregator, and a training task. The CNN gathered deep features from each individual frame, and the aggregator joined sets of frames together to form a single vector in the final representation space. The training task consisted of a single dense layer mapping to these final representations to a binary classification.

SqueezeNet served as the CNN backbone of the model [14]. Squeezenet is a smaller version of AlexNet, one of the foundational class of CNN architectures that has achieved great empirical success. The model consists of several submodules that "squeeze" data through a pointwise filter, and "expand" the results into many feature maps. Downsampling is performed late, to keep activation maps broad and to best exploit the limited number of parameters. We used a vanilla version of SqueezeNet, containing no skip connections. ReLu served as the activation function in the network.

The SqueezeNet outputs were passed through a final dense layer to yield a single vector representation of a frame. These were aggregated through 3 methods: flattening, mean, and an LSTM. In "flattening" the frame vectors were simply concatenated into larger vector. In mean, a new vector with the same dimensions as an individual frame vector was constructed by taking the mean across each element of the frame vectors. In the LSTM, a vector of the same size as the frame vectors was learned by iteratively passing the sequence of frame vectors through this deep model. These aggregators are treated as sub-experiments, and the efficacy of each choice of aggregator is examined in

The training task was simply to map frame sequences back to their original category: hyperfused Mdivi or hyperfragmented LLO. This was achieved by a single dense layer taking in the aggregated frame representation.

All parts of the model were implemented with PyTorch [25]. An untrained implementation of SqueezeNet was taken from pytorch hub, the original output layer removed, and the aggregator and training task appended.

2.1.5 Training

The complete model was trained to minimize cross entropy loss in the binary classification task. The loss function is shown below, where p represents the probability that an input belongs to a particular class ADAM was used as the optimizer.

$$L_{BCE} = -(y \log(p) + (1 - y) \log(1 - p))$$

Training deep models on a small dataset leads to challenges. Deep models, equipped with many parameters, may simply use this wealth of parameters to "memorize" data. In other words, they may learn overly specific features found in the dataset that allow for high accuracy at train time. However, these brittle patterns are often insufficient to map new data to a high quality representation. For example, a model may learn the pattern of a cell collapsing by memorizing the exact shape, location, and velocity of the pixels representing the cell boundary. Without a more abstract parameterization of the "collapse", the model may be confounded by unseen collapses in new configurations. Thus, the goal of training these models is to encourage the model to distribute information between parameters in a more satisfying strategy. The standard way to address this problem is to insure that the model is not "overfitting"; The training loss should not outstrip the validation loss. When working with a dataset as small as our study, overfitting is almost inevitable.

There are several methods to combat overfitting. First of all, the representational power of the model should be appropriate for the scale of the dataset. In our study, we selected SqueezeNet as the base model due to its small size, or fewer parameters. Additionally, the training task was instantiated in a single dense layer, to force representation learning to happen primarily in the CNN and aggregation component.

Second, regularization like dropout was used [32]. Dropout discourages the model from relying on a single parameter to store key information. In order to learn in a setting where no weight is guaranteed to be active on the forward pass, the model must distribute info between parameters. This method quickly overturns brittle mappings during training, and prevents hash-like solutions. Additionally, this may encourage "redundancies" in the mapping. This leads to healthy models which tend to rely on more abstract layers to form the final representation.

Third, regularization like image augmentation is crucial to generalization. The dataset does not contain enough variance in pose, position, scale, and pixel intensity to prepare the model to generalize to very similar clinical settings. Image augmentation serves the dual purpose of artificially diversifying the dataset, and downweighting features overrepresented in the train set. For example, rotations are used to show the model cells in more orientations. On the other hand, artificially altering brightness attempts to knock the model's confidence in pixel intensity as a useful predictor. Our video data contains artifacts where the intensity of pixels will change over the course of the video with no corresponding change in mitochondrial mass. The model could rely on this artifact. Synthetically perturbing frame brightness in the train set discourages the model from relying on this artifact.

The result of image augmentation is that training inputs have an equal chance of being rotated 0, 90, 180 or 270 degrees. The brightness of a frame has a 50% chance of being increased by 10%. All frames in a chunk have the same transformation applied.

Hyperparameter Tuning

Hyperparameter tuning first consisted of a wide search over learning rate, batch size, dropout, weight decay, and shuffling. The best models had a learning rate of 1e-05 , batch size of 16, no weight decay, and negligible impact of shuffling.

After this, additional experiments were run with dropout in two locations - the final dense output of Squeezenet, and in the LSTM module. Both the final Mean and LSTM models contained 50% dropout rate in the final layer of the CNN. The final LSTM model had a dropout rate of 25% between iterations of the LSTM Module. All final models shuffled the training set between rounds. Training used early stopping.

Training of the model was implemented in PyTorch Lighting [7]. All random behavior was controlled with 2 seeds - one global seed for the lighting methods, and 1 seed for the PyTorch dataset construction. Training was distributed between two NVIDIA TITAN X (Pascal) GPU cores.

2.1.6 Dimensionality Reduction and Clustering

Deep embeddings were obtained by passing the test dataset through the fully trained model and hooking the outputs of the penultimate layer. This yielded set of vectors in R^{32}, each representing a chunk of the video. Standard scaling was applied to these vectors. An instance of UMAP with 2 components was fitted to these embeddings, resulting in projections for the data. These were then plotted as individual videos.

The 2D UMAP plots were fitted with a 30 neighbors and .3 minimum distance between projected points. This strikes a balance between global and local for our dataset.

Additional Sparse PCA plots were made to attempt to project the video onto 1 interpretable axis.

2.2 Results

2.2.1 Model Results

The test set consists of 8 videos verbally described in Table 2.3. Each is referenced by its class and an alphabetic number for the remainder of the plots.

The test results are shown in 2.1. The UMAP embeddings for all three aggregators are shown in Figure 2.4. As LSTM 2.1 is the only model resulting in a test score above random chance, these are the only dimension reduction plots explored further. The rest can be found in the Appendix.

Figures 2.5 through 2.9 show 2D UMAP projections of the 8 videos, with each point representing a chunk of frames and each color denoting the timestamp on the first frame of the chunk. Figures 2.14 show these point projecting on only the x-axis through sparse PCA.

Model	Train		Val		Test
	Accuracy (%)	Loss	Accuracy (%)	Loss	Accuracy (%)
Mean 2.2	84.4	.390	92.7	0.332	45.1
Flatten 2.2	87.5	.290	96.4	0.247	47.1
LSTM 2.1	56.3	.686	83.6	0.570	68.9
LSTM 2.2	93.8	.184	95.0	0.132	43.0

Table 2.1: Accuracy and loss scores for final models

Cross-validating Models to Confirm Low Accuracy

To confirm the uneven findings of the model's accuracy, a cross-validated training of the LSTM model was completed. This was a 10-fold cross validation upon the combined train and validation splits, leaving the test split the same as in the previous training. The average train loss was .324 and the average validation loss was .257. Train Accuracy was 91% and validation accuracy was 88.4%. Yet the average train accuracy was 49.6% accuracy.

Brief investigation of Morphological vs Temporal Features

The embedding had both morphological and temporal features to draw from. In order to shed light on how much the temporal features were explored by the models, a revised test set was formed where instead of each time-frame consisting of 5 successive frames, the time-frame consisted of the first frame repeated 5 times. The resulting test accuracies are shown in figure 2.2

Model	Test Set	Flat Test Set
Mean 2.2	45.1	45.5
Flatten 2.2	47.1	46.7
LSTM 2.1	68.9	68.4
LSTM 2.2	43.0	44.2

Table 2.2: Differences in test accuracies given all frames versus just given the first frame

Video Label	Annotation
Mdivi A	A cell in the top right quadrant with several "arms" extending out from it. Network development is slow and subtle, with branches merging and little division. The cell shape remains static.
Mdivi B	An elliptical cell in the center of the right half of the frame. Network development is observed, with branches forming and slightly changing location. There is some division of branches but no punctuate structures. The cell shape remains static.

Mdivi C	A round, diffuse cell in the center of the left half of the frame. Network development is observed, with initially branches merging. There is a dip in brightness, then the same slow merging continues, without a major structural change.
LLO A	A large cell shows signs of fragmentation. After a flare of brightness, the fragmentation continues followed by a sharp collapse. The cell slides towards the center of the frame and deteriorates rapidly, eventually only leaving the barest speck of illumination.
LLO B	An elliptical cell in the top center. Excessive fragmentation is observed, followed by a swift collapse. The cell slides to the center of the frame and loses a third of its mass. It then remains fairly static.
LLO C	A visible ring in the bottom left quadrant. Both division and merging of the visible portion is observed. No collapse or movement.
LLO D	A fairly static cell in the bottom right quadrant. Over the course of the video, there is a faint fragmentation of protein. The brightness of the video fluctuates several times. There is no “collapse” event or any sort of movement.

LLO E	A large cell centered in the frame, with clear definition of cellular protein. The proteins display fragmentation, becoming more punctate. A shift is observed, with boundaries of the cell shrinking. However, the center of the cell stays static - no translation.

Table 2.3: Verbal description of each video in the test set

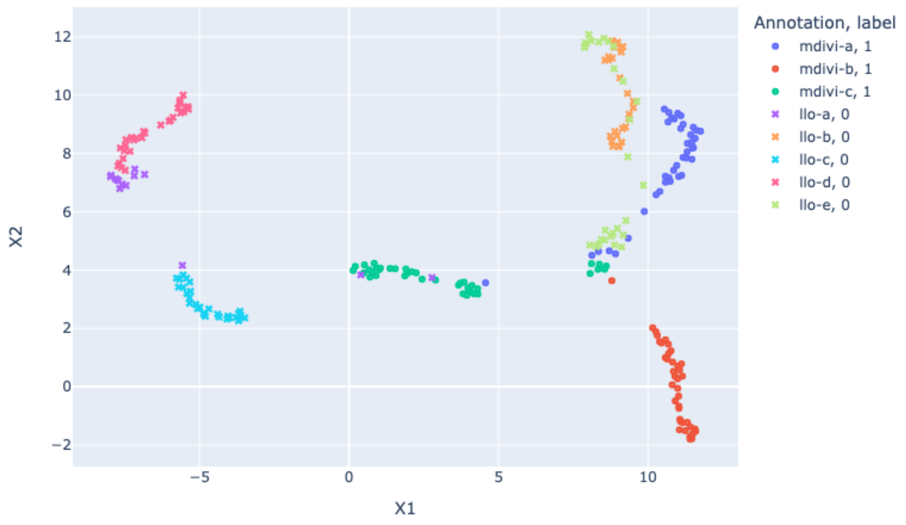

Figure 2.2: 2D UMAP projections from the LSTM embedding.
Videos are shown in different colors. LLO C, D, and the end of A are separated from the rest (these are very static videos). LLO B and E have very similar structure, but are entangled with Mdivi

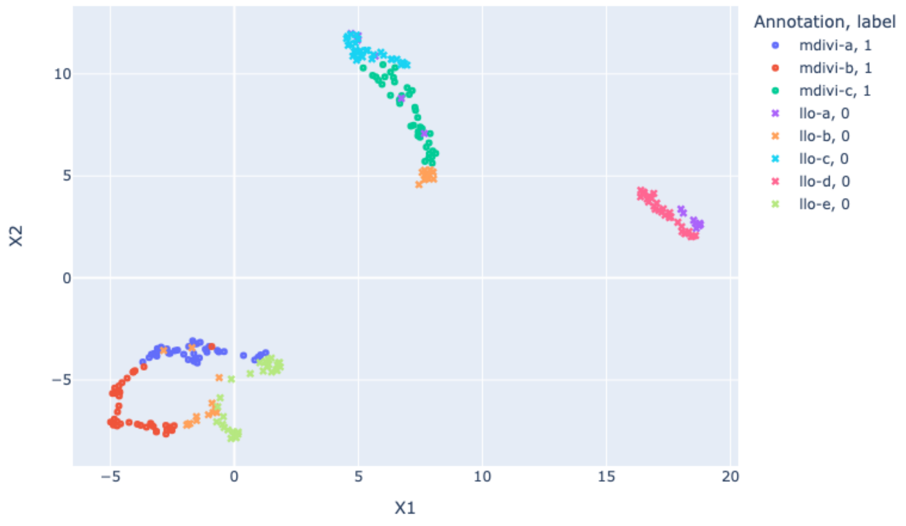

Figure 2.3: 2D UMAP projections from the Flatten embedding.
LLO D and the very end of are separated from the rest, but LLO C and the beginning of LLO A are far away, adjacent to Mdivi C. LLO B and E are less similar than in the LSTM plot, but are still entangled with Mdivi

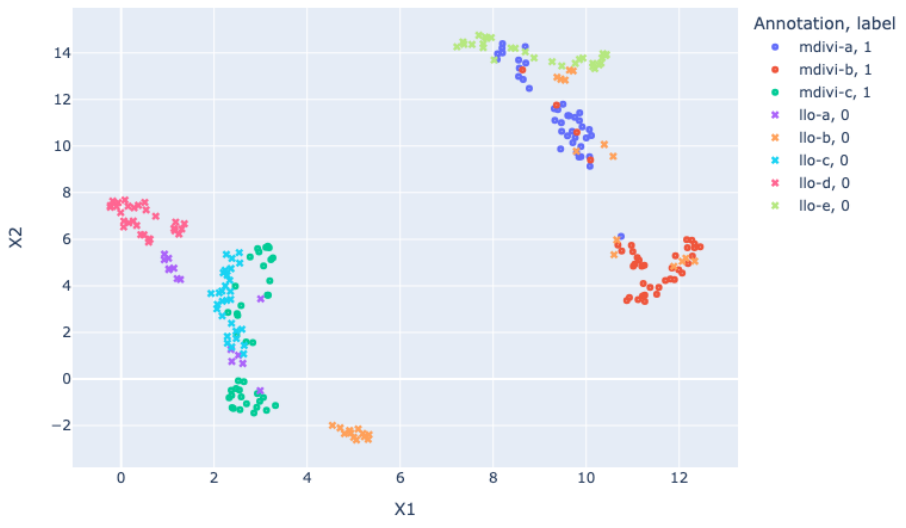

Figure 2.3: 2D UMAP projections from the Mean embedding
Of the plots this has the least linear progression, with some videos changing directions and looping back around. LLO C, D, and the end of A are not separate from Mdivi C (Though this makes it look like the relationship in the Flatten embedding may be a tear in the graph.) LLO B and D are not very similar, but are both entangled with Mdivi.

Figure 2.4: Global UMAP Plots

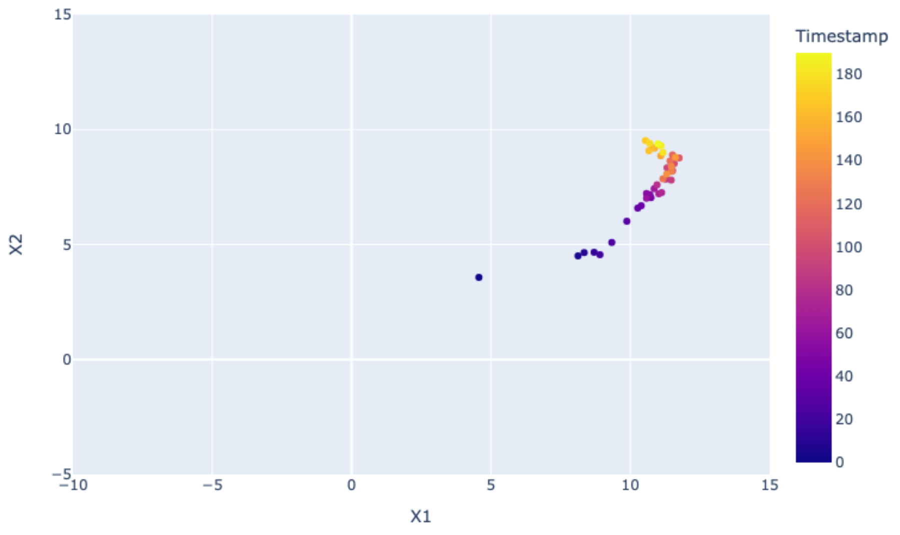

Figure 2.5: Mdivi A - LSTM 2.1 - 2D UMAP

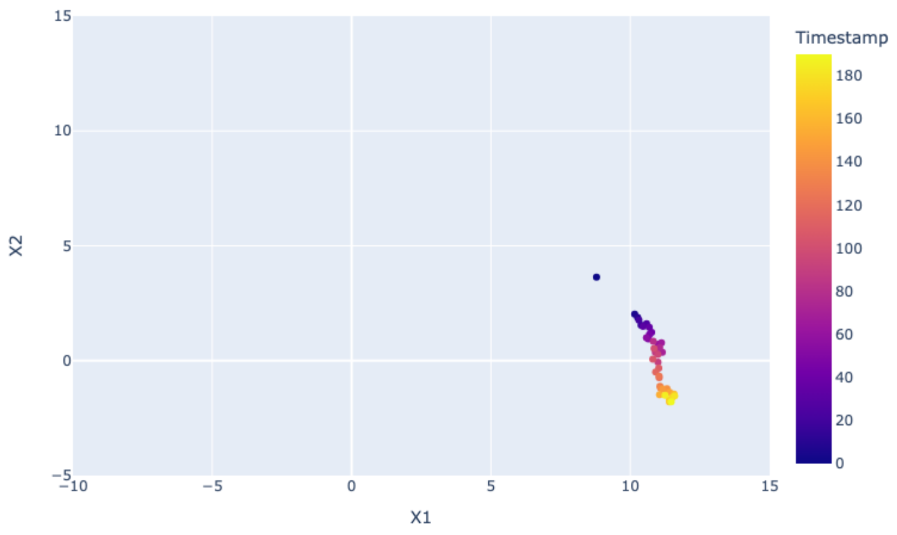

Figure 2.6: Mdivi B - LSTM 2.1 - 2D UMAP

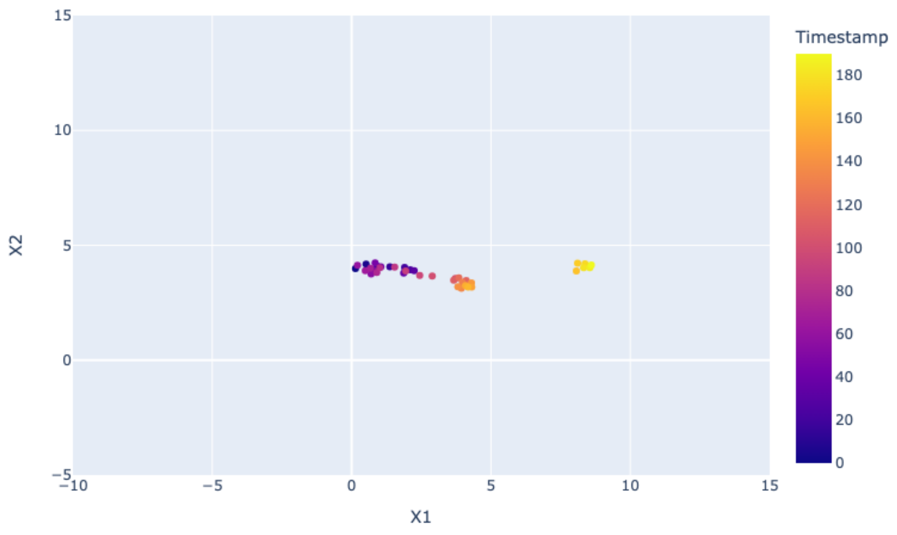

Figure 2.7: Mdivi C - LSTM 2.1 - 2D UMAP

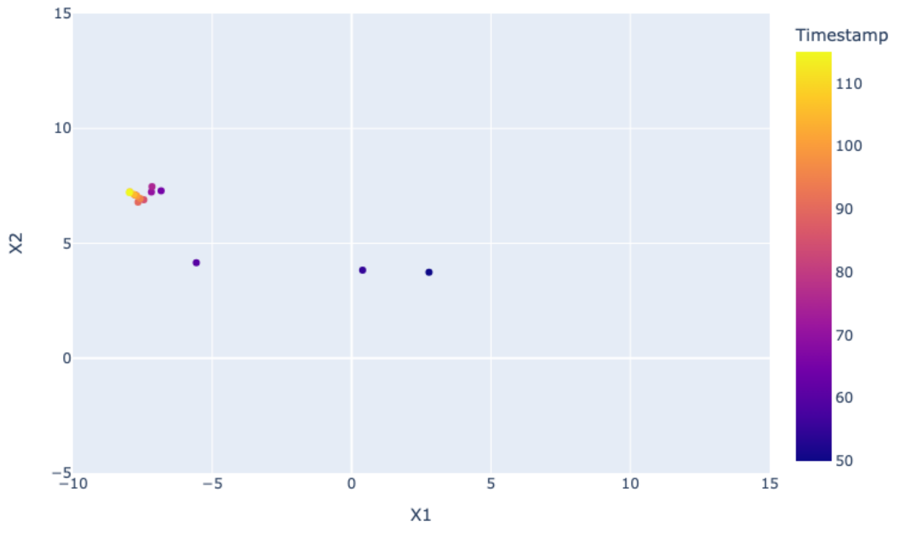

Figure 2.8: LLO A - LSTM 2.1 - 2D UMAP

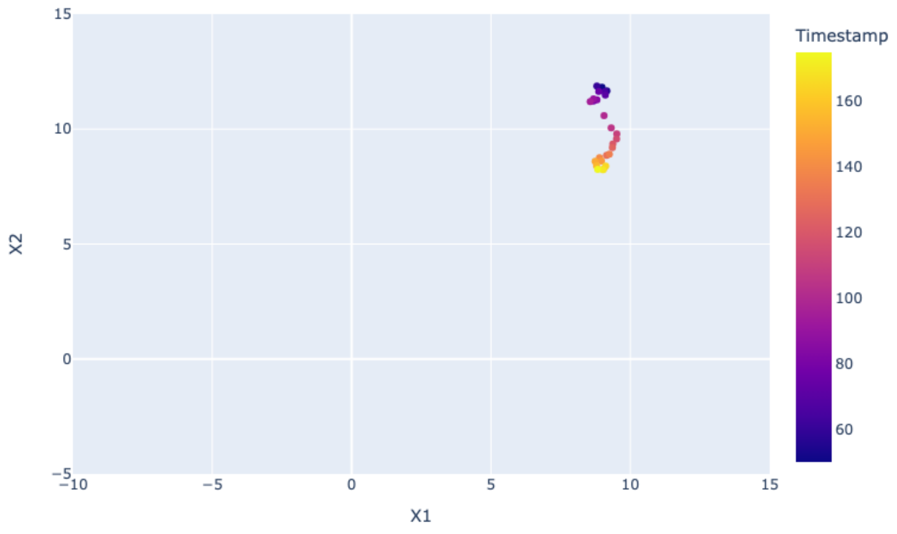

Figure 2.9: LLO B - LSTM 2.1 - 2D UMAP

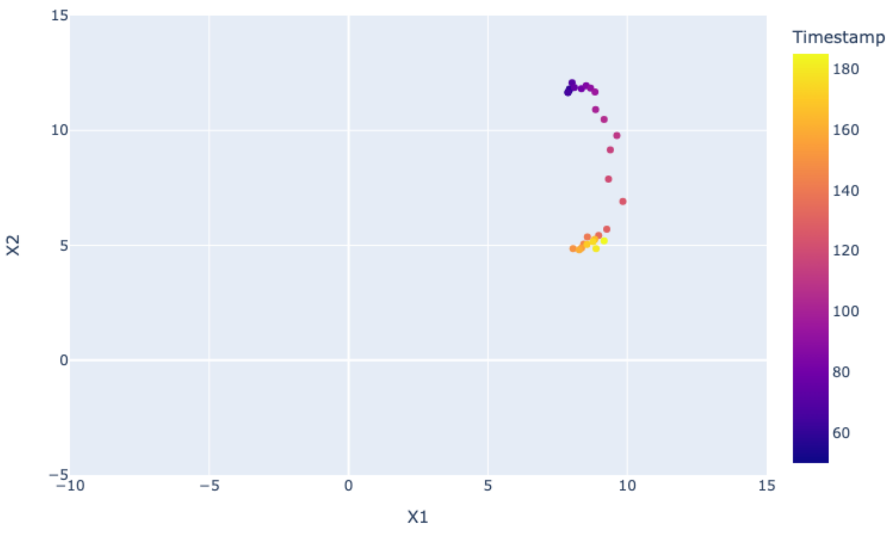

Figure 2.9: LLO E - LSTM 2.1 - 2D UMAP

Figure 2.10: 2D UMAP projections showing LLO videos with a discernible collapse - LSTM 2.1

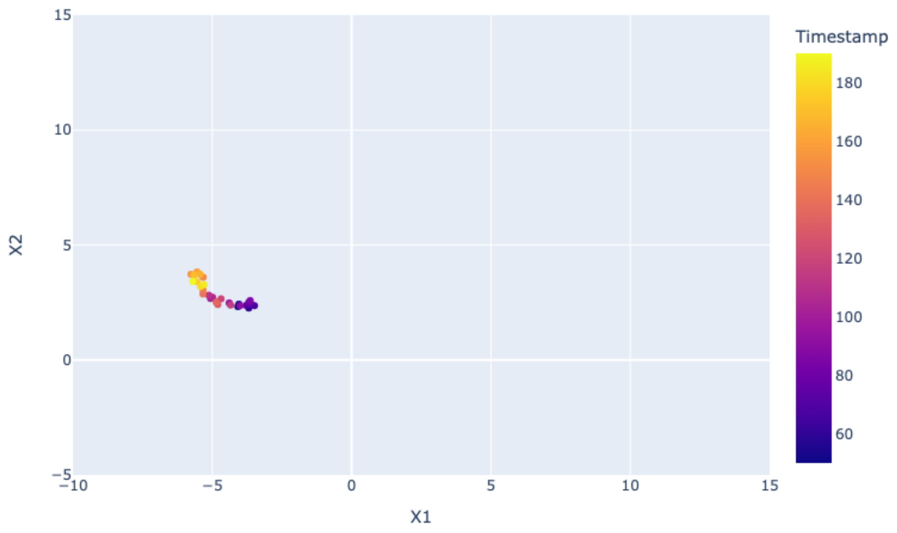

Figure 2.11: LLO C - LSTM 2.1 - 2D UMAP

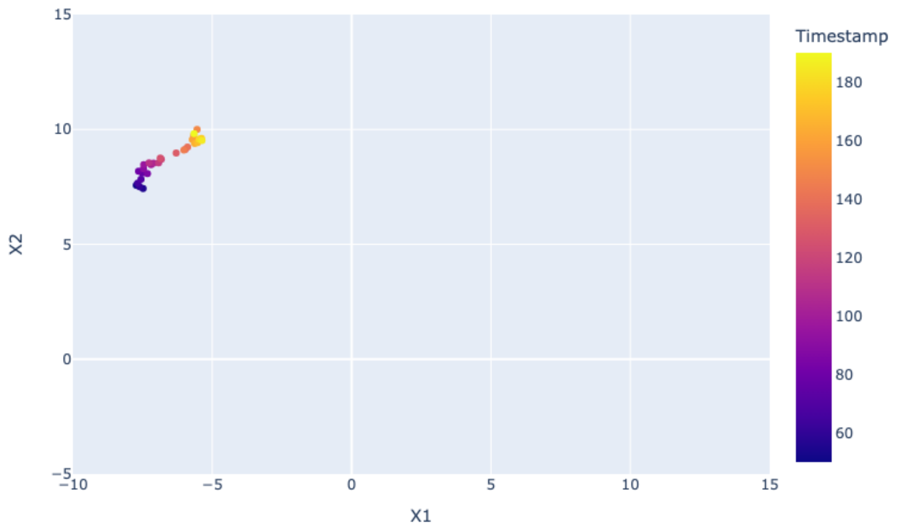

Figure 2.12: LLO D - LSTM 2.1 - 2D UMAP

Figure 2.13: 2D UMAP projections showing LLO videos without a discernible collapse - LSTM 2.1

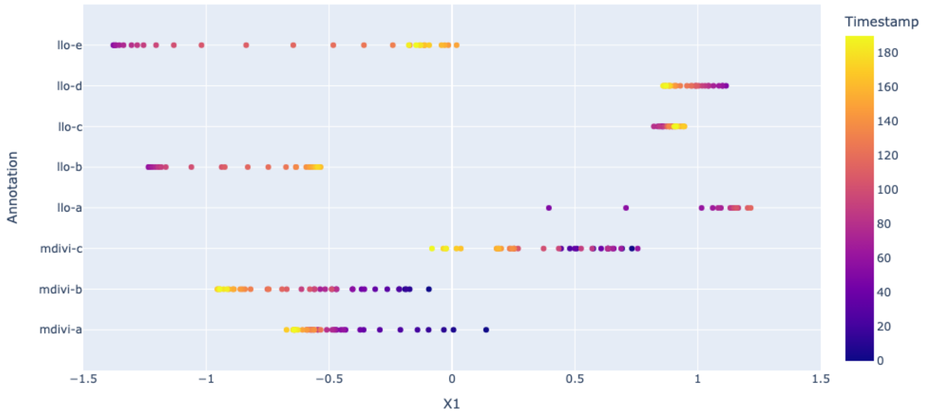

Figure 2.14: 1d PCA Plot.
Each video is projected onto a single axis. Mdivi and LLO videos progress in different directions, overlapping their location. LLO videos with more cellular shifts show a distinct spread compared to the tightly clumped, static LLO videos.

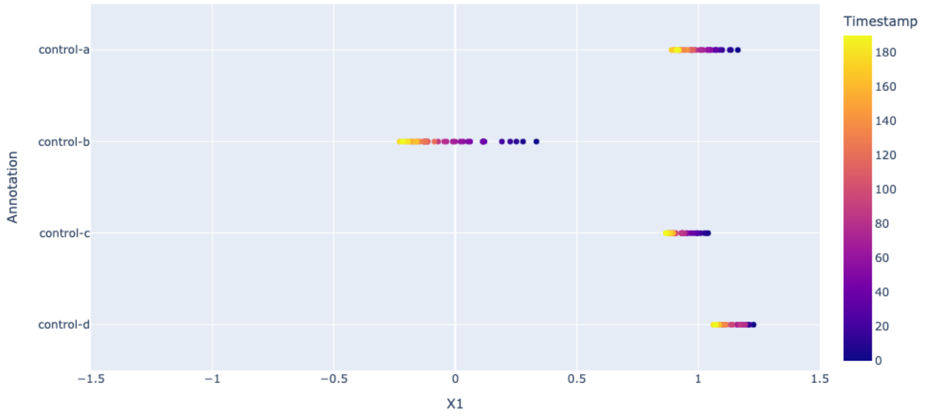

Figure 2.15: 1d PCA Plot for healthy control videos.
Each video is projected onto a single axis. The model has never trained on this class of videos so they are technically OOD. These videos all display similar dynamics, fusing and fissioning in balance. However, Control A, C, and D all remain tightly clumped (similar to the static LLO videos) while control B has a greater spread.

Chapter 3

Discussion and Future Work

3.1 Discussion

3.1.1 Lack of Generalization

Despite an extensive hyperparameter search, no satisfying, generalizable solution is found. It is possible to fit a model that does well on the train and test set, as shown by LSTM version 2.2. This model achieves train/val accuracy of 93.8/95.0%, with loss as low as .184/.132, respectively. However, this solution has the worst test accuracy, at 43%. This indicates that while the model is able to represent the data points well enough, it is not forced into a truly generalizable solution.

Likewise, The Mean and Flatten models perform worse/around random, and the LSTM slightly out-performs random. Thus, these trained models do not successfully generalize to the test set.

The model with the worst train and test scores, LSTM 2.1, happens to perform the best on the train set. This should be interpreted as finding a lucky solution that works for all dataset splits.

As we conducted a thorough hyperparameter search, it is likely that this failure to generalize has more to do with the dataset. Previous work dealt with severe truncations of the videos. Expanding the study to a temporal domain introduces a lot more variance. This puts a strain on the small dataset, where 108 videos are unlikely to show all the diversity of mitochondrial behavior that will be found in the validation and test set. Without the opportunity to master the variance in the population of videos, the model will learn features from the train set that yield a local optima in the loss function. Sometimes, these features will transfer to the validation set, yielding a nice drop in validation loss. We cut the training off at this point, because we hope that the learned parameters that work for the validation set will also work for the test set. Yet in our case, they do not.

Instead of controlling overfitting, the validation set just becomes folded into the problem. There are clearly non-optimal, non-generalizable solutions that predict the train and validation sets without predicting the test set. In this landscape, early stopping is more akin to guessing.

The inductive bias of the model is not strong enough to make up for a lack of good training data. In other words, it is unlikely that a more extensive hyperparameter search would be less fruitful than improving on the existing dataset or model. This issue is discussed further in section 3.2.3.

Still, poor classification could be stemming from overfitting in the decision layer. There might be some quality in the pre-decision representation, thus the next sections continue on to interpret the UMAP projections of this layer. While we examine the representation layers of all three models types in the next section, we then will only seriously consider the LSTM 2.1 model, as it is the only model with some indication of good learning.

Utilization of Time Domain

Section 2.2.1 examines whether there is any difference between giving the model all the frames in a section (how the model was trained), or just the 1st frame in a section. The results show very similar test accuracies on both tasks. The Flatten and LSTM 2.1 aggregators drop by less than half a percentage point without the full time information, and the other models gain a similarly small increase in accuracy. Given the poor quality of the mapping it's uncertain that these changes mean anything significant. However, if the model was using information in the time domain we'd probably see a significant drop in accuracy when this information was taken away. Because we do not see such a drop, it is likely that the model is relying purely on morphology (the structure seen in a still frame).

3.1.2 UMAP Projections

Global insights

Examination of all the videos in Figure 2.4 reveals that while the Mdivi and LLO classes are not linearly separable, the plots seem to capture some local structure for each video. Similar videos have similar distributions. There is a clear progression from structures at the beginning of each video to the end of the video, with some variance (see figs 2.5 through 2.9). This behavior would follow from the CNN capturing the morphology of each frame; similar shapes in the frames yield neighbors in the embedding.

Much of the low test score seems to stem from the entanglement of Mdivi A with LLO B and D, which occurs in all three aggregators. However, this is the only significant entanglement in LSTM 2.1, whereas the Mean and Flatten projections have additional conflations: Mdivi A, B and LLO B, E are intertwined, as well as LLO A,C,D with Mdivi C. However, the LSTM model is able to separate Mdivi A from LLO. It also separates LLO C,D and the end of LLO A from the rest of the embedding. This is a meaningful association, because LLO C and D are videos where nothing happens, and the end of LLO A is a collapsed cell with very low protein signal and little dynamics.

Due to a lower quality embedding and worst test accuracy, the Mean and Flatten models are dropped from the rest of this analysis. The potential of the LSTM embedding is explored further.

Video Insights - UMAP

The progression of timestamps in each video should be taken with a grain of salt, the reduction is just putting them in order because nearest neighbors will be most similar morphologically by default.

The projections generated by LSTM show a few meaningful relationships with the data. The first common thread is the rather smooth progression of Mdivi timestamps. These tend to move in the same direction*, with earlier timestamps being spread a little farther apart than later timestamps. There is some mingling of points which may reflect the way the fusion progresses in the cell, or other artifacts of the training.

Another common thread between aggregators is the difference between LLO embeddings that contain a cellular collapse and cells that stay static. Both the LLO A and LLO B videos contain a cell collapse. In the projections of LLO A, there is a clear separation between initial, early timestamps, and the rest of the later timestamps. In projections of LLO B, there is a more complex structure with two clumps for early and later timestamps, with middle timestamps (between frames 100 and 120) having the most distance between them structure.

Videos of LLO cells that do not collapse have an analogous counterpart in the projections. These projections are clumped very tightly together compared to their more dynamic counterparts. There is some progression of the spread, but nothing particularly distinct. This non-development may be a useful indicator of cell health, as it accurately reflects how these cells did not take on much cellular change at all.

Video Insights - Sparse PCA

The 1D projections attempt to provide a single, interpretable axis that captures how much the cell has fused or fissioned. These plots seem to capture cellular progression, at least in morphology.

Mdivi data points are projected from right to left, whereas LLO data points, in the dynamic videos, proceed from left to right. These are not entirely in order, with some mixing of timestamps occurring in Mdivi B and especially Mdivi A, especially near the end of the video. Similar to the UMAP projections, LLO B and E are mapped to a similar area, whereas LLO A ends up near the static LLO C and D mappings. Note that LLO E timestamps progress from left to right, then loops back to after about frame 140.

The projected embeddings for the wild-type videos are shown in Figure 2.15. These videos are all of slightly different cells, of similar sizes undergoing a similar balance of fusion and fission. In the videos, control A, C and D are all tightly grouped in a similar area as the static LLO videos and the totally collapsed portion of LLO A. However, Control B is located to the left, overlapping with Mdivi and active LLO projections. It also spans a wider area than the other control videos. There is little in the video that explains this aberration, so it may be an artifact of the embedding.

3.2 Future Work

Future work for this project involves steps to learn a better embedding through advancements to the model, training task, and dataset.

3.2.1 Improvements to Training Task

The classification task is too limited for our data. First of all, a "good enough" classification can be made from overly specific attributes of the data, and classification does not force a beautiful, generalized solution that applies to the validation/test set. Secondly, given the results of 2.2, it is clear that the classification task does not force the model to rely on any temporal features. The classification results are near the same whether 5 frames of information are given or one frame.

Generative training paradigms like autoencoders can spur a better embedding in latent space in order to perform tasks such as image reconstruction. These have the advantage of requiring less annotation of the data. However, these come with their own issues. Most significantly to our data, reconstruction loss contains biases towards global shape. There is a higher penalty for getting a shape with a large volume wrong with the right texture, than for getting the right texture but the wrong volumetric shape. A generative training task will need to take special care to preserve important local features.

An autoencoder tasked with recreating a short sequence would pick up on temporal features, but it would not necessarily give special attention to fusion or fission. As a result, an autoencoder fine-tuned on the hyperfusion/fragmentation classification task would be ideal. This model would get the learning benefits of the autoencoder with an emphasis on the features we care about.

3.2.2 Improvements to Model

We sought to learn short-term dependencies in a supervised manner, then model global dynamics with unsupervised tools. The resulting representation shows a progression of "events" in the video, but does not indicate any concrete relationships between the small time chunks. This is sufficient structure to represent broad cell states, but may not be sufficient for modeling more realistic scenarios. For instance, cellular invasion by a pathogen may incur changes in mitochondria morphology that connect to much later cell events. Explicitly learning longer dependencies may be a helpful tool for our purposes.

Video models that incorporate self-attention decompose a video into a set of spatiotemporal tokens, consider all possible relationships between tokens, and learn which relationships are the most predictive of some learning task [2], [10], [36]. This paradigm is borrowed from natural language processing, where their notion of tokens (words, subwords) is rather rigid.In video modeling, a diverse set of "tokens" may be appropriate. These make look like single frames passed through a CNN, the result of 3D convolutions, or the representations in this study.

Some studies take multi-headed self-attention to its fullest extreme, replacing all modules of a video model with transformers [1], [29]. Transformers require more data to learns than CNNs, and they do not contain the same inductive biases that make CNNs more appropriate for mitochondrial dynamics. As such, the middle ground approach is recommended: use convolutions to generate spatiotemporal tokens, then self-attention to learn longer temporal relationships.

3.2.3 Improvements to Dataset

Deep learning methods are notoriously data-hungry. Quality footage of live mitochondria takes a lot of labor and expertise to obtain. As a result, the literature is peppered with studies at a similar scale to our study; cells numbered in the 100s. The variation of imaging conditions are unique to each study, yet monolithic within the study. Thus these small datasets do not span a very diverse partition of the mitochondrial imaging domain. Not only does this limit the models and training that can be done though deep learning, but it leaves regular machine learning models open to skewed dataset splits, generating misleading accuracy scores. For clinical settings, we need stronger guarantees of accuracy.

3.2.4 Augmenting Data

Our dataset was augmented to increase variance in pose. Future work recommends even more use of these augmentations, with more rotations, tessellations, and even skewing of frames. One might also include local patches of the frame, which are typically predictive of fragmentation. The end effect is to encourage the model to rely on the relational change in protein over features like brightness and location.

In particular, tessellations may have the greatest impact on whether the cell is relying on local protein transformation or global shape. Tessellations may look like duplicating a single cell to fill up the input space. However, it may be simpler to return to the unsegmented data with multiple cells in the same frame. Including some to all cells may discourage the model from relying on the cell boundary for decisions. However, cells that are spread spread apart may be less helpful for this purpose as the CNN can simply combine single cell lessons deep into the network. We want flush tessellations.

Ultimately, these attempts to make the dataset stronger will not be enough. A deep model robust enough to be trusted in a clinical setting will most likely need more data than we are working with in these experiments.

3.2.5 More Data

Though one solution is to pay for the acquisition of more data, another solution is clear: merge datasets. With the combined data of many labs, organized and well-documented, we could create a solid benchmark dataset accessible to all researchers. The creation of similar benchmark datasets has spurred machine learning development in fields like image recognition and natural language processing. It is the best practice for research of this nature, as it allows scientists to compare their methods against an objective target. Creating this public dataset would spur rapid evolution of deep learning models on the task of representing mitochondrial dynamics.

Chapter 4

Conclusion

This work explores the extension of deep models to modeling mitochondrial dynamics. The combination of a CNN with an LSTM to model short time frames shows some promise in capturing morphological changes in the cell. The model embeds sections of the video in a sensible order mostly because similar structures occur in adjacent frames. This embedding can be projected into lower dimensional structures that differentiate between hyperfusion and hyperfission through the direction the video progresses on a projected axis. There is also a discernible difference between projections of videos where the cell collapses and the cell stays static. Future work could explore this structure to map shifts in the projection back to frames of the video.

However, the model suffers from weak generalizability and under-utilization of the temporal domain. It is recommended that future work change the training paradigm to first embed short sections of the video with an autoencoder, then fine-tune on the hyperfusion/fragmentation classification task to emphasize this behavior in the representation. After these basic steps, it may be fruitful to explore other mechanisms for learning temporal dependencies like attention. In order to generate reliable results, more data is needed. Future work is held back by the partitioning of biomedical imaging data. If biomedical deep learning is to succeed at the academic level, comprehensive benchmarks must replace small datasets that produce unreliable accuracy results in the literature.

Overall, deep learning tools show promise for cellular imaging problems. However, much more work is needed to successfully capture the dynamics of organelles like mitochondria. Success on classification tasks isn't sufficient for learning nuanced dynamics. Such nuance is critical to building useful models for pressing questions like pathogenic invasion.

Bibliography

[1] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lucic, and C. Schmid, "Vivit: A video vision transformer," *CoRR*, vol. abs/2103.15691, 2021. arXiv: `2103.15691`. [Online]. Available: `https://arxiv.org/abs/2103.15691`.

[2] A. Arnab, C. Sun, and C. Schmid, "Unified graph structured models for video understanding," Oct. 2021, pp. 8097–8106. DOI: `10.1109/ICCV48922.2021.00801`.

[3] S. Asalla, K. Mohareer, and S. Banerjee, "Small molecule mediated restoration of mitochondrial function augments anti-mycobacterial activity of human macrophages subjected to cholesterol induced asymptomatic dyslipidemia," *Frontiers in Cellular and Infection Microbiology*, vol. 7, Oct. 2017. DOI: `10.3389/fcimb.2017.00439`. [Online]. Available: `https://doi.org/10.3389/fcimb.2017.00439`.

[4] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2011, pp. 29–39. DOI: `10.1007/978-3-642-25446-8_4`. [Online]. Available: `https://doi.org/10.1007/978-3-642-25446-8_4`.

[5] N. S. Chandel, "Mitochondria as signaling organelles," *BMC Biology*, vol. 12, no. 1, May 2014. DOI: `10.1186/1741-7007-12-34`. [Online]. Available: `https://doi.org/10.1186/1741-7007-12-34`.

[6] A. Durden, A. Loy, B. Reaves, *et al.*, "Dynamic social network modeling of diffuse subcellular morphologies," in *Proceedings of the Python in Science Conference*, SciPy, 2018. DOI: `10.25080/majora-4af1f417-000`. [Online]. Available: `https://doi.org/10.25080/majora-4af1f417-000`.

[7] W. Falcon and The PyTorch Lightning team, *PyTorch Lightning*, version 1.4, Mar. 2019. DOI: `10.5281/zenodo.3828935`. [Online]. Available: `https://github.com/Lightning-AI/lightning`.

[8] G. M. Fogo, A. R. Anzell, K. J. Maheras, *et al.*, “Machine learning-based classification of mitochondrial morphology in primary neurons and brain,” *Scientific Reports*, vol. 11, no. 1, Mar. 2021. DOI: 10.1038/s41598-021-84528-8. [Online]. Available: https://doi.org/10.1038/s41598-021-84528-8.

[9] J. R. Friedman and J. Nunnari, “Mitochondrial form and function,” *Nature*, vol. 505, no. 7483, pp. 335–343, Jan. 2014. DOI: 10.1038/nature12985. [Online]. Available: https://doi.org/10.1038/nature12985.

[10] R. Girdhar, J. J. Carreira, C. Doersch, and A. Zisserman, “Video action transformer network,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019. DOI: 10.1109/cvpr.2019.00033. [Online]. Available: https://doi.org/10.1109/cvpr.2019.00033.

[11] M. Hill, M. Fazli, R. Mattson, *et al.*, “Spectral analysis of mitochondrial dynamics: A graph-theoretic approach to understanding subcellular pathology,” in *Proceedings of the Python in Science Conference*, SciPy, 2020. DOI: 10.25080/majora-342d178e-00d. [Online]. Available: https://doi.org/10.25080/majora-342d178e-00d.

[12] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[13] D. A. Hood and S. Iqbal, “The role of mitochondrial fusion and fission in skeletal muscle function and dysfunction,” *Frontiers in Bioscience*, vol. 20, no. 1, pp. 157–172, 2015. DOI: 10.2741/4303. [Online]. Available: https://doi.org/10.2741/4303.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, *Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size*, 2016. arXiv: 1602.07360 [cs.CV].

[15] M. S. Iqbal, S. El-Ashram, S. Hussain, *et al.*, “Efficient cell classification of mitochondrial images by using deep learning,” *Journal of Optics*, vol. 48, no. 1, pp. 113–122, Jan. 2019. DOI: 10.1007/s12596-018-0508-4. [Online]. Available: https://doi.org/10.1007/s12596-018-0508-4.

[16] S. Jamwal, M. K. Midha, H. N. Verma, A. Basu, K. V. S. Rao, and V. Manivel, "Characterizing virulence-specific perturbations in the mitochondrial function of macrophages infected with mycobacterium tuberculosis," *Scientific Reports*, vol. 3, no. 1, Feb. 2013. DOI: `10.1038/srep01328`. [Online]. Available: `https://doi.org/10.1038/srep01328`.

[17] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan. 2013. DOI: `10.1109/tpami.2012.59`. [Online]. Available: `https://doi.org/10.1109/tpami.2012.59`.

[18] I. M. Johnstone and A. Y. Lu, *Sparse principal components analysis*, 2009. arXiv: `0901.4392 [math.ST]`.

[19] M. Karbowski and R. J. Youle, "Dynamics of mitochondrial morphology in healthy cells and during apoptosis," *Cell Death &amp Differentiation*, vol. 10, no. 8, pp. 870–880, Jul. 2003. DOI: `10.1038/sj.cdd.4401260`. [Online]. Available: `https://doi.org/10.1038/sj.cdd.4401260`.

[20] A. E. Y. T. Lefebvre, D. Ma, K. Kessenbrock, D. A. Lawson, and M. A. Digman, "Author correction: Automated segmentation and tracking of mitochondria in live-cell time-lapse images," *Nature Methods*, vol. 19, no. 6, pp. 770–770, Apr. 2022. DOI: `10.1038/s41592-022-01506-2`. [Online]. Available: `https://doi.org/10.1038/s41592-022-01506-2`.

[21] R. Mattson, D. Ott, V. Schneider, *et al.*, *Baseline models for representing mitochondrial dynamics*, Conference Poster, 2022.

[22] L. McInnes, J. Healy, N. Saul, and L. Grossberger, "Umap: Uniform manifold approximation and projection," *The Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018.

[23] K. Mohareer, J. Medikonda, G. R. Vadankula, and S. Banerjee, "Mycobacterial control of host mitochondria: Bioenergetic and metabolic changes shaping cell fate and infection outcome," *Frontiers in Cellular and Infection Microbiology*, vol. 10, Sep. 2020. DOI: `10.3389/fcimb.2020.00457`. [Online]. Available: `https://doi.org/10.3389/fcimb.2020.00457`.

[24] J. Nikolaisen, L. I. H. Nilsson, I. K. N. Pettersen, *et al.*, "Automated quantification and integrative analysis of 2d and 3d mitochondrial shape and network properties," *PLoS ONE*, vol. 9, no. 7, I. Georgakoudi, Ed., e101365, Jul. 2014. DOI: 10.1371/journal.pone.0101365. [Online]. Available: https://doi.org/10.1371/journal.pone.0101365.

[25] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[26] K. L. Patrick and R. O. Watson, "Mitochondria: Powering the innate immune response to mycobacterium tuberculosis infection," *Infection and Immunity*, vol. 89, no. 4, K. M. Ottemann, Ed., Mar. 2021. DOI: 10.1128/iai.00687-20. [Online]. Available: https://doi.org/10.1128/iai.00687-20.

[27] N. Pulagam, M. Hill, M. Fazli, *et al.*, "Classification of diffuse subcellular morphologies," in *Proceedings of the Python in Science Conference*, SciPy, 2021. DOI: 10.25080/majora-1b6fd038-00f. [Online]. Available: https://doi.org/10.25080/majora-1b6fd038-00f.

[28] D. E. Rumelhart and J. L. McClelland, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362.

[29] G. Sharir, A. Noy, and L. Zelnik-Manor, "An image is worth 16x16 words, what is a video worth?" *CoRR*, vol. abs/2103.13915, 2021. arXiv: 2103.13915. [Online]. Available: https://arxiv.org/abs/2103.13915.

[30] L. Shi, Q. Jiang, Y. Bushkin, S. Subbian, and S. Tyagi, "Biphasic dynamics of macrophage immunometabolism during imycobacterium tuberculosis/i infection," *mBio*, vol. 10, no. 2, D. A. Garsin, Ed., Apr. 2019. DOI: 10.1128/mbio.02550-18. [Online]. Available: https://doi.org/10.1128/mbio.02550-18.

[31] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Proceedings of the 28th International Conference*

on Neural Information Processing Systems - Volume 1, ser. NIPS'15, Montreal, Canada: MIT Press, 2015, pp. 802–810.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014, ISSN: 1532-4435.

[33] S. Suga, K. Nakamura, B. M. Humbel, H. Kawai, and Y. Hirabayashi, "An interactive deep learning-based approach reveals mitochondrial cristae topologies," Jun. 2021. DOI: `10.1101/2021.06.11.448083`. [Online]. Available: `https://doi.org/10.1101/2021.06.11.448083`.

[34] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018. DOI: `10.1109/cvpr.2018.00675`. [Online]. Available: `https://doi.org/10.1109/cvpr.2018.00675`.

[35] A. J. Valente, L. A. Maddalena, E. L. Robb, F. Moradi, and J. A. Stuart, "A simple ImageJ macro tool for analyzing mitochondrial network morphology in mammalian cell culture," *Acta Histochemica*, vol. 119, no. 3, pp. 315–326, Apr. 2017. DOI: `10.1016/j.acthis.2017.03.001`. [Online]. Available: `https://doi.org/10.1016/j.acthis.2017.03.001`.

[36] C.-Y. Wu, C. Feichtenhofer, H. Fan, K. He, P. Krahenbuhl, and R. Girshick, "Long-term feature banks for detailed video understanding," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019. DOI: `10.1109/cvpr.2019.00037`. [Online]. Available: `https://doi.org/10.1109/cvpr.2019.00037`.

[37] P. A. Yushkevich, J. Piven, H. C. Hazlett, *et al.*, "User-guided 3d active contour segmentation of anatomical structures: Significantly improved efficiency and reliability," *NeuroImage*, vol. 31, no. 3, pp. 1116–1128, Jul. 2006. DOI: `10.1016/j.neuroimage.2006.01.015`. [Online]. Available: `https://doi.org/10.1016/j.neuroimage.2006.01.015`.

[38] A. Zahedi, V. On, R. Phandthong, *et al.*, "Deep analysis of mitochondria and cell health using machine learning," *Scientific Reports*, vol. 8, no. 1, Nov. 2018. DOI: `10.1038/s41598-018-34455-y`. [Online]. Available: `https://doi.org/10.1038/s41598-018-34455-y`.