GENERATIVE SPATIOTEMPORAL MODELING OF NEUTROPHIL BEHAVIOR

by

NARITA PANDHE

(Under the direction of Shannon Quinn)

ABSTRACT

Cell motion and appearance have a strong correlation with cell cycle and disease progression. Many contemporary efforts in machine learning utilize spatiotemporal models to predict a cell's physical state and, consequently, the advancement of disease. Alternatively, generative models learn the underlying distribution of the data, creating holistic representations that can be used in inference. In this work, we propose an aggregate model that combine Generative Adversarial Networks (GANs) and Autoregressive (AR) models to predict cell motion and appearance in human neutrophils imaged by differential interference contrast microscopy. We bifurcate the task of learning cell statistics by leveraging GANs for the spatial component and AR models for the temporal component. The aggregate model learned results offer a promising computational environment for studying changes in organellar shape, quantity, and spatial distribution over large sequences.

INDEX WORDS: Biomedical Imaging, Deep Learning, Computer Vision, Generative Adversarial Networks (GANs), Autoregression

GENERATIVE SPATIOTEMPORAL MODELING OF NEUTROPHIL BEHAVIOR

by

NARITA PANDHE

B.E., University of Pune, 2012, India

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

2017

GENERATIVE SPATIOTEMPORAL MODELING OF NEUTROPHIL BEHAVIOR

by

NARITA PANDHE

Major Professor:	Shannon Quinn
Committee:	Tianming Liu
	Yi Hong

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
December 2017

DEDICATION

For Mom, Dad and Sagar.

ACKNOWLEDGMENTS

Firstly, I would like to extend heartfelt gratitude towards my advisor, Dr. Shannon Quinn for his patience, continuous guidance, encouragement, and insight. I gratefully acknowledge and thank him for extending to me, the resources from Faculty Research Grants (University of Georgia) and AWS Education Grants. I would also like to thank my committee members, for their constructive feedback and guidance. Many thanks to the members of the Quinn Research group for interesting and inspiring discussions. I would like to thank all my professors and the University of Georgia for making this a remarkable learning experience. Thanks to my family and friends for their support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Polymorphonuclear neutrophil granulocytes (neutrophils) are the most abundant white blood cells in most mammals. They are highly motile phagocytic cells that constitute the first line of defense of the innate immune system [Sehgal, 2005]. Neutrophils can be considered as surveillance systems that flow through the bloodstream, inspecting for infections in tissues or other inflammatory events.

Study of neutrophils and their underlying motion patterns provide insights into a host's response and behavior as a function of specific stimulus. To study their behavior and detect possible abnormalities, data is crucial. Our understanding of cell behavior and the sources of cellular variation can be further aided and validated by generating synthetic cells and simulating their behavior from the currently available, limited neutrophil data.

Generative models have been extensively applied to real-world natural images. Examples of models include Variational Autoencoders [Kingma and Welling, 2013], PixelCNNs [Oord et al., 2016] and Generative Adversarial Networks (GANs) [I. Goodfellow et al., 2014]. Generative models have the ability to learn distributions over data and can generate values, both, those that can be observed from the data and the ones that can only be computed from the observed ones. It can learn sophisticated conditional relationships as well.

Autoregressive (AR) models, traditionally used on temporal signals, are used for image processing (e.g. image and video texture completion and reconstruction) [Köppel et al., 2015]. An autoregressive model predicts future behavior based on the past behavior. It is used for forecasting when there is some correlation between values in a time series and the

values that precede and succeed them. Autoregressive models are powerful density estimators which have been successfully applied to dynamic texture synthesis ([Quinn et al., 2015], [Hyndman et al., 2007]).

Considering the limited dataset available, in this work, we propose to simulate the behavior of human neutrophils, using an aggregate application of Generative Adversarial Networks (GANs) and Autoregressive (AR) models. We bifurcate our approach as two tasks: generating neutrophils appearance and its motion, capturing the statistics independently. GAN learns the appearance and spatial statistics, while the AR model captures the temporal aspect.

1.1 Previous Work

Several computational methods have been proposed for constructing from image data, statistical models of cellular and subcellular structures. [Cootes et al., 1995] introduced Active Shape Models (ASM), methods to build flexible models of image structures whose shapes can vary. This method relies on a set of points used to present every object or image structure from the training set. Points can represent either boundary or internal features and have to be placed in the same way on each training sample. Based on these points, a point distribution model is derived, which gives the average shape and has parameters which control the main modes of variation. Given such a model and an image containing an example of the modeled object, ASM tries to find the best fit of the model to the image. Downside of this approach, it relies on an initial rough guess of the best shape, orientation, scale and position which is then refined by comparing the hypothesized model instance using differences between the model and image to deform the shape. This technique has been widely used to analyze images of faces, mechanical assemblies and medical images (in both 2D and 3D) [Song et al., 2012]. Based on the trained parameters of ASM ([Song et al., 2012], [Agianpuye and Minoi, 2014]) synthesized 2D shape of tongue and 3D face shape respectively. [Zhao and Murphy, 2007] constructed nested set of conditional models based on 2D

images. They used a medial-axis model to represent a cell's nucleus shape and a texture model to represent DNA distribution within that cell's shape. Using the nucleus as a starting point, the final cell shape model is generated. Combining the medial axis shape model with the texture model gave the ability to synthesize nuclear images with proper shape and approximate chromatin texture. [Buck et al., 2012] built parametric and nonparametric models for the study of subcellular organization and protein patterns in 2D and 3D images. Based on the approach of [Zhao and Murphy, 2007], cell's nucleus shape is modeled and then cell shape is modelled as statistically dependent on the corresponding nuclei's shape. Recently, [A. Osokin and Vaggi, 2017] applied GANs to fluorescence microscopy images of cells from the LIN [Dodgson et al., 2017] dataset. Each image consisted of two independent fluorescence imaging channels (red and green) corresponding to the two different proteins tagged with red or green-emitting fluorophores, respectively. Their aim was to synthesize biological images, investigate co-localization of proteins and study the dynamical changes in cellular localization that proteins undergo through time as cells grow and divide.

[Buck et al., 2012] also briefly discussed an approach for modeling cellular and nuclear shapes to consider temporal evolution. To that end, they used nonparametric shape models to produce a random walk based simulation. Random Walk is a type of autoregressive model. AR modeling is a parametric technique, which models every point in sequence as a linear combination of d previous points. The number of lagged values, known as the order, affects the fit of the model and also increases the required amount of training data [Oziem et al., 2004]. AR model of order d can be formulated as (Eq. 1.1) where $\overrightarrow{v_t}$ is white noise and coefficients $B = B_1, B_2, ..., B_d$ define the transition from state to state. An AR model of order 1 (AR(1)) can be modeled as (Eq. 1.2). Random Walk has the same form as AR(1) but with $B_1 = 1$.

$$\overrightarrow{x_t} = B_1 \overrightarrow{x}_{t-1} + B_2 \overrightarrow{x}_{t-2} + .. + B_d \overrightarrow{x}_{t-d} + \overrightarrow{v_t} \qquad \text{(Eq. 1.1)}$$

$$\overrightarrow{x_t} = B_1 \overrightarrow{x}_{t-1} + \overrightarrow{v_t} \quad \text{(Eq. 1.2)}$$

In ([Oziem et al., 2004], [Campbell et al., 2004], [Arikan et al., 2003]) AR processes synthesize video sequences based on the existing ones. [Arikan et al., 2003] presents an algorithm that synthesizes motions based on annotations that describe it. Motion is constructed by splitting segments of movement from a corpus of motion data and assembling them. Each segment is modeled using an autoregressive process. This helps in modeling complicated non-stationary sequences which a single autoregressive process cannot handle. In [Oziem et al., 2004], frames of original videos are projected into low-dimensional space and then an AR model is learned. [Campbell et al., 2004] extends this approach by overcoming the problems of non-linearities in the data either using a spline-fitting approach or a combined appearance model.

1.2 Objectives and Restrictions

Given the relevant background, the objectives of this thesis are as follows:

- To review the related literature and the solutions proposed for cell segmentation and reconstruction.
- To apply semantic segmentation techniques to extract neutrophils from a stack of RGB, two-dimensional medical images.
- Build generative models based on the segmented cells to generate synthetic neutrophil images.
- To reveal the issues and solutions during the training of the deep generative models for avoiding possible model collapses.

- Build a weak motion model to synthesize neutrophil motion. Towards this goal we use time-series based autoregressive models.
- Use appearance and content synthesized from generative models and motion synthesized from autoregressive model to realistically simulate a biological system.

In the scope of the thesis, considering the time and resources allocated, the restrictions are specified as follows:

- We are dealing with videos imaging the two dimensional motion of human neutrophilic granulocytes. Often cells overlap each other or some of the cells appear cluttered or some of them are deformed. We do not consider cells that appear fluorescent because they are overlapped by others or cells that are highly deformed.
- Since neutrophils are lobed white blood cells, we consider only those cells which are mostly circular.
- Due to the limited computing power, only a few convolutional neural network design approaches are investigated to generate neutrophil images.

1.3 Thesis Outline

This thesis is organized as follows:

Chapter 2 gives an overview about the related background. It discusses deep learning based medical image segmentation techniques that have been previously used. It also provides details about generative models utilized for creating synthetic images and autoregression - time-series models for synthesizing sequences.

Chapter 3 presents the details about the proposed system.

Chapter 4 analyzes all the results and discusses the strategies used for thorough evaluation.

Chapter 5 Discusses about future improvements and concludes this study.

CHAPTER 2

BACKGROUND

2.1 Segmentation

Segmentation is the task of separating an image into distinct parts (each part represents a class). Distinct parts can be identified by differences between colors, textures or other image features. Semantic segmentation is where the same label is assigned to semantically identical objects. In the case of neutrophil images, segmentation refers to the construction of a map where each pixel is marked with a label that corresponds to either of the following classes: neutrophil cell itself, cell border and background. To eventually train generative models, this work employs semantic segmentation - a precursor, to automatically extract individual cells from videos. This section discusses state-of-the-art neural network based architectures for semantic segmentation that best fits the nature of the problem, achieving the best possible performance.

Fully Convolutional Neural Networks (FCNs)

Introduced by [Long et al., 2014], the primary goal of fully convolutional neural networks (FCNs) is to predict semantic segmentation maps. Given an input image, the predicted output is an image that is same in size as the input, but in which every pixel is classified to belong to one of the C classes (C is number of classes we are segmenting the image into). [Long et al., 2014] takes in images from the Pascal VOC [Everingham et al., 2010] dataset as input and outputs a 21 - class semantic segmentation map of the corresponding images. The basic idea of fully convolutional network is that it is "fully convolutional", meaning

that all layers are convolutional layers and they do not have any fully connected layers. This enables the network to read input images of arbitrary size and predict output maps that are of same height and width as the input images, but number of channels is equivalent to the number of classes. In the intermediate layers, input tensors[1] typically get smaller because of operations like pooling reduce the height and width of the tensors. FCNs use deconvolutions - backward convolutions to upsample the outputs of intermediate tensors so that the final output can be of same dimensions as the input. The coarse segmentation map acquired through deconvolution is made finer by combining it with segmentation maps computed from earlier stages in the network [Kayalibay et al., 2017].

U-Net

[Ronneberger et al., 2015] attempted the task of biomedical image segmentation with better results, using the fully convolutional approach of [Long et al., 2014] as discussed in Section 2.1. They follow a design that is similar to that of a convolutional autoencoder: the network consists of a downsampling and upsampling path resembling shape of the english letter U. In the downsampling path, features from the input images are extracted. These features then serve as input to the upsampling path to predict the segmentation mask, which has the same height and width as the original input image. To compensate for the loss of spatial information during downsampling, authors propose skip connections between the downsampling and upsampling paths. Skip connections help the upsampling path to recover fine-grained information. This network won the EM segmentation challenge at ISBI 2012.

[1] a mathematical object analogous to but more general than a vector, represented by an array of components that are functions of the coordinates of a space. Tensors are high dimensional generalizations of matrices.

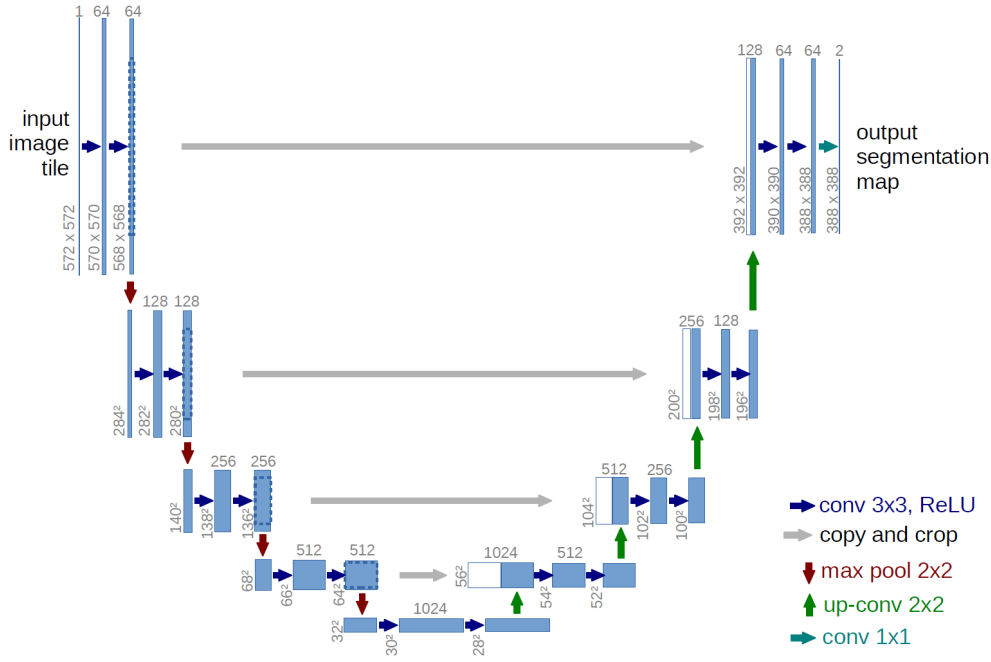


Figure. 2.1: Architecture of U-Net, fully convolutional neural network which performed biomedical image segmentation [Ronneberger et al., 2015]. Number of feature maps is denoted on top of every box. Width - Height dimensions of the input images are denoted at the lower left edge of the every box. White boxes represent concatenated feature maps.

DenseNets

[Huang et al., 2017] incorporates the observation that convolutional neural networks can be substantially deeper, accurate and efficient to train if they contain shorter connections between between layers close to the input and those close to the output. It introduces Dense Convolutional Networks (DenseNets) that connects every layer to every other layer in a feed-forward fashion. Every layer uses feature-maps of all the preceding layers as input and its own feature-maps as inputs to all the subsequent layers. The feature-maps are stacked/concatenated, unlike ResNets [He et al., 2015a] where the feature-maps are added with the inputs. At every layer, DenseNet adds only a small set of feature-maps to the network's collective knowledge and remaining feature maps are unchanged. The final classifier

makes a decision based on all feature maps in the network. Feature map concatenation provides the advantage of improved information and gradient flow, making it easier to train the network. Each layer has direct access to the gradients from the loss function and the original input signal, leading to an implicit deep supervision [Huang et al., 2017]. This helps in training of deeper network architectures and alleviating the problem of vanishing-gradient. Dense connections have a regularizing effect, which reduces overfitting and thus makes it suitable for training in situations where the dataset size is small. Other advantages of this architecture are: number of parameters is substantially reduced as there is no need to relearn the redundant feature maps, feature propagation is strengthened because each layer reads the states from its previous layers and passes its own to the subsequent layers. This helps in explicitly notifying about the state changes that need to be preserved.

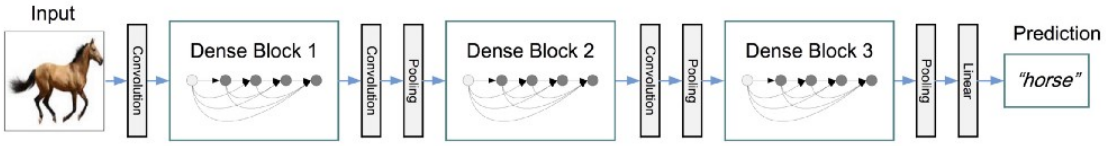
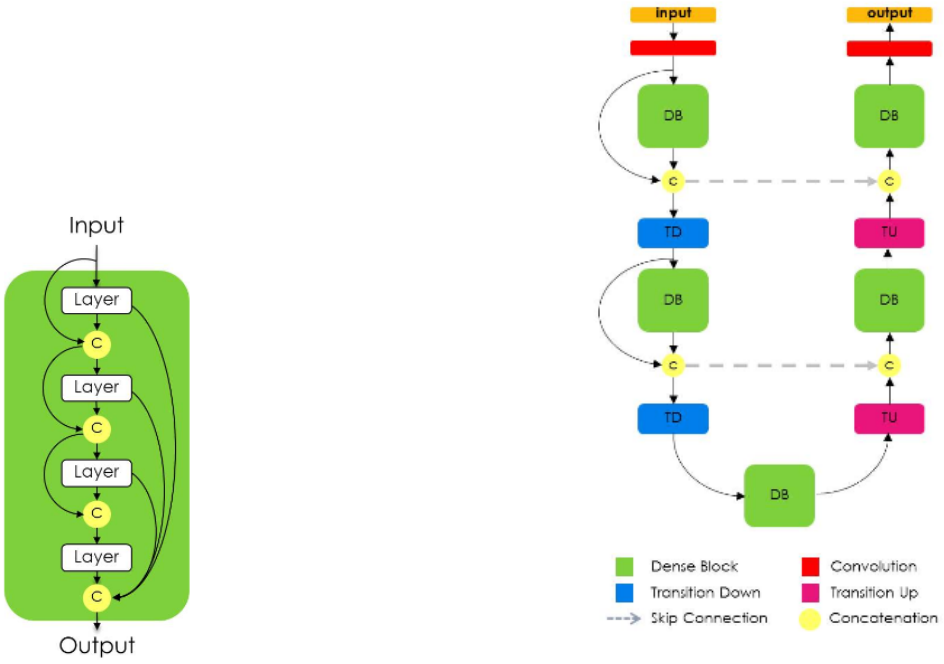


Figure. 2.2: DenseNet architecture with three dense blocks [Huang et al., 2017]. Feature maps of layers in a block are concatenated, which gives an advantage of improved information and gradient flow.

Fully Convolutional DenseNet

[Jégou et al., 2016] extend DenseNets to work as FCNs by adding an upsampling path to recover the full input resolution. Instead of naively building an upsampling path, to avoid feature explosion, authors propose to only upsample the feature maps created by the preceding dense block. The higher resolution information is passed by means of standard skip connections between the downsampling and the upsampling paths. Fig. 2.3a depicts the components of a dense block in which the first layer is applied to the input to create k feature maps, which are concatenated to the input. A second layer is then applied to create another k features maps, which are again concatenated to the previous feature maps. The

operation is repeated 4 times. The output of the block is the concatenation of the outputs of the 4 layers, and thus contains $4 * k$ feature maps. Fig. 2.3b depicts the architecture used for semantic segmentation by [Jégou et al., 2016]. It consists of dense blocks, downsampling and upsampling path. Circles represent concatenation of feature maps and arrows represent the connectivity pattern. Gray horizontal arrows represent skip connections. Note, in the upsampling path, dense blocks are not concatenated with their outputs.



(a) Dense Block

(b) FCN DenseNet Architecture.

Figure. 2.3: FCN DenseNet Architecture [Jégou et al., 2016]. It combines fully convolutional networks and DenseNets by adding upsampling path to DenseNet.

2.2 Generative Models

There are several ways to categorize machine learning models. Some of the categories are: supervised vs unsupervised, regression vs classification and discriminative vs generative. Discriminative models learn the conditional probability $P(y|x)$. During classification, they try to find a line or hyperplane that separates the samples based on their class.

Generative models aim at modeling how the data is generated. They learn the joint probability of the input data and labels simultaneously, i.e. $P(x, y)$. This lends them the ability to synthesize samples based on the distribution of input data. Generative models have the potential to understand and explain the underlying structure of the input data even when there are no labels. Recently, generative models like Variational Autoencoders [Kingma and Welling, 2013], PixelCNNs [Oord et al., 2016] and Generative Adversarial Networks (GANs) [I. Goodfellow et al., 2014] have been extensively utilized for natural images. In this thesis, synthetic neutrophil images are generated by applying GANs. Following sub-sections cover its details.

Generative Adversarial Networks (GANs)

In 2014, [I. Goodfellow et al., 2014] proposed Generative Adversarial Networks(GANs), an adversarial framework for estimating generative models. GANs are a family of successful models, that do not rely on training objectives related to log-likelihood, instead they are analogous to game theory. GANs consists of 2 neural networks competing against each other: Generator (G) and Discriminator (D). G generates images from random noise. While doing so, it tries to get as close as it can, to the distribution of real images. D classifies between the real images and fake images generated by G. Both are trying to perform best at their respective tasks and maximize their gains. D is characterized as adversarial loss for training G. GANs were originally applied to the MNIST [LeCun and Cortes, 2010] dataset.

Formally, consider a set of training images, $x \in X_{real}$ coming from a real distribution P_d. The generator is a neural network $G(z, \theta G)$ parameterized by θG and discriminator is a neural network $D(x, \theta D)$ parameterized by θD. $G(z, \theta G)$ takes in random noise z from the distribution P_z and generates images $x \in X_{fake}$. $D(x, \theta D)$ takes images from $x \in X_{real}$ and $x \in X_{fake}$, both, and outputs a scalar between [0, 1]. The output is higher if the sample belongs to X_{real} else X_{fake}. Both G and D are trained simultaneously. The goal of

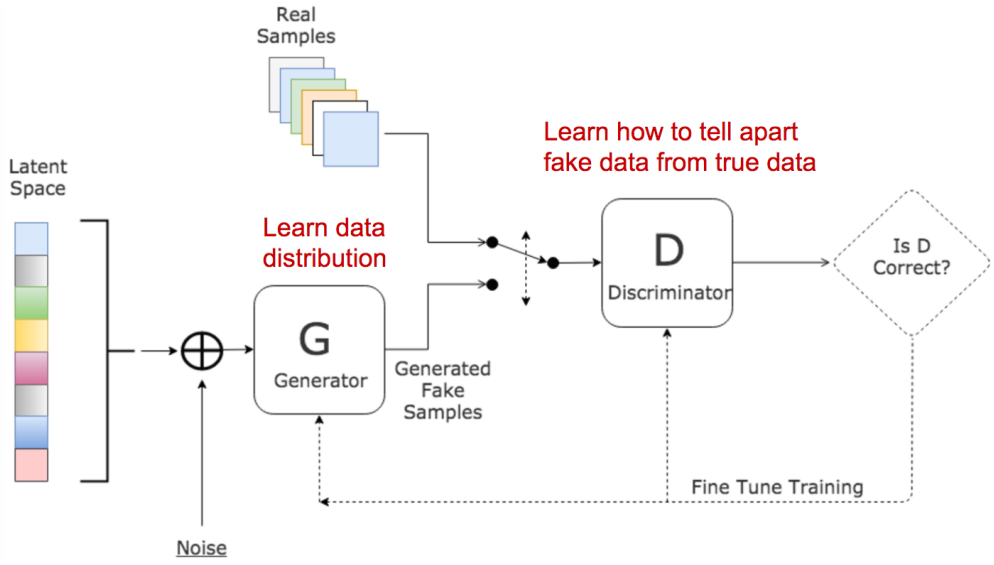


Figure. 2.4: Adversarial framework of GANs [Gharakhanian, 2017]. It consists of two neural networks: Generator G and Discriminator D, competing against each other. G tries to fool D by generating plausible images of the training set and D tries to distinguish between real training set images and fake images produced by G.

D is to maximize the probability of assigning correct labels to an input while G minimizes $\log\left(1-D(G(z)\right)$. As a result D and G can be seen as playing a minimax game, as formulated in (Eq. 2.1).

$$\min_{G}\max_{D} V(D,G) = \mathbb{E}_{x\sim p_{data(x)}}\big[\log D(x)\big] + \mathbb{E}_{z\sim p_{z(z)}}\big[\log 1 - D(G(z))\big] \qquad \text{(Eq. 2.1)}$$

Algorithm 1 Vanilla GAN algorithm. Minibatch SGD training. No. of times the discriminator is trained before generator is trained, is k. The least expensive option is to train discriminator once and then train the generator, so $k = 1$.

for *number of total training iterations* **do**
  for k *steps* **do**
    Sample minibatch of m noise samples $z^{(1)}, ..z^{(m)}$ from noise prior $p_g(z)$.
    Sample minibatch of m examples $x^{(1)}, ..x^{(m)}$ from data generating distribution $p_{data}(x)$.
    Update the discriminator by ascending its stochastic gradient.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [\log D(x^{(i)}) + \log (1 - D(G(z^{(i)})))] \tag{2.2}$$

  end
  Sample minibatch of m noise samples $z^{(1)}, ..z^{(m)}$ from noise prior $p_g(z)$.
  Update the generator by descending its stochastic gradient.

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log (1 - D(G(z^{(i)}))) \tag{2.3}$$

end
The gradient-based updates can use any standard gradient-based learning rule.

Deep Convolutional GANs (DCGANs)

Historical attempts to scale up GANs using CNNs to model images have been generally unsuccessful [I. Goodfellow et al., 2014]. Deep Convolutional GAN (DCGANs) [Radford et al., 2015] identified a family of architectures that resulted in stable training and can generate higher resolution images. The paper suggested following changes to Vanilla GANs:

- Replace any pooling layers with strided convolutions (for D) and fractional strided convolutions (for G).
- Remove fully connected hidden layers.
- Use batch normalisation in both G (all layers except output layer) and D (all layers except input layer).

- Use LeakyReLU in all layers of the D.
- Use ReLU activation in all layers of the G (except output layer which uses Tanh).

Wasserstein GAN (WGAN)

$$JS(P_r, P_g) = \frac{1}{2}KL(P_r||P_m) + \frac{1}{2}KL(P_g||P_m) \quad \text{(Eq. 2.4)}$$

The basis of each GAN algorithm is to measure the distance between the real and fake probability distributions and alter the network parameters of G. [I. Goodfellow et al., 2014] showed that the minimax formulation (Eq. 2.1) can be reformulated via minimization of the Jensen Shannon (JS) divergence (Eq. 2.4), between the data-generating distribution P_d and the distribution P_g induced by P_z and G. [Arjovsky et al., 2017] theoretically justified the JS minimized by GANs behaves badly(e.g. when the two distributions do not overlap) and is potentially not continuous w.r.t to the generator's parameters. This leads to training difficulty as it cannot provide any useful direction. They propose using an alternative distance - Earth Mover's distance short for EM formulated as (Eq. 2.5).

$$W(P_r, P_g) = \inf_{\gamma \in \prod(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|] \quad \text{(Eq. 2.5)}$$

$$W(P_r, P_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{(x)\sim P_r}[f(x)] - \mathbb{E}_{(x)\sim P_\theta}[f(x)] \quad \text{(Eq. 2.6)}$$

EM is also known as Wasserstein Distance, $W(q, p)$, can be informally defined as: minimum cost of transferring mass in order to transform one distribution q to another distribution p. The cost is calculated by the amount of mass moved times the moving distance. Unfortunately, computing the Wasserstein distance is intractable. [Arjovsky et al., 2017] showed an approximate solution to the same using Kantorovich-Rubinstein duality to obtain (Eq. 2.6) where D is the set of 1-Lipschitz functions and P_g is once again the model distribution implicitly defined by $\tilde{x} = G(z), z \sim p(z)$ [Gulrajani et al., 2017]. In that case, under an

optimal discriminator (called a critic in the paper, since it's not trained to classify), minimizing the value function with respect to the generator parameters minimizes $W(P_r, P_g)$. To enforce the Lipschitz constraint on the critic, [Arjovsky et al., 2017] proposes to clip the weights of the critic to lie within a compact space $[-c, c]$. (Eq. 2.7) represents the WGAN value function whose gradients w.r.t. its input is better behaved in comparison to Vanilla GAN's value function, making optimization of the generator easier. Additionally, WGAN has the desirable property that its value function correlates with sample quality, which is not the case for Vanilla GANs [Arjovsky et al., 2017].

$$\min_G \max_{D \in D} \mathbb{E}_{x \sim P_r}[D(x)] - \mathbb{E}_{\tilde{x} \sim P_g}[D(\tilde{x}))] \qquad \text{(Eq. 2.7)}$$

Algorithm 2 WGAN algorithm. Learning rate $\alpha = 0.00005$, weights clipped to $c = 0.01$, minibatch size $m = 64$, no. of iterations discriminator D is trained for $n_{critic} = 5$ before training the generator G. In comparison to Vanilla GAN training algorithm, there are no logarithms and after every gradient update on critic, the weights are clamped between $[-c, c]$.

w_0, initial critic parameters
θ_0, initial generator's parameters.
while θ *has not converged* **do**
  for *t= 0, ..., n_{critic}* **do**
    Sample $x_{(i)}{}_{i=1}^{m} \sim \mathbb{P}_r$ *a batch from real data.*
    Sample $z_{(i)}{}_{i=1}^{m} \sim p(z)$ *a batch of prior samples.*
    $g_w = \nabla_w[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))]$
    $w = w + \alpha \cdot RMSProp(w, g_w)$
    w = clip(w,-c,c)
  end
  Sample $\{z_{(i)}\}_{i=1}^{m} \sim p(z)$ *a batch of prior samples.*
  $g_\theta = -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta = \theta - \alpha \cdot RMSProp(\theta, g_\theta)$
end

Improved WGAN

Recently, [Gulrajani et al., 2017] discussed optimization problems due to weight clipping in WGAN and recommended an alternative way to enforce the Lipschitz constraint. Imple-

menting a k-Lipschitz constraint via weight clipping biases the critic towards much simpler functions [Gulrajani et al., 2017]. They also demonstrate the difficulty in WGAN optimization process is because of interactions between the weight constraint and the cost function, which inevitably result in either vanishing or exploding gradients, depending on the value of the clipping threshold c. So, instead of weight clipping, authors advocate to penalize the norm of the critic's gradient with respect to its input. The objective function (Eq. 2.8) leads to stable training and enables training of a wide variety of GAN architectures with almost no hyperparameter tuning.

$$L = \underbrace{\mathbb{E}_{\widetilde{x}\sim\mathbb{P}_g}\big[D(\widetilde{x})\big] - \mathbb{E}_{x\sim\mathbb{P}_r}\big[D(x)\big]}_{\text{Original Critic Loss}} + \underbrace{\lambda\,\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\big[(\|\nabla D(\hat{x})\|_2 - 1)^2\big]}_{\text{Gradient Penalty}} \qquad \text{(Eq. 2.8)}$$

Algorithm 3 Improved WGAN algorithm with gradient penalty. Default values no. of iterations discriminator D is trained for $n_{critic} = 5$ before training the generator G, learning rate $\alpha = 0.0001$, $\lambda = 10$, $\beta_1 = 0$, $\beta_2 = 0.9$. In comparison to Vanilla GAN training algorithm, there are no logarithms and after every gradient update on critic, the weights are clamped between $[-c, c]$.

w_0, initial critic parameters
θ_0, initial generator's parameters.
while θ *has not converged* **do**
  for *t=1, ..., n_{critic}* **do**
    for *i=1,...,m* **do**
      Sample real data $x \sim \mathbb{P}_r$, latent variable $z \sim p(z)$, a random number $\epsilon \sim U[0,1]$.
      $\tilde{x} = G_\theta(z)$
      $\hat{x} = \epsilon x + (1-\epsilon)\tilde{x}$
      $L^{(i)} = D_w(\tilde{x}) - D_w(x) + \lambda(\|\nabla_{\hat{x}} \nabla D(\hat{x})\|_2 - 1)^2$
    end
    $w = Adam(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end
  Sample a batch of latent variables $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$.
  $\theta = Adam(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(z)), \theta, \alpha, \beta_1, \beta_2)$
end

2.3 Autoregression

For computer vision systems, motion synthesis is still a challenging task and is drawing more contemporary research attention. Synthesis can be defined as generating new versions of a dataset which follow the original distribution and can be closely related to modeling. One of the targets of our research is synthesizing new video clips based on existing ones. Such a technique has great appeal since, once a short video is obtained, new sequences (potentially infinitely long) can be synthesized based upon it [Campbell et al., 2004]. AR processes have been used in tracking and for synthesizing video textures (a temporal texture) [Oziem et al., 2004]. It's a parametric modeling technique, which models every point in sequence as a linear combination of d previous points. The number of lagged values, known as the order, affects the fit of the model and also increases the required amount of training data [Oziem et al., 2004]. It is likely that an AR process will produce a good model if the dataset follows Gaussian distribution.

An autoregressive model consists of two components: appearance and dynamic. The state of the system is determined by the appearance component and dynamic component captures how the state change over time. An AR process for a series of points in a n-dimensional space can be modelled as (Eq. 2.9, Eq. 2.10). (Eq. 2.9) decomposes each video frame $\overrightarrow{y_t}$ into a low-dimensional state vector $\overrightarrow{x_t}$ and a white noise term $\overrightarrow{u_t}$ and (Eq. 2.10) denotes new state $\overrightarrow{x_t}$ in a low-dimensional subspace defined by an orthogonal basis C at time t. It also represents the new state $\overrightarrow{x_t}$ is a function of the sum of d of its previous states $\overrightarrow{x}_{t-1}$, $\overrightarrow{x}_{t-2}$, .., $\overrightarrow{x}_{t-d}$ each multiplied by corresponding coefficients $B = B_1, B_2, ..., B_d$ [Quinn et al., 2015]. Matrices $B = B_1, B_2, ..., B_d$ define the transition from state to state. The noise terms u and v represent the residual difference between the observed data and the solutions to the linear equations, assumed to be Gaussian White noise.

$$\overrightarrow{y_t} = C\overrightarrow{x_t} + \overrightarrow{u_t} \qquad \text{(Eq. 2.9)}$$

$$\overrightarrow{x_t} = B_1 \overrightarrow{x}_{t-1} + B_2 \overrightarrow{x}_{t-2} + .. + B_d \overrightarrow{x}_{t-d} + \overrightarrow{v_t} \quad \text{(Eq. 2.10)}$$

Appearance Component

Videos are represented as trajectories of individual objects consisting of its center Cartesian coordinates across all the frames. Multiple object's trajectory can be represented as (m x n x 2) i.e. x, y center coordinates of m objects over n frames. Processing this high dimensional data can be computationally expensive. Dimensionality reduction enables us to handle huge amounts of data. The low-dimensional representation not only allows to use more data, but also helps in retaining the relevant information concisely and ignoring the uninformative redundancies. (Eq. 2.9) decomposes trajectories of each video frame $\overrightarrow{y}_t$ into a low-dimensional vector $\overrightarrow{x}_t$ and white noise term $\overrightarrow{u}_t$, using an orthogonal basis C. Principal Component Analysis (PCA) can be used to define this low-dimensional space. PCA exploits the redundancy in data by rotating the image space with orthogonal axes along the directions of highest variance. C can be derived using singular value decomposition (SVD), which is an efficient approach for extracting the principal components ([Quinn et al., 2015], [Hyndman et al., 2007]). Trajectories of objects consisting of center coordinates over n frames serve as input to SVD. Therefore, if the number of objects were given by m, and the number of frames as n, the dimensions of the input matrix Y^T would be (γ x n), where γ is (m x 2), depicting the aggregate motion of all the m objects. The singular value decomposition of the trajectories is $Y^T = U \sum V^T$. Matrix U is a (γ x n) matrix, matrix $\sum$ is (n,) and V is (n x n). Matrix U contains the principal components. $C \equiv \hat{U}$, where $\hat{U}$ is a matrix containing first q columns on U. $\hat{V}$ is composed of first q columns of V, and $\hat{\sum}$ is a diagonal matrix consisting the largest q singular values in matrix $\sum$. The low-dimensional subspace matrix $X^T \equiv \hat{\sum} \hat{V}_T$ is a (q x n) matrix, where q determines the dimensionality of the subspace C.

Dynamic Component

The temporal aspect of trajectories changing over time is captured by the dynamic component of AR as formulated in (Eq. 2.10). It is modeled using deterministic and stochastic components. The deterministic component described by (Eq. 2.11) is a linear Markov model [Hyndman et al., 2007]. It basically implies that linear combination of past states can be used to generate the current state. Information that is not captured by the linear model is modeled by the stochastic component $\overrightarrow{v_t}$. Once the low-dimensional subspace, X and C are learned using the appearance component as discussed in Sec. 2.3, the goal is to determine the transition matrix B which best describes the transitions between states and minimizes the total squared error between the true states and approximated states. Eq. 2.12 can be used to solve for transition matrices, where $D^{\diamond} = D^T(DD^T)^{-1}$

$$\overrightarrow{x_t} = B_1 \overrightarrow{x}_{t-1} + B_2 \overrightarrow{x}_{t-2} + .. + B_d \overrightarrow{x}_{t-d} \qquad \text{(Eq. 2.11)}$$

$$A = X_2^T (X_1^{T-1})^{\diamond} \qquad \text{(Eq. 2.12)}$$

CHAPTER 3

PROPOSED SOLUTION

3.1 System Overview

In this chapter the overall system architecture is introduced. For each subsystem a short description of its purpose is given which serves as ground work for the detailed explanations in the following sections. The system can be seen in Fig. 3.1.

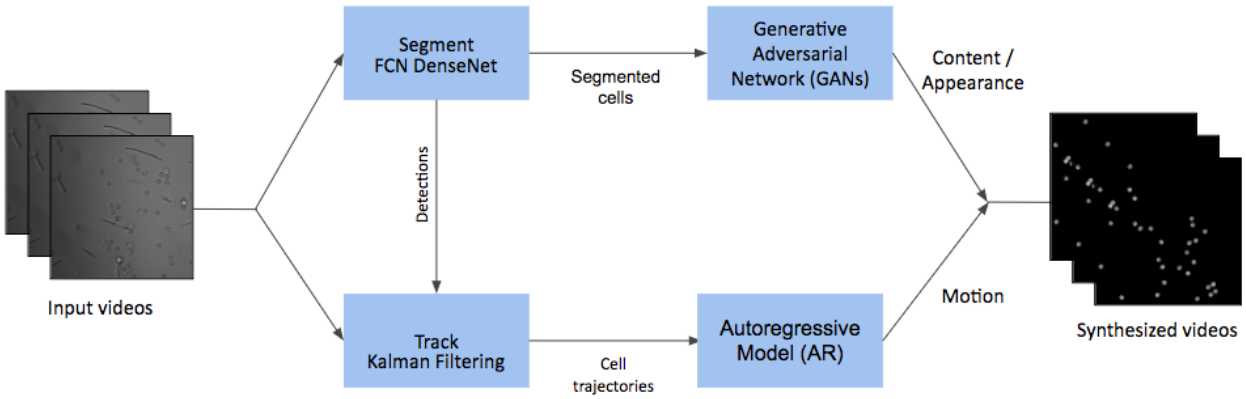


Figure. 3.1: Overview of the system. It consists of four components - Segment, Track, GANs and AR. Segment and Track components extract the input data in relevant format in order to train the generative components - GANs and AR. GANs learn the appearance and spatial statistics, while AR learns the motion and temporal statistics.

The first component - Segment, is based on fully convolutional neural networks (CNN). It reads the input images and generates the segmentation maps. Based on these segmentation maps, the component - Track, extracts trajectories of the segmented objects. The function of these two blocks (Segment and Track) is to extract the input data into a usable form so as to train the next two blocks. Segmented objects serve as input to train the block - Generative Adversarial Network, which is responsible to create synthetic images. Trajectories obtained

from tracking are fed to the block - Autoregressive models, which learns the object movements and is used to synthesize motion.

3.2 Dataset

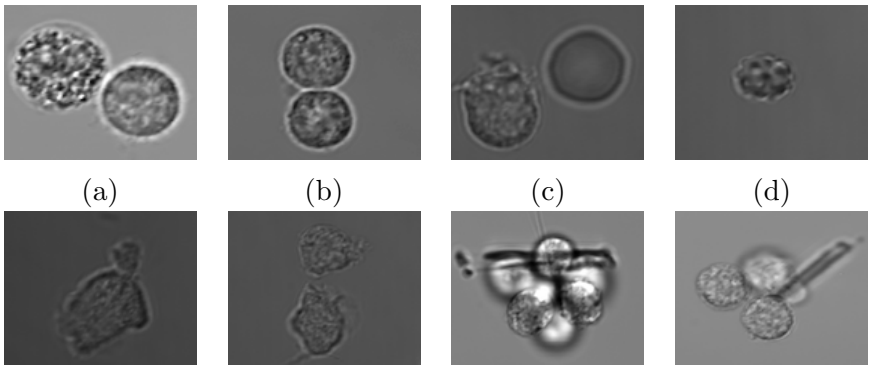


Figure. 3.2: Sample neutrophil images. Fig. 3.2a, Fig. 3.2b depicts lobed neutrophils with texture. (left) object from Fig. 3.2c denotes a neutrophil, while (right) object is not a neutrophil. Fig. 3.2d is not a neutrophil. Fig. 3.2e display deformed neutrophils. Fig. 3.2g displays fluorescent neutrophils. Because they are overlapped by other neutrophils the underlying neutrophils appear bright.

Videos imaging the two dimensional motion of human neutrophilic granulocytes were provided by Dr. Balazs Rada (Department of Infectious Diseases, University of Georgia). The videos were recorded using DIC microscopy which enhances the contrast in unstained, transparent samples. The dataset consisted of 11 videos, including 3 videos of normal neutrophils and 8 videos of neutrophils treated with an inhibitor, MRS2578, targeting a purinergic receptor. The average length of neutrophil videos was 3.0secs. Using ffmpeg individual frames were extracted at the rate of 20fps. Extracted frames were of size (1024 x 1024). All the frames were converted to grayscale and saved separately as PNG files.

Ground Truth Segmentation Maps

To synthesize individual cells, neutrophils had to be segmented from the video frames. Unfortunately, annotations to identify neutrophils were not provided. As stated in Section 1.2 neutrophils are mostly circular. The cell population varied in the range of 25 - 100 cells per frame, and cells freely enter/exit the field of view. As we lacked pathological training and depended on a few guidelines, some cells were hard to discern either because they were overlapping, tightly cluttered or underwent significant appearance variation. Ultimately it was deemed that a consistent segmentation could not be reached. These issues are, listed in Fig. 3.2.

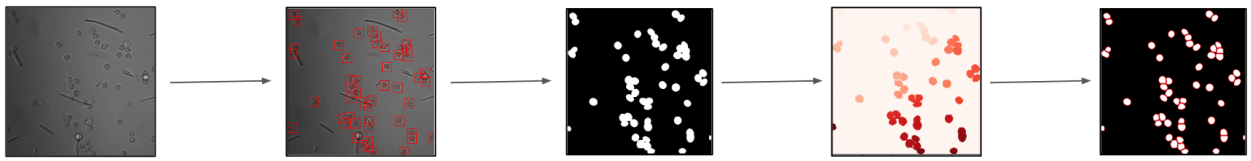
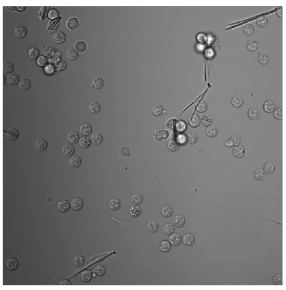
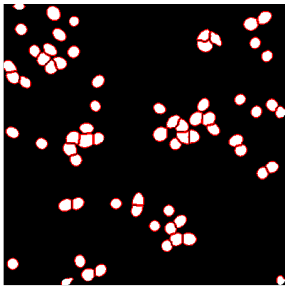


Figure. 3.3: Segmentation Pipeline: 1) Given an input image, we manually marked the centers of every cell. 2) A (64 x 64) patch was extracted surrounding every center. 3) An ellipse was fit to every box followed by morphological erosion which produced the binary segmentation maps. 4) To separate cluttered cells, marker based Watershed segmentation was applied.

To segment neutrophils, we manually marked the centers of every cell in all the frames using bounding box annotation tool [Yen, 2017]. A (64 x 64) patch was extracted around the centers of each cell. Every patch was then thresholded using adaptive thresholding to output a binary image. An ellipse was fit to every binary patch followed by morphological erosion. Subsequently, binary segmentation maps were produced. Cluttered cells appeared as big blobs in binary maps. There were no cell borders separating each cell instance. As a workaround, we applied marker based Watershed segmentation [Beucher and Meyer,] and added a two-pixel-wide border around each cell. This effectively created a new class: neutrophil border apart from neutrophils and background. The segmentation pipeline is depicted in Fig. 3.3 and the final segmentation map corresponding to an image can be seen in Fig. 3.4.



(a) Sample input frame extracted from video.



(b) Segmentation map

Figure. 3.4: Input image and its corresponding segmentation map produced by the procedure discussed in Section. 3.2

3.3 Experiments

This section presents the details of each block of the system from Fig. 3.1.

Segmentation

To automate the process of segmenting neutrophils, we utilize FC-DenseNet-103 architecture as described in [Jégou et al., 2016]. Fig. 3.5a defines the building blocks of the network. The output dimension of each layer has k feature maps where k is referred to as the growth rate parameter and is set to $k = 16$. Fig. 3.5b depicts all the layers of the network. This architecture consists a total of 103 convolutional layers: 38 layers in downsampling path, 38 in upsampling path, one layer at the input and output, 15 layers in the bottleneck, one convolution layer in each of the 5 transition up and transition down blocks.

A total of 412 grayscale frames/images were extracted from 8 videos of Neutrophils which consisted of normal and inhibited cells, both. Groundtruth segmentation maps for 3 classes - neutrophil, neutrophil border and background were obtained using the procedure described in Section. 3.2. All the images were standardized using mean subtraction, normalized by

Layer
Batch Normalization
ReLU
3×3 Convolution
Dropout $p = 0.2$

Transition Down (TD)
Batch Normalization
ReLU
1×1 Convolution
Dropout $p = 0.2$
2×2 Max Pooling

Transition Up (TU)
3×3 Transposed Convolution $stride = 2$

(a) Building blocks of FC-DenseNet

Architecture
Input, $m = 3$
3×3 Convolution, $m = 48$
DB (4 layers) + TD, $m = 112$
DB (5 layers) + TD, $m = 192$
DB (7 layers) + TD, $m = 304$
DB (10 layers) + TD, $m = 464$
DB (12 layers) + TD, $m = 656$
DB (15 layers), $m = 880$
TU + DB (12 layers), $m = 1072$
TU + DB (10 layers), $m = 800$
TU + DB (7 layers), $m = 560$
TU + DB (5 layers), $m = 368$
TU + DB (4 layers), $m = 256$
1×1 Convolution, $m = c$
Softmax

(b) Architecture of FC-DenseNet103

Figure. 3.5: FC-DenseNet103 Architecture used for segmentation. It consists of 103 convolutional layers. Following are the notations: DB stands for Dense Block, TD stands for Transition Down, TU stands for Transition Up, BN stands for Batch Normalization and m corresponds to the total number of feature maps at the end of a block. c stands for the number of classes [Jégou et al., 2016].

dividing by the standard deviation and resized to (224 x 224). This preprocessing technique helps avoid issues caused by poor contrast images. To make the most of our few training examples and increase the accuracy of the model, the dataset was augmented via random vertical and horizontal flips. This also helps to prevent overfitting and improve the model's ability to generalize. All frames of the videos were considered as cells enter/exit the field of view. We set aside 46 of the 412 images as a holdout test set and randomly choose samples from the remaining 366 for training and validation using a 90/10% train-validation split.

We do not use any pre-trained models and initialize our model using HeUniform [He et al., 2015b]. RMSprop [Tieleman and Hinton, 2012], with an initial learning rate of $1e-3$ is used. We regularized our models with a weight decay of $1e-4$ and a dropout rate of 0.2. The model is trained by minimizing the weighted pixelwise cross-entropy loss. The cell body and cell borders separating touching cells, obtain a large weight in the loss function in comparison to the background class. The pixels from background class are weighted by inverse of their frequency.

Extracting Segmented Cells

To synthesize a neutrophil, individual cells were extracted from the images based on the segmentation maps and the statistics shown in Table .3.1. Only those cells which were under ± 2 standard deviations of the average area, major axis, and minor axis length, were considered. An additional constraint on their eccentricity was imposed. If the eccentricity was between 0 and +1 standard deviation, only then the cell was extracted. Once extracted, cells were center positioned on a black patch of size (64 x 64).

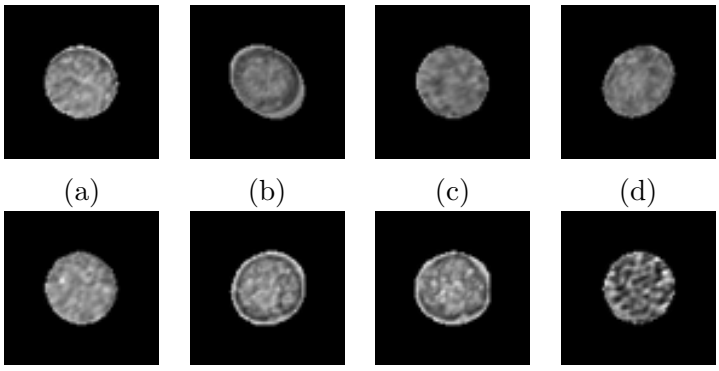


Figure. 3.6: Samples of segmented neutrophils used for training the generative models.

	mean	std
area	1430.736241	376.554888
eccentricity	0.507366	0.159627
major axis length	46.581709	6.338599
minor axis length	38.994317	6.169081

Table. 3.1: Statistics of segmented neutrophils. We extracted only those neutrophils from the segmentation maps which fell under ± 2 standard deviations of area, major, and minor axis lengths. As neutrophils are lobed cells, we imposed constrains on the eccentricity as well. Neutrophils having eccentricity in the range of $(0, +1)$ were extracted.

Tracking

To develop a motion model and synthesize sequences based on it, neutrophil trajectories were required. Trajectories were extracted by associating the segmented cells across consecutive video frames using Kalman filters based on a constant velocity model. Kalman filtering is a method for predicting the future states of a system based on the previous ones. The state of the Kalman filter was defined as: $x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]$, where (u, v) represent the center pixel coordinates, while s and r represent the scale (area) and the aspect ratio of the neutrophil's bounding box respectively. $(\dot{u}, \dot{v}, \dot{s})$ represent the respective velocities. The aspect ratio was considered to be constant. The Hungarian algorithm was applied to optimally match the Kalman filter predicted location with the detection. Introduced by [Bewley et al., 2016], we have utilized these two classical yet efficient methods for tracking multiple cells.

Generative Adversarial Networks (GANs)

As [A. Osokin and Vaggi, 2017], we trained models based on DCGAN architecture with GAN, Wasserstein GAN (WGAN) and Improved WGAN loss functions. Fig. 3.7 demonstrates the architecture of generator(G) with specific dimensions of each layer (width x

height x depth). The generator network consists of 5 convolutional layers, each of them followed by a BatchNormalization and ReLU (except the input and output) layer. The input is a noise vector of dimension (1 x 100) dimension drawn from Gaussian distribution. The generator transforms this input vector into a (64 x 64 x 1) dimensional neutrophil image by passing through each layer in G. Similar architecture is followed for discriminator (D), as shown in Fig. 3.8. Discriminator (D) takes an image (either from training set or fake image generated by generator (G)). Table. 3.2 contains the rest of the training details.

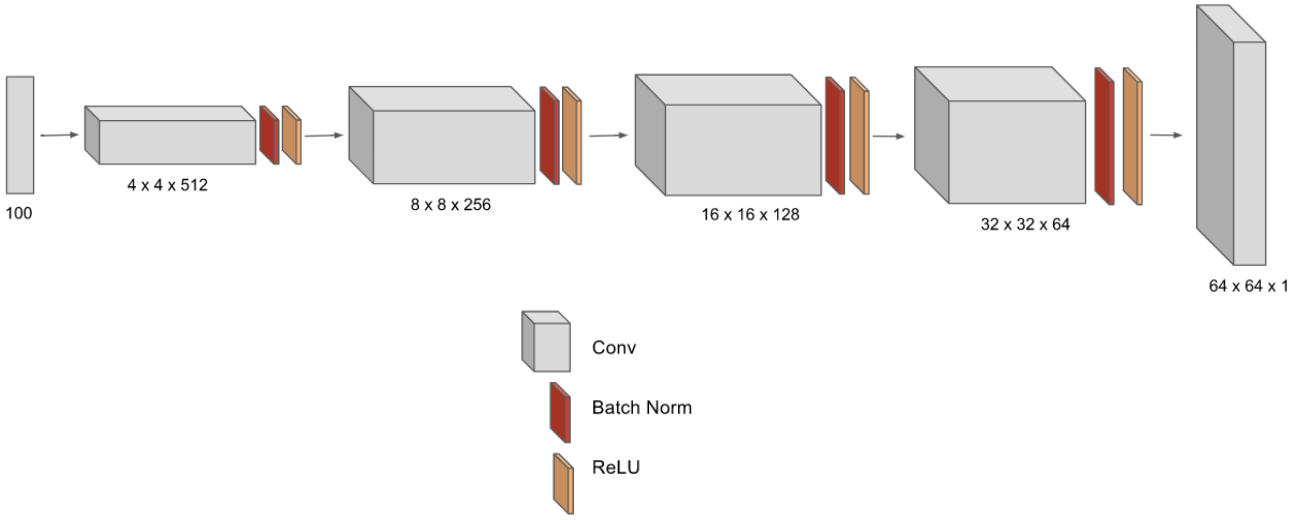


Figure. 3.7: Architecture of Generator (G). Z - a 100 dimensional uniform distribution is projected to a small spatial extent convolutional representation. Width x height x no. of feature maps is denoted at the bottom of every block. 64 x 64 x 1 image is obtained after series of convolutions.

Autoregression (AR)

Different motion patterns are observed based on the cell conditions. We build a global motion model for normal and inhibited cells respectively. It is sufficient for us to build a coarse motion model for neutrophils, by considering each neutrophil as a single compartment. Based on the existing motion characteristics, new sequences can be synthesized for the corresponding cells using AR models.

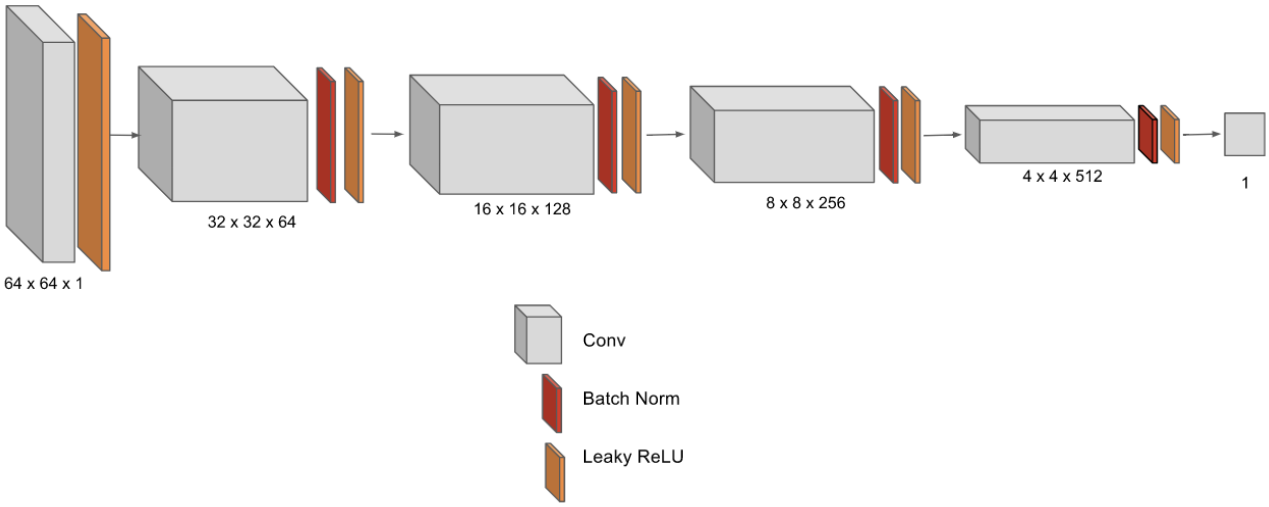


Figure. 3.8: Architecture of discriminator (D). width x height x no. of feature maps is denoted at the bottom of every block. D reads an image either from real training set or from G and after a series of convolutions, classifies it as real or fake.

Neutrophil motion is represented as trajectories of individual cells consisting of its center Cartesian coordinates across all the frames. Because we assume all the pixels (under the same conditions i.e. normal and inhibited cells) surrounding the center move similarly, we avoid building models which take into consideration movement of individual pixels. All the trajectories were truncated at 61 frames. A single neutrophil trajectory is represented as (61 x 2) vector. Repeating this process over all normal and inhibited neutrophils, and stacking the respective trajectories together, the result was (34 x 61 x 2) vector for normal neutrophils, and (44 x 61 x 2) vector for inhibited neutrophils. 34 represents number of normal neutrophils. We flattened it to (61 x 68) vector (i.e. (61 x 34 x 2)) to build an average AR model of all the normal neutrophils moving over 61 frames. Similarly, 44 represents the number of inhibited neutrophils which were flattened into (61 x 88) vector (i.e. (61 x 44 x 2)). Trajectories belonging to normal and inhibited cells are pooled separately and normalized. As discussed in Section. 2.3, these high dimensional sequences are then projected into a low-

	GAN	WGAN	Improved WGAN
Optimizer	Adam	RMSProp	Adam
Learning Rate	$D, G = 0.0002$	$D, G = 0.00005$	$D, G = 0.001$
Exponential Decay	beta1 = 0.5 beta2 = 0.999	-	beta1 = 0.0 beta2 = 0.9
Iterations of D	-	5	5
No. of epochs	10000	10000	10000
Loss Functions	(Eq. 2.1)	(Eq. 2.7)	(Eq. 2.8)

Table. 3.2: Training details of GAN framework. With DCGAN as the network architecture, we trained three different models based on the Vanilla GAN, Wasserstein GAN(WGAN) and Improved WGAN loss functions.

dimensional space. Principal Component Analysis (PCA) was used to define the appearance component of AR. PCA exploits the redundancy in data by rotating the image space with orthogonal axes along the directions of highest variance. The principal components C are extracted using Singular Value Decomposition (SVD). The original trajectories can be viewed as digital signatures through the low-dimensional space. Having C, and solving for x in (Eq. 2.9), AR coefficients B in (Eq. 2.10) are determined. New sequences can be synthesized by moving through the low-dimensional space and creating similar signatures using (Eq. 2.10). Parameter q determines the dimensionality of the subspace C; parameter d determines the order of AR coefficients $B = B_1, B_2, ...B_d$. We performed grid search over $q \in [2, 10]$ and $d \in [1, 10]$.

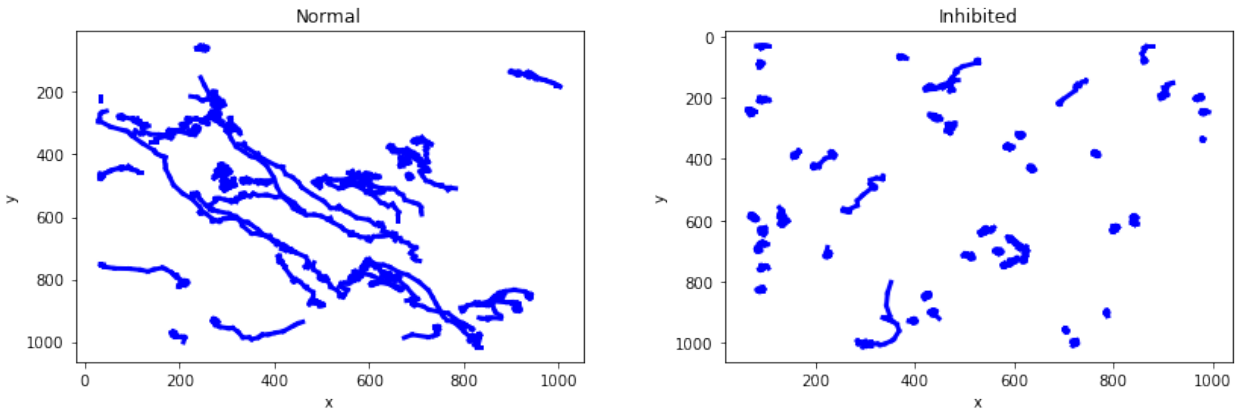


Figure. 3.9: 2D trajectory plots of normal neutrophil and inhibitor-treated (MRS) neutrophil. The inhibitor-treated (MRS) neutrophil tend to exhibit less movements in comparison to the normal ones.

CHAPTER 4

RESULTS AND ANALYSIS

4.1 Segmentation

The motive of training FC-DenseNet was to take advantage of the trained model to automatically segment neutrophils from images. Thus, in the future this model can be used instead of relying on manual segmentation as discussed in Section .3.2. 96% pixel classification accuracy was achieved on the holdout test set, using the network discussed in the Section. 3.3. Fig. 4.1 displays sample results of segmentation. Table. 4.1 shows dice coefficient results.

	Dice Coefficient
Neutrophil border	0.478704134139
Neutrophil cell body	0.877087162892
Background	0.988928566552

Table. 4.1: Dice coefficient is used to quantify the performance of segmentation. It measures the overlap between the predicted and ground truth segmentation map. It ranges from 0 (no overlap) to 1 (perfect overlap).

4.2 Generative Adversarial Networks (GANs)

GAN generated images are shown in Fig. 4.2, WGAN images are shown in Fig. 4.3 and Improved WGAN in Fig. 4.4.

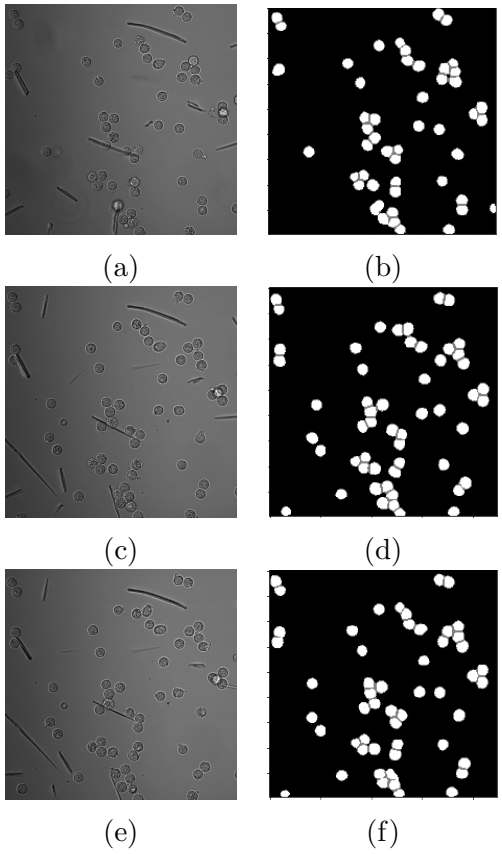


Figure. 4.1: Segmentation results. (left) column displays the test images and (right) column displays the corresponding predicted segmentation maps.

Optimization to Reconstruct the Test Set

Visual examination of images generated by GAN doesn't inform much about its performance. In the context of GANs and other likelihood-free approaches, evaluation is even harder, because the models do not provide a way to compute the log-likelihood on the test set, which is the most common evaluation technique. Several techniques have been discussed in [A. Osokin and Vaggi, 2017] for evaluation of GANs. We employed the optimization-based approach to check if the test samples can be reconstructed well. This can be used to test for mode collapse, a common failure in GANs, wherein very poor diversity is exhibited amongst generated samples. For a fixed trained generator G we examine how well it can reconstruct

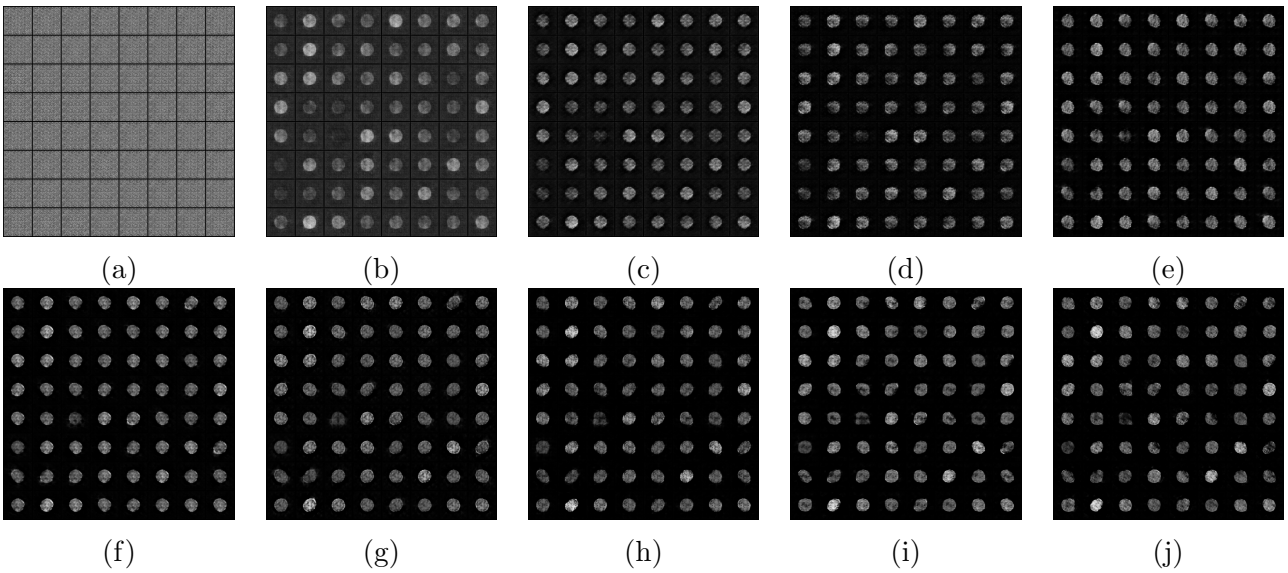


Figure. 4.2: GAN synthesized images of neutrophils. Images are displayed every 100 epochs starting from 100^{th} upto 1000^{th} epoch.

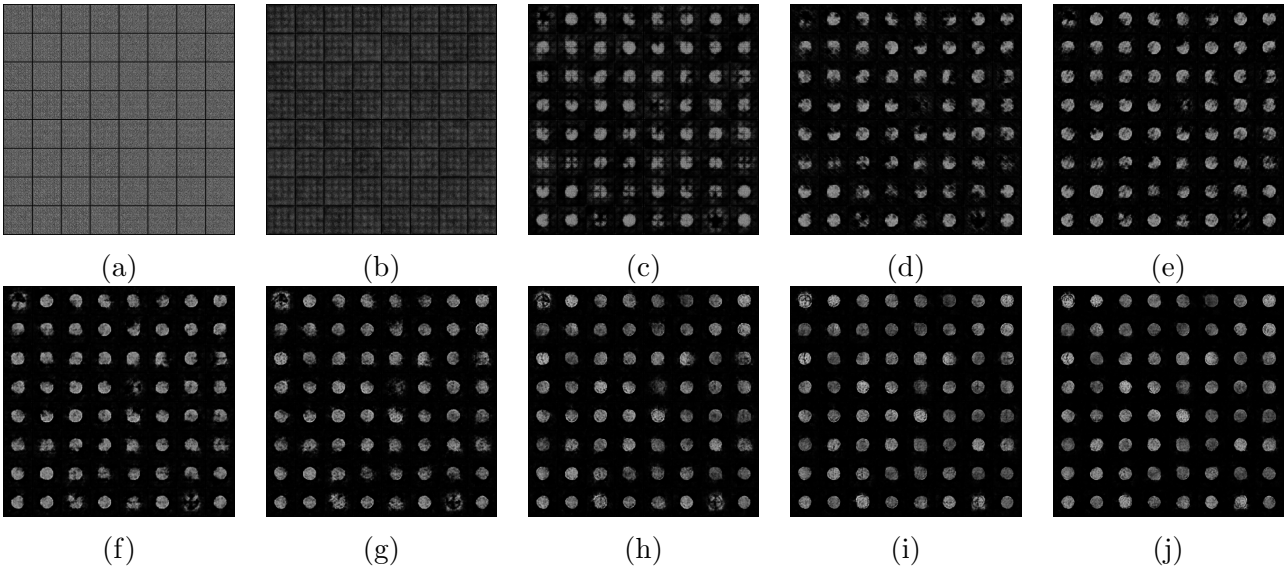


Figure. 4.3: WGAN synthesized images of neutrophils. Images are displayed every 100 epochs starting from 100^{th} upto 1000^{th} epoch. WGAN takes more time to train in comparison to GAN because discriminator D is trained for more iterations before generator G is trained.

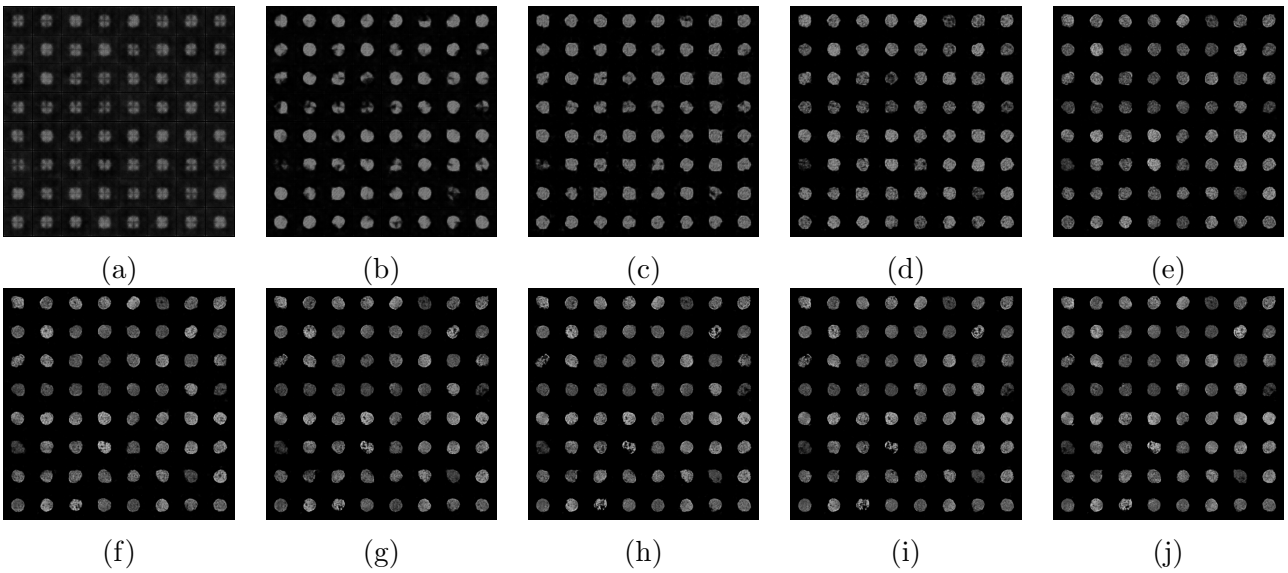


Figure. 4.4: Improved WGAN synthesized images of neutrophils. Images are displayed every 100 epochs starting from 100^{th} upto 1000^{th} epoch. The generated samples display more variation even in the initial epochs in comparison to samples generated by GAN.

images from a held out test set. For each image in the test set, we minimize the $L2$-distance between the generated and test images w.r.t. the noise vector z. We use 50 iterations of L-BFGS [Liu and Nocedal, 1989] and select the best reconstruction out of 3 runs. We also report the negative log likelihood (NLL) w.r.t. the prior P_z of the noise vectors z. Fig. 4.5 displays the results.

	GAN	WGAN	Improved WGAN
100	0.432 ± 0.004	0.376 ± 0.006	0.165 ± 0.027
500	0.142 ± 0.025	0.073 ± 0.013	0.041 ± 0.006
1000	0.064 ± 0.008	0.044 ± 0.007	0.037 ± 0.007
5000	0.054 ± 0.006	0.034 ± 0.005	$\mathbf{0.031 \pm 0.005}$
10000	0.044 ± 0.006	$\mathbf{0.033 \pm 0.005}$	$\mathbf{0.033 \pm 0.005}$

Table. 4.2: $L2$-error of reconstructions to hold out test set. Lower the better

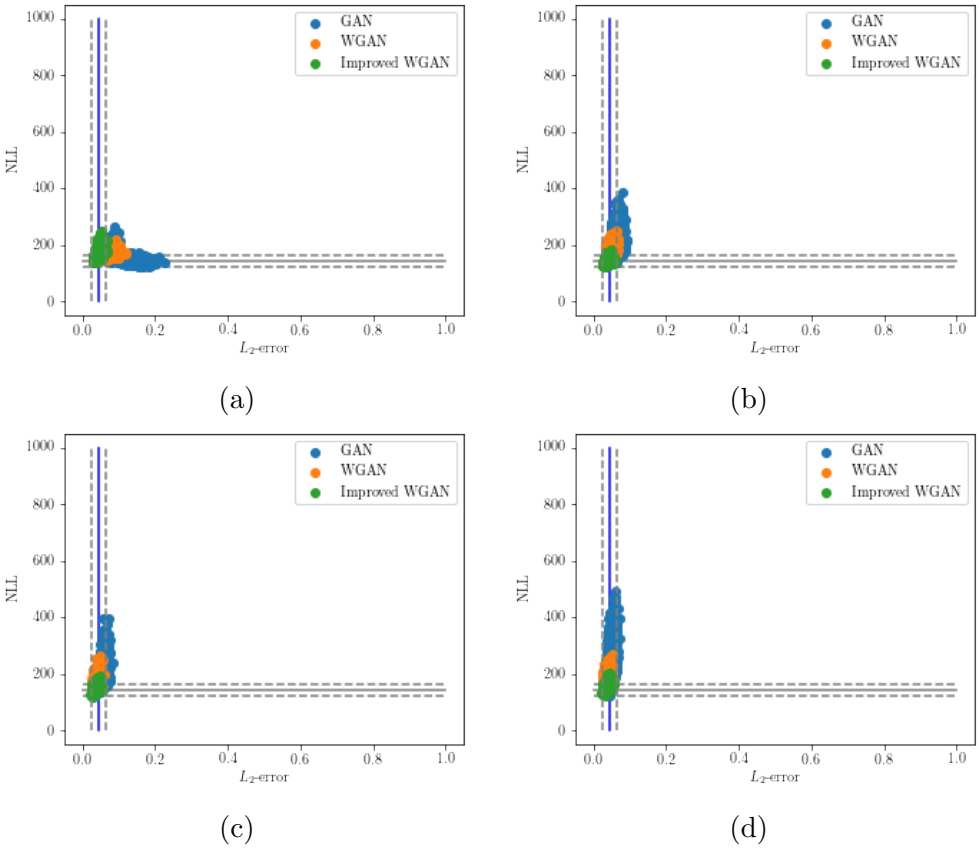


Figure. 4.5: Reconstruction errors against negative log likelihood (NLL) of the latent vectors found by reconstruction are displayed for 500, 1000, 5000 and 10000 epochs. The vertical blue line shows the mean $L2$-error. Horizontal gray lines show mean NLL ($\pm 3std$) of the noise sampled from the Gaussian prior. Lower values for both are better. Improved WGAN consistently shows lower values for NLL and L2.

Latent Space Walk

We can interpolate between points in the latent space and understand the landscape Fig. 4.6. Walking the manifold can identify if there are any sharp transitions and whether the network has memorized. If walking the latent space results in smooth semantic changes to the image generations we can reason that the model has learned relevant, interesting representations [Radford et al., 2015].

	GAN	WGAN	Improved WGAN
100	$\mathbf{138 \pm 6}$	144 ± 7	152 ± 17
500	146 ± 12	172 ± 9	170 ± 14
1000	242 ± 29	191 ± 13	$\mathbf{141 \pm 7}$
5000	204 ± 36	165 ± 17	143 ± 10
10000	230 ± 49	178 ± 18	149 ± 10

Table. 4.3: NLL of the noise vectors z w.r.t. the prior P_z. Lower the better.

4.3 Autoregressive (AR) Models

AR models are trained according to the procedure described in Section. 3.3. The trained model is used to synthesize novel sequences which are perceptually similar to the original ones rather than identical. Fig. 4.7 displays the synthesized sequences superimposed over the original sequences. This helps to validate that the synthesized sequences do follow the distribution of original sequences. In this section we discuss the impact of changing the dimensions of subspace C and the order of AR model and the error measures used to assess the quality of results.

PCA Analysis

As discussed in Section. 3.3, the high-dimensional trajectories of normal and inhibited neutrophils are projected into a low-dimensional space C in order to reduce the complexity. The original trajectories can be viewed as digital signatures through the low-dimensional space. Fig. 4.8 depicts the variance explained by top five principal components. As the top three principal components capture maximum variance of the data, parameter q which determines the dimensionality of the subspace C was set to $q = 3$. Fig. 4.9 illustrates the characteristics

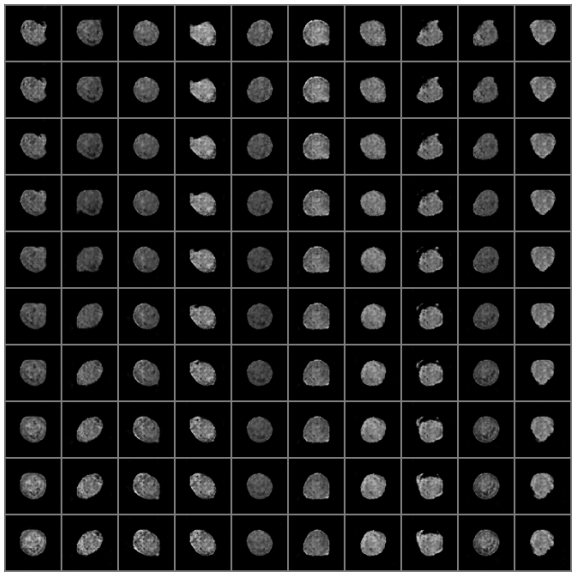
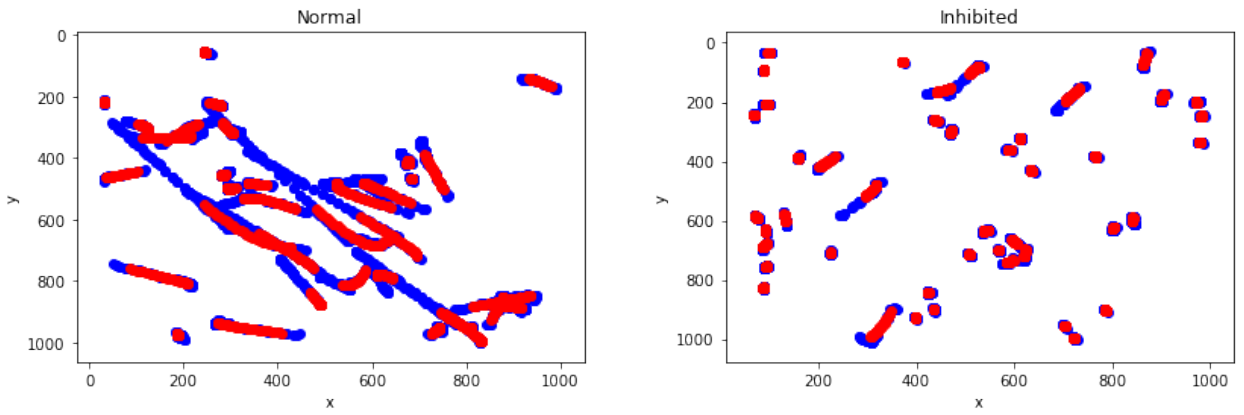


Figure. 4.6: Interpolation between a series of ten random points in the latent space depicts that the space learned has smooth transitions. Top row depicts the starting location for each of the 10 points. Last row depicts the respective ending locations.

of inhibited neutrophils are significantly different in comparison to normal neutrophils in the subspace C.

One Step Prediction Error

There are several ways to evaluate synthesized sequences. We employed the one-step prediction error discussed in [Hyndman et al., 2007]. To calculate one-step prediction error of order d all sets of $d+1$ consecutive frames are used. Model is initialized with d frames and $d+1$ frame is generated. Mean squared error is used to calculate the difference between synthesized frame and the original frame. Fig. 4.10 depicts that the one-step prediction error is improved when more temporal information is used to generate subsequent sequences. It also reflects that using higher order AR models degrade the performance. As AR(2) performs



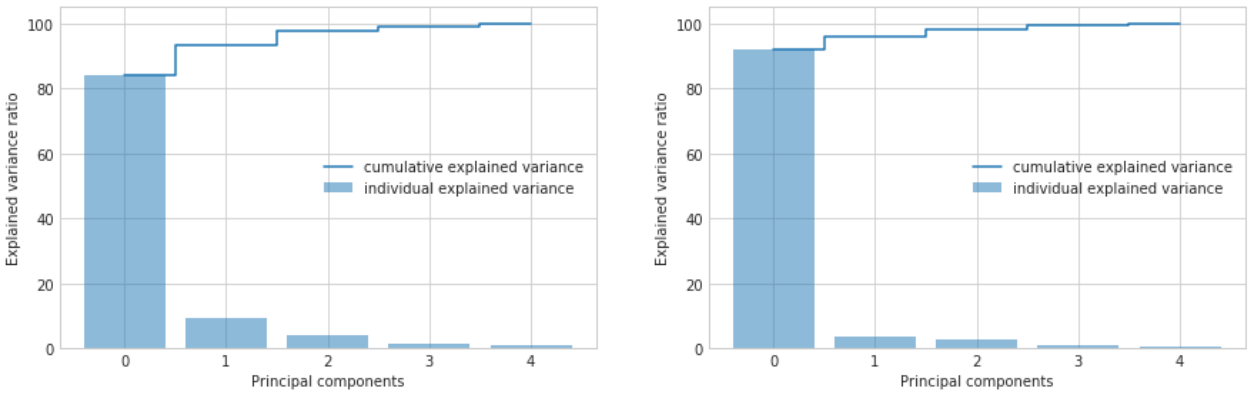
(a) Synthesized sequence of normal neutrophils (b) Inhibited Neutrophils

Figure. 4.7: To synthesize a sequence we randomly selected a starting point and predicted their motion for 25 frames ahead in the sequence for all the neutrophils. Superimposing synthesized sequences(red) over the original sequences(blue) helps to understand that the synthesized sequences do indeed follow the original distribution.

consistently better than others, parameter d which determines the order of AR model, was set to $d = 2$.

4.4 Synthesis

Synthesized neutrophil behavior consists of two parts: content and appearance is sampled from the trained Improved WGAN generator G and motion is sampled from a point in subspace C. Using (Eq. 2.10) iteratively, different sequences are generated. These new sequences are then projected back into the original space, leading to a new motion pattern synthesized entirely from the eigenvector information. The separation of motion and appearance in two streams enable GANs and AR process to identify the respective key features. This results in movement of only the foreground cells and leaves the rest untouched. It also gives an advantage of synthesizing video clips of different cells following different trajectories but



(a) Normal Neutrophils (b) Inhibited Neutrophils

Figure. 4.8: Percentage of variance explained by top five principal components of normal and inhibited neutrophils. The top three were used to compute AR parameters because these components capture the maximum variance of the data.

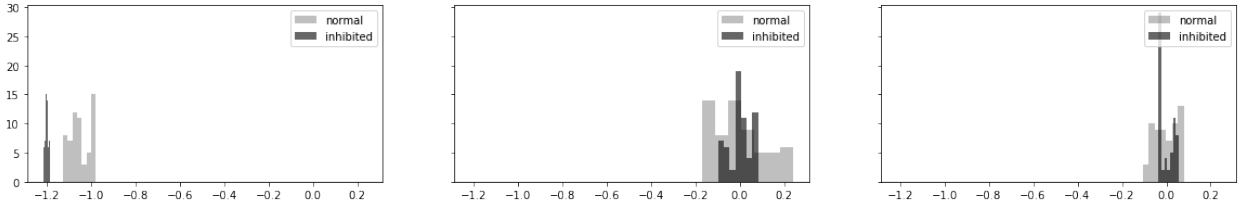


Figure. 4.9: Distribution of values of top three principal components for normal and inhibited neutrophils superimposed. The inhibited neutrophil values follow a restricted distribution in comparison to normal neutrophils.

nonetheless looks similar to the existing motion patterns. Fig. 4.11, 4.12, 4.13 display the results of synthesis.

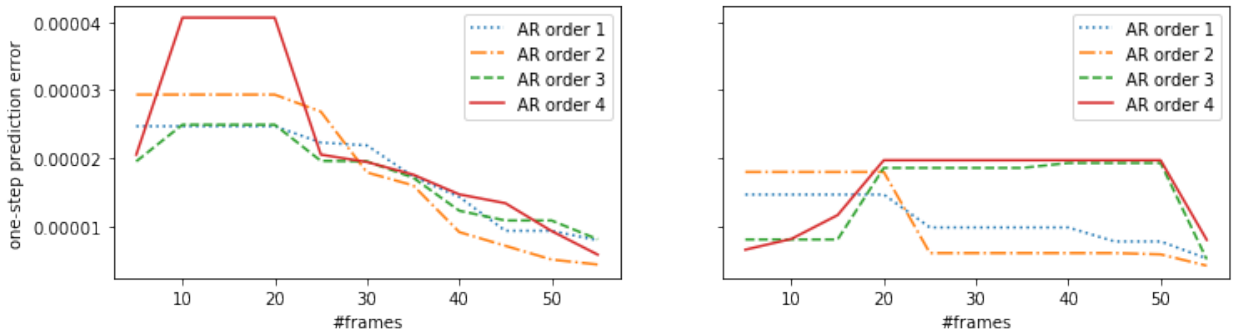


Figure. 4.10: One-step prediction error for normal(left) neutrophils and (right) represents the same for inhibited neutrophils. It displays the impact of using higher order AR model and using more frames for training the AR model. AR model of order $d = 2$ consistently performs better than AR model of order $d = 1$. But, the performance degrades when higher order AR models are used. The error curves were passed through a median filter for visualization purposes.

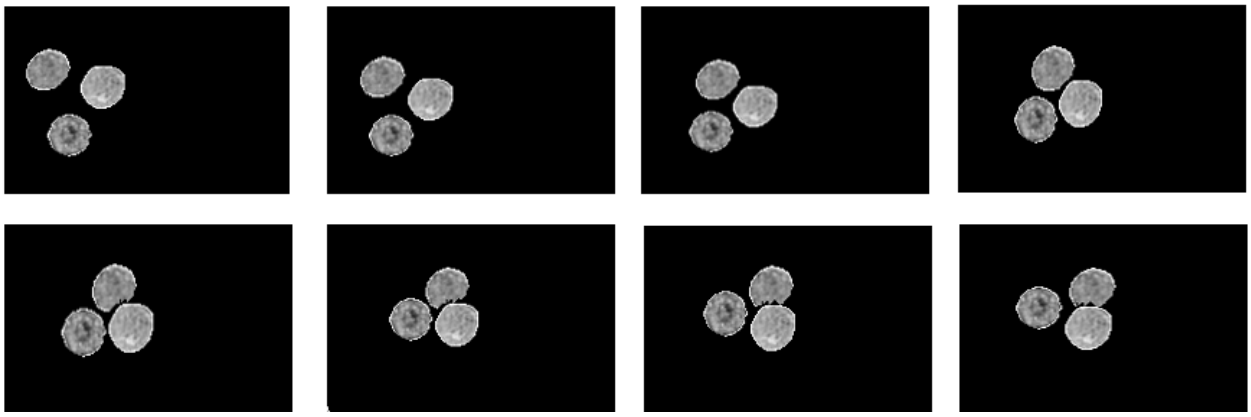


Figure. 4.11: Sample results of appearance and motion synthesis combined. First frame is displayed in the leftmost column - top row and the last image is displayed in the rightmost column - bottom row. Every image represents the frames between starting and ending frames for every epoch. Each neutrophil represents a different appearance and follows a different trajectory.

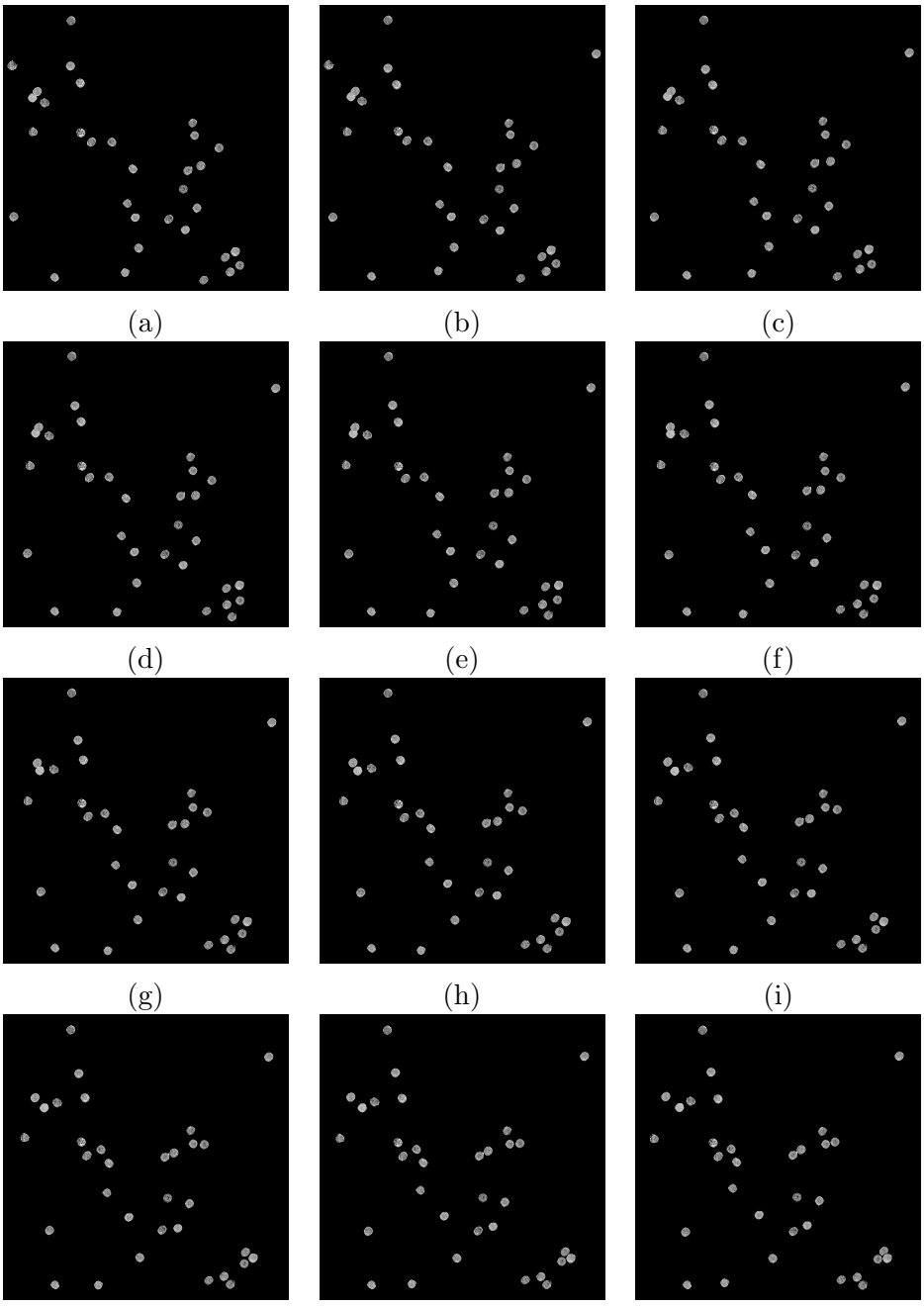


Figure. 4.12: First 12 frames of synthesized sequence of normal neutrophils, from top to bottom.

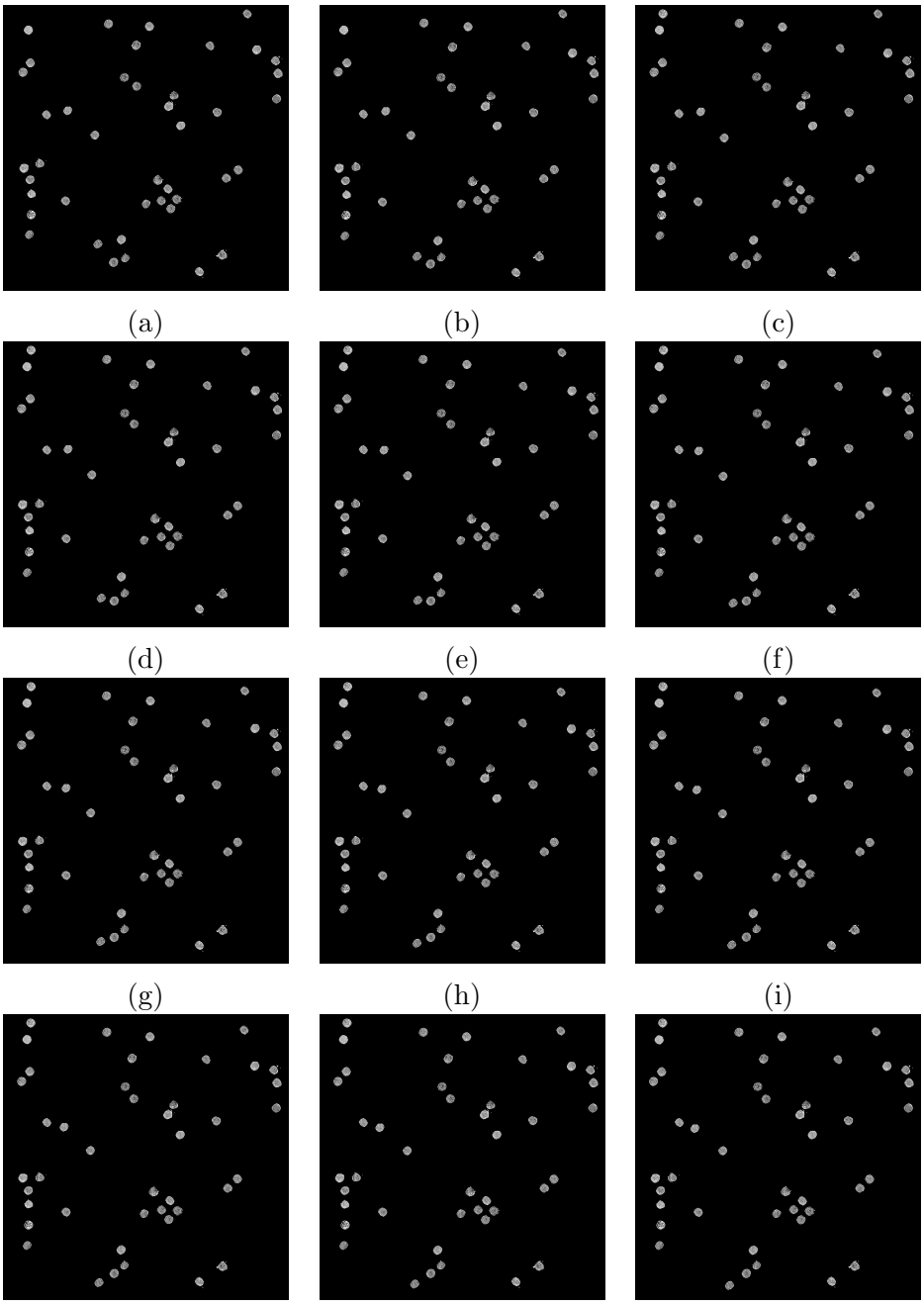


Figure. 4.13: First 12 frames of synthesized sequence of inhibited neutrophils, from top to bottom.

CHAPTER 5

CONCLUSION

This thesis presents a two stream approach to simulate human neutrophil behavior. Neutrophil behavior can be decomposed into two components: appearance and motion. Appearance encodes the spatial statistics of the objects and motion encodes the temporal dynamics. Based on this notion and owing to the very limited data at our disposal, we utilized GANs to learn the appearance of neutrophils, and AR models to learn their motion. GAN training consists of adversarial training of a generator (G) against a discriminator (D) w.r.t the images of segmented neutrophils. AR models are trained by regressing over the past values of neutrophil trajectories in a low-dimensional subspace. To synthesize neutrophil behavior: content and appearance is sampled from the trained generator (G) and motion is sampled from a point in the subspace and using (Eq. 2.10) iteratively to generate a sequence. We have validated through experiments that our approach does indeed synthesize neutrophil appearance as well as its motion which is on par with distribution of real data.

5.1 Future Work

The proposed model combine GANs and AR. GANs encode the spatial aspect, while autoregression encodes the temporal aspect of neutrophil. Both models are trained independently. Because of this disconnect between GAN and AR, we cannot model neutrophil appearance changes w.r.t to time. Recently, ([Tulyakov et al., 2017],[Vondrick et al., 2016]) presented similar two-stream approaches for video synthesis entirely based on generative adversarial networks. Their models learn to model the appearance and motion of objects of interest in

unsupervised manner. [Tulyakov et al., 2017] has shown that in spite of lacking supervision signals about the decomposition of motion and content in natural videos, their framework can learn to disentangle these two factors through a novel adversarial training scheme. Given sufficient data, these techniques can be extended for neutrophils as well.

BIBLIOGRAPHY

[A. Osokin and Vaggi, 2017] A. Osokin, A. Chessel, R. C. S. and Vaggi, F. (2017). GANs for biological image synthesis. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

[Agianpuye and Minoi, 2014] Agianpuye, A. S. and Minoi, J.-L. (2014). Synthesizing neutral facial expression on 3d faces using active shape models. In *Region 10 Symposium, 2014 IEEE*, pages 600–605. IEEE.

[Arikan et al., 2003] Arikan, O., Forsyth, D., and O'Brien, J. (2003). Motion Synthesis from Annotations. *"ACM Transactions on Graphics (TOG)"*, 22:402–408.

[Arjovsky et al., 2017] Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. *arXiv preprint arXiv:1701.07875*.

[Beucher and Meyer,] Beucher, S. and Meyer, F. The morphological approach to segmentation: the watershed transformation.

[Bewley et al., 2016] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468.

[Buck et al., 2012] Buck, T. E., Li, J., Rohde, G. K., and Murphy, R. F. (2012). Toward the virtual cell: automated approaches to building models of subcellular organization learned from microscopy images. *Bioessays*, 34(9):791–799.

[Campbell et al., 2004] Campbell, N., Dalton, C., Gibson, D., Oziem, D., and Thomas, B. (2004). Practical generation of video textures using the auto-regressive process. *Image & Vision Computing*, 22 (10):819 – 827. Publisher: Elsevier.

[Cootes et al., 1995] Cootes, T. F., Taylor, C. J., Cooper, D. H., and Graham, J. (1995). Active shape models—their training and application. *Comput. Vis. Image Underst.*, 61(1):38–59.

[Dodgson et al., 2017] Dodgson, J., Chessel, A., Vaggi, F., Giordan, M., Yamamoto, M., Arai, K., Madrid, M., Geymonat, M., Abenza, J. F., Cansado, J., Sato, M., Csikasz-Nagy, A., and Carazo Salas, R. E. (2017). Reconstructing regulatory pathways by systematically mapping protein localization interdependency networks. *bioRxiv:11674*.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

[Gharakhanian, 2017] Gharakhanian, A. (2017). Generative Adversarial Networks - Hot Topic in Machine Learning. `https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html`.

[Gulrajani et al., 2017] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved Training of Wasserstein GANs. *arXiv preprint arXiv:1704.00028*.

[He et al., 2015a] He, K., Zhang, X., Ren, S., and Sun, J. (2015a). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

[He et al., 2015b] He, K., Zhang, X., Ren, S., and Sun, J. (2015b). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

[Huang et al., 2017] Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[Hyndman et al., 2007] Hyndman, M., Jepson, A. D., and Fleet, D. J. (2007). Higher-order autoregressive models for dynamic textures. In *BMVC*, pages 1–10.

[I. Goodfellow et al., 2014] I. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

[Jégou et al., 2016] Jégou, S., Drozdzal, M., Vázquez, D., Romero, A., and Bengio, Y. (2016). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *CoRR*, abs/1611.09326.

[Kayalibay et al., 2017] Kayalibay, B., Jensen, G., and van der Smagt, P. (2017). Cnn-based segmentation of medical imaging data. *CoRR*, abs/1701.03056.

[Kingma and Welling, 2013] Kingma, D. and Welling, M. (2013). Auto-encoding variational bayes. *CoRR*, abs/1312.6114.

[Köppel et al., 2015] Köppel, M., Doshkov, D., Racape, F., Ndjiki-Nya, P., and Wiegand, T. (2015). On the usage of the 2d-ar-model in texture completion scenarios with causal boundary conditions. *Image Commun.*, 32(C):106–120.

[LeCun and Cortes, 2010] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

[Liu and Nocedal, 1989] Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528.

[Long et al., 2014] Long, J., Shelhamer, E., and Darrell, T. (2014). Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038.

[Oord et al., 2016] Oord, A., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., and Kavukcuoglu, K. (2016). Conditional Image Generation with PixelCNN Decoders. *arXiv preprint arXiv:1606.05328*.

[Oziem et al., 2004] Oziem, D., Campbell, N., Dalton, C., D., G., and B, T. (2004). Combining Sampling and Autoregression for Motion Synthesis. *"Proceedings of the Computer Graphics International"*, pages 510–513.

[Quinn et al., 2015] Quinn, S., Zahid, M., JR, D., RJ., F., CW, L., and SC, C. (2015). Automated identification of abnormal respiratory ciliary motion in nasal biopsies. *"Science Translational Medicine"*, 7:299ra124.

[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv preprint arXiv:1511.06434*.

[Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

[Sehgal, 2005] Sehgal, A. (2005). How Neutrophils Kill Microbes. *Annual Review of Immunology*, 23:197–223.

[Song et al., 2012] Song, C., Wei, J., Fang, Q., Liu, S., Wang, Y., and Dang, J. (2012). Tongue shape synthesis based on active shape model. In *Chinese Spoken Language Processing (ISCSLP), 2012 8th International Symposium on*, pages 383–386. IEEE.

[Tieleman and Hinton, 2012] Tieleman, T. and Hinton, G. (2012). Rmsprop adaptive learning.

[Tulyakov et al., 2017] Tulyakov, S., Liu, M., Yang, X., and Kautz, J. (2017). Mocogan: Decomposing motion and content for video generation. *CoRR*, abs/1707.04993.

[Vondrick et al., 2016] Vondrick, C., Pirsiavash, H., and Torralba, A. (2016). Generating videos with scene dynamics. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 613–621. Curran Associates, Inc.

[Yen, 2017] Yen, Y. H. (2017). BBox with angle Label Tool. *GitHub repository*.

[Zhao and Murphy, 2007] Zhao, T. and Murphy, R. F. (2007). Automated learning of generative models for subcellular location: building blocks for systems biology. *Cytometry Part A*, 71(12):978–990.