Towards an Unsupervised Spatiotemporal Representation of Cilia Video Using A Modular Generative Pipeline

Meekail Zain[‡§†], Sonia Rao[‡†], Nathan Safir[‡], Quinn Wyner[§], Isabella Humphrey[‡§], Alex Eldridge[§], Chenxiao Li[‖], BahaaEddin AlAila[‡**], Shannon Quinn[‡¶*]

Abstract—Motile cilia are a highly conserved organelle found on the exterior of many human cells. Cilia beat in rhythmic patterns to transport substances or generate signaling gradients. Disruption of these patterns is often indicative of diseases known as ciliopathies, whose consequences can include dysfunction of macroscopic structures within the lungs, kidneys, brain, and other organs. Characterizing ciliary motion phenotypes as healthy or diseased is an essential step towards diagnosing and differentiating ciliopathies. We propose a modular generative pipeline for the analysis of cilia video data so that expert labor may be supplemented for this task. Our proposed model is divided into three modules: preprocessing, appearance, and dynamics. The preprocessing module augments the initial data, and its output is fed frame-by-frame into the generative appearance model which learns a compressed latent representation of the cilia. The frames are then embedded into the latent space as a low-dimensional path. This path is fed into the generative dynamics module, which focuses only on the motion of the cilia. Since both the appearance and dynamics modules are generative, the pipeline itself serves as an end-to-end generative model. This thorough and versatile model allows experts to spend less time caught in the minutiae of cilia biopsy analysis, while also enabling new insights by quantifying subtle patterns that would be otherwise difficult to categorize.

Index Terms—Machine Learning, Data Science, Video Analysis, Generative Modeling, Variational Autoencoder, Modular, Pipeline

Introduction

Motile cilia are organelles commonly found throughout the human body, such as in the bronchial and nasal passages [HGGRD99][SS90]. Cilia beat in synchronous, rhythmic patterns to expel foreign matter, collectively forming the mucociliary defense, a vital mechanism for sinopulmonary health [BMO17]. Ciliopathies are genetic disorders which can adversely affect the motion of cilia [FL12]. Disorders resulting from the disruption of ciliary motion range from sinopulmonary diseases such as primary ciliary dyskinesia (PCD) [OCH+07] to mirror symmetric organ placement and situs inversus [CA17] or randomized left-right organ placement as in heterotaxy [GZT+14]. Precise diagnosis of patients exhibiting abnormal ciliary motion prior to surgery may provide clinicians with opportunities to institute prophylactic respiratory therapies to prevent complications. Therefore, the study of ciliary motion may have a broad clinical impact.

Visual examination of the ciliary waveform by medical professionals is critical in diagnosing ciliary motion defects, but such manual analysis is highly subjective and prone to error [RWH+14] [KSC17]. This approach also precludes the possibility of cross-institutional and longitudinal studies which include assessment of ciliary motion. Therefore, we aim to develop an unsupervised, computational approach to analyze ciliary motion, developing a quantitative "library" of well-defined, clinically relevant ciliary motion phenotypes. Clustering and classification are established problems in machine learning. However, their applications to ciliary waveform analysis are difficult, as cilia exhibit subtle, rotational, non-linear motion [QFLC11]. While attempts have been made at addressing this problem, we note that generic dynamics models fail to classify and cluster this type of motion accurately or meaningfully, and are insufficient for generating a semantically potent representation. We thus apply a novel machine learning approach to create an underlying representation which then can be used for downstream tasks such as classification and clustering, and any other tasks that experts may deem necessary. Furthermore, we avoid using labeled data—specifically videos anotated based on the health/type of ciliary motion displayed—in order to free the model from systematic assumptions naturally imposed by labels: the choice of labels themselves can inadvertently limit the model by asserting that all data *must* conform to those exact labels. An unsupervised model has the freedom to discover potential semantically meaningful patterns and phenotypes that fall outside current clinical thinking. Furthermore, an unsupervised model is independent of expert input. Pragmatically, an unsupervised model can be trained and used directly after data acquisition, rather than having to wait on expert labeling. This simultaneously reduces the barriers to access as a scientific tool, and the associated expenses of use.

Our approach is to create a pipeline that learns a low-dimensional representation of ciliary motion on unlabeled data. The model we propose considers the spatial and temporal dimensions of ciliary motion separately. The pipeline encodes each

† *These authors contributed equally.*
‡ *Computer Science Department, Franklin College of Arts and Sciences*
§ *Mathematics Department, Franklin College of Arts and Sciences*
‖ *Comparative Biomedical Sciences, College of Veterinary Medicine*
** *Institute for Artificial Intelligence, Franklin College of Arts and Sciences*
* *Corresponding author: spq@uga.edu*
¶ *Cellular Biology Department, Franklin College*

frame of the input video and then encodes the paths between frames in the latent space. The low-dimensional latent space in this pipeline will have semantic significance, and thus the distribution and clustering of points in the latent space should be meaningful for those studying ciliary motion and its connection to ciliopathies.

Related Works

A computational method for identifying abnormal ciliary motion patterns was proposed by Quinn 2015 [QZD+15]. The authors hypothesize ciliary motion as an instance of a dynamic texture, which are rhythmic motions of particles subjected to stochastic noise [DCWS03] and include familiar patterns such as flickering flames, rippling water, and grass in the wind. Each instance of dynamic texture contains a small amount of stochastic behavior altering an otherwise consistent visual pattern. The authors chose to consider ciliary motion as a dynamic texture as it consists of rhythmic behavior subject to stochastic noise that collectively determine the beat pattern. They then used autoregressive (AR) representations of optical flow features that were fed into a support vector machine classifier to decompose high-speed digital videos of ciliary motion into "elemental components," or quantitative descriptors of the ciliary motion, and classify them as normal or abnormal.

While this study proved there is merit in treating ciliary motion as a dynamic texture, the use of an AR model for the classification task imposed some critical limitations. While AR models are often used in representing dynamic textures, they are primarily used in distinguishing distinct dynamic textures (e.g., rippling water from billowing smoke), rather than identifying different instances of the same texture (e.g., cilia beating normally versus abnormally). Additionally, AR models impose strong parametric assumptions on the underlying structure of the data, rendering AR models incapable of capturing nonlinear interactions. Lastly, even though the majority of the pipeline is automated, their study relied on clinical experts to manually annotate the video data with regions of interest (ROIs) in order to serve as ground truth for the inference. Drawing ROIs required specialized labor, increasing the cost and time of clinical operations. This is also potentially problematic in that expert drawn ROIs introduce the same subjective bias that the study is ostensibly attempting to remove.

The model proposed by Quinn 2015 was improved upon by Lu 2018 [LMZ+18], the latter attempt using stacked Fully Convolutional DenseNets [HLW16] and Long Short-Term Memory (LSTM) networks [GSC99]. Densely Connected Convolutional Networks, referred to as DenseNets, do not make strong parametric or linear assumptions about the underlying data, allowing more complex behavior to be captured. Once Lu 2018 extract segmentation masks using their 74-layer FCDenseNet, ciliary motion is treated as a time series using convolutional long short-term memory (Conv-LSTM) networks, a specific type of recurrent neural network (RNN), to model the long-term temporal dependencies in the data.

We aim to build upon these studies by developing a fully unsupervised approach to characterizing ciliary motion phenotypes. This pipeline is advantageous in that it does not need hand-drawn ROI maps nor a labeled dataset as training data. While clinicians acknowledge the existence of distinct ciliary waveform phenotypes beyond "normal" and "abnormal", experts lack standard guidelines for qualitatively or quantitatively categorizing ciliary beat pattern. Additionally, experts may not observe the level of quantitative detail required to associate complex motion phenotypes with specific ciliopathies and genetic mutations [QZD+15]. Thus, we shift away from a classification-style task (classifying abnormal versus normal ciliary motion) to a representational learning task to generate meaningful, low-dimensional representations of ciliary motion. Unsupervised representation learning enables a model to learn families of complex ciliary motion phenotypes beyond the normal-abnormal binary.

Methods

Our proposed model is divided into three modules: preprocessing, appearance, and dynamics. The preprocessing module primarily serves to supplement input data by generating segmentation masks and extracting dense optical flow vector fields and pertinent differential quantities. Segmentation masks are used to limit spatial representation learning to video regions containing cilia, and optical flow fields are computed from consecutive frames as a compressed representation of temporal behavior. The predicted segmentation masks and optical flow entities are concatenated with the original video data as additional channels to each frame to form an augmented video. Each expanded video is fed frame-by-frame to the appearance module which utilizes a Variational Autoencoder (VAE) [KW19] to learn a compressed spatial representation for images of cilia. Videos are then embedded as sequences of points in the compressed latent space. The dynamics module employs another VAE to learn a representation from this compressed sequence, in order to reduce the amount of irrelevant information considered. If it were to instead train on the original video itself, the information would be too high-volume, potentially drowning out useful information in a sea of noise. This compressed sequence allows it to focus only on the motion of cilia. The dynamics module VAE is trained on potentially random subsequences of the embedded representations of video in order to assure that the temporal representation learned is adequately robust to reconstruct arbitrary parts of the sequence. Through this construction, we factor the representation of cilia into disentangled spatial and temporal components.

Data

Our data, obtained from the Quinn 2015 study, consist of nasal biopsy samples observed in patients with diagnosed ciliopathies and in healthy controls [QZD+15]. Nasal epithelial tissue was obtained from the inferior nasal turbinate using a Rhino-Pro curette, and cultured for three passages prior to recording. Grayscale video data was recorded for 1.25 seconds using a Phantom v4.2 high speed camera at 200 frames per second, resulting in 250 frames per sample. Recorded videos vary in dimension, ranging from 256 to 640 pixels on either axis. Segmentation masks used during the training of the preprocessing module were generated manually using ITK-SNAP, where each pixel is a binary value corresponding to whether the pixel contains cilia. Our dataset has a total of 325 sample videos, taken from Quinn 2015's cohort sampled at the University of Pittsburgh, and 230 ground-truth segmentation masks.

Because healthy cilia rhythmically beat at around 10-12Hz and our grayscale videos are recorded at 200 frames per second, there are approximately 17 frames per single ciliary beat cycle [QZD+15]. As such, we truncate our videos to 40 frames to capture at minimum 2 full beat cycles; the starting frame is randomly sampled. Because each video varies in dimensions, we

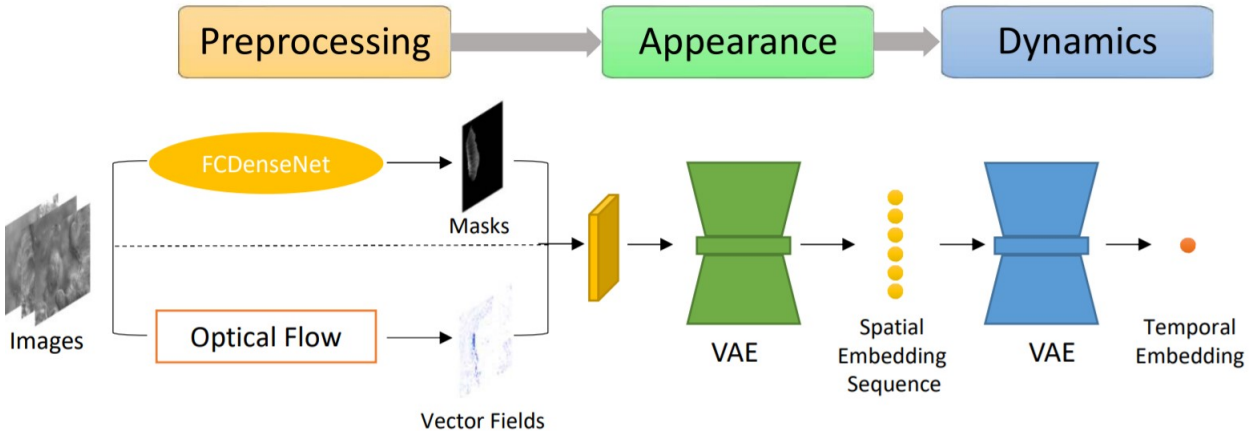

Fig. 1: *The proposed framework for creating a disentangled spatiotemporal representation*

obtain patches of size 128×128 as inputs to both the preprocessing and appearance modules. Instead of randomly sampling crops, we extract the first frame of the truncated video, and tile each frame-mask set such that no 128×128 patches overlap. The preprocessing module supplemented the 95 raw videos without corresponding ground-truth segmentation masks with segmentation masks predicted by a Fully Convolutional DenseNet.

Preprocessing

The preprocessing module primarily functions to generate segmentation masks that distinguish spatial regions containing cilia from background noise and supplement cilia data with measures of temporal behavior, such as optical flow and its derivative values.

Because we are interested in modelling the spatiotemporal behavior of *only* cilia, segmentation masks, which provide a direct mapping to pixels of interest within each frame, are critical within the appearance module to limit representation learning to cilia localities and ignore background noise. Although the end-to-end pipeline provides an unsupervised framework to represent and characterize complex and dynamic ciliary motion phenotypes, this module utilizes *supervised* segmentation to produce initial segmentation masks. Because we do not have ground-truth segmentation masks for every sample in our dataset, a supervised network allows us to augment our set such that each raw video has a corresponding segmentation mask to be used in subsequent modules. We draw upon prior supervised segmentation literature to implement FCDenseNet, a fully convolutional dense network that is able to leverage deep learning advantages without excessive parameters or loss of resolution. Each layer in a DenseNet is connected to every other layer in a feed-forward fashion; each layer takes the previous layers' feature maps as input, and its respective feature map is used by following layers. Fully Connected DenseNets (FCDenseNets) expand on this architecture with the principle goal of upsampling to recover input resolution [JDV+17]. Building a straightforward upsampling path requires multiplication of high-resolution feature maps, resulting in a computationally intractable number of feature maps. To mitigate this "feature explosion" issue, FCDenseNets upsample only the preceding dense block instead of upsampling all feature maps concatenated in previous layers. We modify and train a FCDenseNet to generate usable segmentation masks as input to the appearance module. Our architecture, shown in 2, consists of dense blocks, transition blocks, and skip connections totalling to 103 layers.

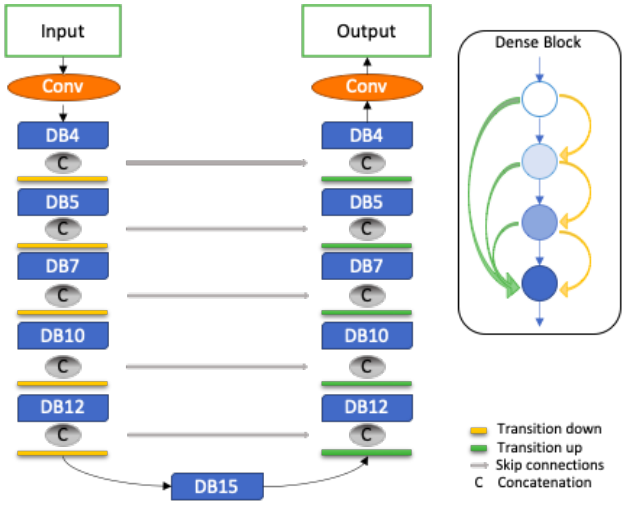

Fig. 2: *Fully Convolutional Dense Net with 103 layers*

Although we utilize a supervised segmentation network, we note that this is not necessary. We will be pursuing unsupervised methodologies with comparable efficacy, and chose the supervised network for the sake of creating an initial implementation and proof of concept

Since we aim to represent both spatial and temporal features, it is critical to obtain optical flow vector fields as a quantifiable proxy for ciliary movement. Two dimensional motion can be thought of as the projection of three dimensional motion on an image plane, relative to a visual sensor such as a camera or microscope. As such, optical flow represents the apparent motion of pixels within consecutive frames, relative to the visual sensor. To calculate pixel displacement, optical flow algorithms are contingent on several assumptions.

1) Brightness constancy assumes that a pixel's apparent intensity does not change between consecutive frames
2) Small motion assumes that pixels are not drastically displaced between consecutive frames
3) Spatial and temporal coherence assumes that a pixel's neighbors likely exhibit similar motion over gradual time

Solving these constraints yields a series of dense optical flow vector fields; each vector represents a pixel, and the magnitude

and direction of each vector signal the estimated pixel position in the following frame. We refer to Beauchemin and Barron [BB95] for detailed mathematical expression of optical flow derivation. Healthy cilia largely exhibit delicate textural behavior in which patches of cilia move synchronously, slowly, and within a set spatial region near cell boundaries. Additionally, our imaging modality allowed for consistent object brightness throughout sequences of frames. As such, we explored optical flow solutions that focus on brightness constancy, small motion, and spatial coherence systems of equations.

Our optical flow fields are computed using a coarse-to-fine implementation of Horn-Schunck's influential algorithm. Although we tested other methods, namely Farneback [Far03], Lucas-Kanade [LK81], and TV-L1 [SPMLF13], coarse-to-fine Horn-Schunck produced fields more robust to background movement. Horn-Schunck operates by firstly assuming motion smoothness between two frames; the algorithm then minimizes perceived distortions in flow by iteratively updating a global energy function [HS81]. The coarse-to-fine aspect transforms consecutive frames into Gaussian image pyramids; at each iteration, corresponding to levels in the Gaussian pyramids, an optical flow field is generated by Horn-Schunck, and then used to "warp" the images toward one another. This process is repeated until the two images converge. While Horn-Schunck has potential to be noise-sensitive due to its smoothness assumption, we observe that this is mitigated by the coarse-to-fine estimation and hyperparameter tuning. Additionally, we find that this estimation is more computationally and time efficient than its contemporaries.

For further insight into behavioral patterns, we extract first-order differential image quantities from our computed optical flow fields. Estimating linear combinations of optical flow derivatives results in orientation-invariant quantities: curl, deformation, and divergence [FK04]. Curl represents apparent rotation; each scalar in a curl field signaling the speed and direction of local angular movement. Deformation is the shearing about two different axes, in which one axis extends while the other contracts. Divergence, or dilation, is the apparent movement toward or away from the visual sensor, in which object size changes as a product of varied depth. Because our cilia data are captured from a top-down perspective without possibility of dilation, we limit our computation to curl and deformation, similar to Quinn 2011 [QFLC11].

Introduction To Autoencoders

Both the appearance and dynamics modules ultimately rely on a choice of a particular generative model. The chosen model greatly affects the rendered representation, and thus the efficacy of the entire pipeline. Our current choice of generative model is a VAE, an architecture that generates a low-dimensional representation of the data, parameterized as a probability distribution. A VAE can be considered a modified autoencoder (AE). A general AE attempts to learn a low-dimensional representation of the data by enforcing a so-called "bottleneck" in the network. This bottleneck is usually in the form of a hidden layer whose number of nodes is significantly smaller than the dimensionality of the input. The AE then attempts to reconstruct the original input using only this bottleneck representation. The idea behind this approach is that to optimize the reconstruction, only the most essential information will be maintained in the bottleneck, effectively creating a compressed, critical information based representation of the input data. The size of the bottleneck is a hyperparameter which determines how much of the data is compressed.

With this task in mind, an AE can be considered as the composition of two constituent neural networks: the encoder, and the decoder. Suppose that the starting dataset is a collection of n-dimensional points, $S \subset \mathbb{R}^n$, and we want the bottleneck to be of size l, then we can write the encoder and decoder as functions mapping between $\mathbb{R}^n$ and $\mathbb{R}^l$:

$$E_\theta : \mathbb{R}^n \to \mathbb{R}^l, \quad D_\theta : \mathbb{R}^l \to \mathbb{R}^n$$

The subscript θ denotes that these functions are constructed as neural networks parameterized by learanble weights θ. The encoder is tasked with taking the original data input and sending it to a compressed or *encoded* representation. The output of the encoder serves as the bottleneck layer. Then the decoder is tasked with taking this encoded representation and reconstructing a plausible input which could have been encoded to generate this representation, and thus is encouraged to become an approximate inverse of the encoder. The loss target of a AE is generally some distance function (not necessarily a metric) between items in the data space, which we denote as

$$d : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}.$$

Given a single input $x \in S$, we then write the loss function as

$$L_\theta(x) = d(x, D_\theta(E_\theta(x)))$$

where a common choice for d is the square of the standard euclidean norm, resulting in

$$L_\theta(x) = \|x - D_\theta(E_\theta(x))\|^2.$$

The AE unfortunately is prone to degenerate solutions where when the decoder is sufficiently complex, rather than learning a meaningful compressed representation, it instead learns a hash of the input dataset, achieving perfect reconstruction at the expense of any generalizability. Notably, even without this extreme hash example, there is no restraint on continuity on the decoder, thus even if a point $z \in E(S) \subset \mathbb{R}^l$ in the latent space decodes into a nice, plausible data point in the original dataset, points close to z need not nicely decode.

The Variational Autoencoder

A VAE attempts to solve this problem by decoding neighborhoods around the encoded points rather than just the encoded points themselves. A neighborhood around a point $z \in \mathbb{R}^l$ is modeled by considering a multivariate gaussian distribution centered at $\mu \in \mathbb{R}^l$ with covariance $\Sigma \in \mathbb{R}^{l \times l}$. It often suffices to assert that the covariance be a diagonal matrix, allowing us to write $\Sigma = \text{diag}(\sigma)$ for some $\sigma \in \mathbb{R}^l$. While the decision to model neighborhoods via distributions deserves its own discussion and justification, it falls outside the scope of this paper and thus we omit the technical details while referring curious readers to [Doe16] for further reading. Instead, we provide a sort of rationalization of the conclusions of those discussions in the paragraphs that follow. While this is a little backwards, we find it does a better job of communicating the nature of the techniques to most audiences than does touring the complex mathematical underpinnings. The idea of modeling neighborhoods as distributions is implemented by changing the encoder to a new function

$$\tilde{E}_\theta : \mathbb{R}^n \to \mathbb{R}^l \times \mathbb{R}^l, \quad \tilde{E}_\theta : x \mapsto (\mu, \sigma)$$

where μ is the analog to the encoded z in the AE. However now we also introduce σ, which is the main diagonal of a covariance

matrix Σ, which determines how far, and in what direction, to randomly sample around the mean μ. What this means is after encoding, we no longer get a singular point, but a distribution modeling a neighborhood of points as promised. This distribution is referred to as the *posterior distribution* corresponding to x, written as $q(z|x) = \mathcal{N}(\mu, \Sigma)$. We sample from this posterior using the following construction

$$z \sim q_\theta(z|x) \iff z = \mu + \Sigma\varepsilon, \text{ where } \varepsilon \sim \mathcal{N}(0, I_l)$$

to ensure that we may complete backpropagation, since μ, σ are dependent on weights within the network. This is known as the reparameterization trick. Our modified loss is then

$$L_\theta(x) = \|x - D_\theta(z)\|^2.$$

Through this change, over the course of training we obtain a Monte Carlo estimation of the neighborhoods around the embedded points, encouraging continuity in their decoding. This result is still incomplete in that there's no guarantee that the decoder doesn't degenerate to setting σ arbitrarily close to zero, resulting in a slightly more complex AE. Thus we assert that if one were to sample from some predetermined *prior distribution* on the latent space, written as $p(z)$, then the sampled point can be reasonably decoded as a point in the starting data space. To break that down, this means that the portions of the latent space that our model should be best trained on should follow the prior distribution. A common choice for prior, due to simplicity, is the unit-variance Gaussian distribution. This is implemented by imposing a Kullback–Leibler Divergence (KL Divergence) loss between the posterior distributions (parameterized by our encoder via μ, σ) and the prior distribution (in this case $\mathcal{N}(0, I_l)$). Thus our final loss function is

$$L_\theta(x) = \|x - D_\theta(z)\|^2 + \mathrm{KL}(q_\theta(z|x) \,\|\, p(z)).$$

Now we finally have a vanilla VAE, wherein it can not only encode and decode the starting dataset, but it can also decode points in the latent space that it hasn't explicitly trained with (though with no strict promises on the resulting quality). Further improvements to the VAE framework have been made in recent years. To empower the decoder without introducing a significant number of parameters, we implement a spatial broadcast decoder (SBD), as outlined in [WMBL19]. To achieve greater flexibility in terms of the shape of the prior and posterior distributions, we employ the VampPrior in [TW17] with an added regularization term. Both these changes afford us greater flexibility and performance in creating a semantically meaningful latent space. The VampPrior is an alternative prior distribution that is constructed by aggregating the posteriors corresponding to K learned pseudo-inputs $\chi_1, \ldots, \chi_K$. The distribution is given by

$$p(z) = \frac{1}{K} \sum_i^K q(z|\chi_i)$$

This choice of prior optimizes the pipeline for downstream tasks such as clustering and phenotype discovery. We apply a regularization term to the loss to encourage that these pseudo-inputs look as though they could be reasonably generated by the starting dataset 3. Thus our loss becomes

$$\tilde{z}_i \sim q(z|\chi_i)$$

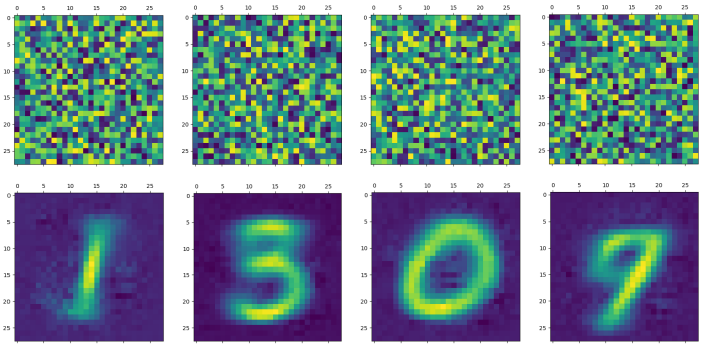

Fig. 3: *Pseudo-inputs of a VampPrior based VAE on MNIST without additional regularization term (top row), and with regularization term (bottom row)*

$$L_\theta(x) = \|x - D_\theta(z)\|^2 + \mathrm{KL}(q_\theta(z|x) \,\|\, p(z)) + \gamma\left(\sum_i^K \|\chi_i - D_\theta(\tilde{z}_i)\|^2 + \mathrm{KL}(q_\theta(z|\chi_i) \| p(z))\right)$$

This has an immediate use in both clustering and semantic pattern discovery tasks. Rather than the embedding $E(S) \subset \mathbb{R}^l$ of the dataset being distributed as a unit gaussian, it is distributed as a mixture of gaussians, with each component being a posterior of a pseudo-input. Consequently, the pseudo-inputs create notable and calculable clusters, and the semantic significance of the clusters can be determined, or at least informed, by analyzing the reconstruction of the pseudo-input responsible for that posterior distribution.

Appearance

The appearance module's role is to learn a sufficient representation so that, frames are reconstructed accurately on an individual basis, and that spatial differences of frames over time is represented with a meaningful sequence of points in the latent space. The latter is the core assumption of the dynamics module.

The appearance module is designed to work with generalized videos, regardless of specific application. Specifically it is designed to take as input singular video frames, augmented with the information generated during the preprocessing phase, including optical flow quantities such as curl. These additional components are included as additional channels concatenated to the starting data, and thus is readily expandable to suit whatever augmented information is appropriate for a given task. In the case of our particular problem, one notable issue is that cilia occupy a small portion of the frame as shown in Figure 6, and thus, the contents of the images that we are interested in exist in some subspace that is significantly smaller than the overall data space. This can result in problems where the neural network optimizes components such as background noise and image artifacts at the expense of the clinically critical cilia information. To remedy this, we leverage the segmentation masks created during the preprocessing phase to focus the network on only the critical portions of the image.

To that effect, we mask the augmented frame data—the raw images concatenated with additional information such as optical flow quantities—using the segmentation masks and train the network on these masked quantities. Mathematically we refer to a single augmented frame with k channels as f, a doubly-indexed collection of vectors, writing the channel information of pixel (i, j) as $f_{i,j} \in \mathbb{R}^k$. We similarly write the generated segmentation mask

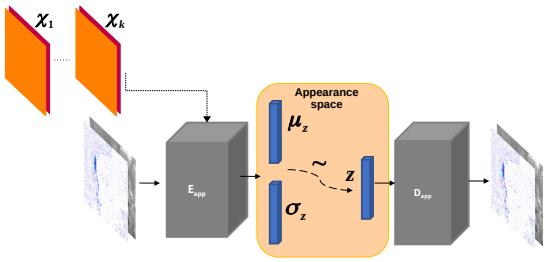

Fig. 4: *The Appearance Module pipeline*

m as a doubly-indexed collection of scalars with $m_{i,j} \in [0,1] \subset \mathbb{R}$, then we construct the augmented frame

$$\tilde{f}_{i,j} := f_{i,j} \cdot m_{i,j}$$

The appearance module ultimately embeds a segmented region of cilia observed in a single frame into what we refer to as the appearance latent space. Due to the temporally static nature of individual frames, this latent space is an encoded representation of *only* the spatial information of the underlying processed data. This spatial information includes aspects such as the shape and distribution of cilia along cells, as well as factors such as their length, orientation and overall shape. These spatial features can be then used in downstream tasks such as phenotype discovery, by drawing a strong connection between apparent patterns in the appearance latent space as and semantically meaningful patterns in the underlying data as well.

Our proposed model for the appearance module uses a variation of ResNet [HZRS15] as an encoder, while employing SBD in the decoder, as well as an upsampling, ResNet-like network in the decoder. Figure 4 shows the training pipeline for the appearance module, the encoder E_{app} is the neural network implementing the variational distribution $q(z|x)$, by estimating parameters to a normal distribution given a certain input frame $x = f$. $q(z|x)$ is therefore $\mathcal{N}(\mu_z, diag(\sigma_z))$ where μ_z and σ_z are the mean and standard deviation vectors of a normal distribution estimated given a certain input frame, or a pseudo-input frame. The decoder D_{app} is trained to reconstruct the input from a sampled $z \sim \mathcal{N}(\mu_z, diag(\sigma_z))$, by minimizing the $L2$ loss between the input and the reconstructed output. The pseudo-inputs $\chi_{1..k}$ are only used during the training, to enforce the prior constraint through a Monte Carlo estimation of the KL divergence, as mentioned earlier.

Dynamics

While the appearance module handles representing the video frames individually under a generative model, the dynamics module is where the temporal behavior is represented. We propose a VAE generative seq2seq module that consists of both an encoder and a decoder to embed the temporal dynamics in a latent semantic space for motion patterns (dynamics). The encoder handles embedding the dynamics of the observed video frames (input) into a latent vector w in the dynamics semantic space $\mathbb{R}^{d_{\text{dyn}}}$. This vector w encodes the dynamics of the video subsequence observed by the encoder. The decoder, then, is able to extrapolate the video into future time-steps by unrolling a sampled latent vector w from the dynamics space into a sequence of vectors $c_{1..k}$. These vectors are not the extrapolated sequence themselves, but instead represent a sequence of changes to be made on a supplied appearance vector $\hat{z}_0$. This vector serves as an initial frame—a starting point for extrapolation—and can be any frame from the video since the vector w encodes the dynamics of the *entire* video. Applying this sequence of change vectors to the initial appearance vector one-by-one, using an aggregation operator $\phi(z,c)$, which could be as simple as vector addition, results in a sequence of appearance vectors $\hat{z}_{1..k}$ which represent the extrapolated sequence. This sequence can then be decoded back into video frames through the decoder of the appearance module D_{app}.

Since the encoder and the decoder of the dynamics module need to process sequences of vectors, they are modeled using a Gated Recurrent Unit (GRU) [CvMG+14] and an LSTM unit, respectively. They are types of RNN with unique architectures that allow them to handle longer sequences of data than a generic RNN could. A GRU cell operates on an input vector x_t, and a hidden state vector s^t at a certain time-step t. Applying a GRU step results in an updated state vector s^{t+1}. An LSTM cell is similar, but it also has an additional output state h^t that gets updated as well like the hidden state.

Figure 5 depicts the pipeline of the proposed dynamics module, showing the encoder steps, sampling from the dynamics space, and the decoder steps. The dynamics encoder GRU, E_{dyn}, starts from a blank state vector $s^0_{\text{enc}} = 0$ that updates every time the appearance vector of the next video frame is fed-in. After feeding in the appearance vector of the final input frame z_n, the state vector s^n_{enc} would encompass information about the motion patterns in the observed video frames $z_{1..n}$, and would then constitute a latent vector in the dynamics semantic space $w = s^n$.

The dynamics decoder LSTM D_{dyn} starts from a latent dynamics vector as its hidden state $s^0_{\text{dec}} = w$, a blank output state vector $h^0_{\text{dec}} = 0$ and an initial supplied appearance vector to act as the beginning output frame. Note that this supplied vector can be any point but the last within the original input sequence, thus we set $\hat{z}_0 = z_i$ for some $i \in \{1,\ldots,n-1\}$. Applying each step results in a change vector $c^{t+1} = h^{t+1}$ (output state vector), that gets applied to the most recent appearance vector in the output sequence to predict the next appearance vector $\hat{z}_{t+1} = \phi(\hat{z}_t, c^{t+1})$, which in turn is used as an input vector to the next LSTM step. The sequence of predicted appearance vectors are then passed through the appearance decoder $D_{\text{app}}(\hat{z}_1), ..., D_{\text{app}}(\hat{z}_k)$, to generate back the video frames. During training time, an L_2 loss is minimized on the predicted k points in the appearance latent space and the true ones.

A prior constraint is imposed on the encoder's output, as per the VAE formulation. Therefore, the size of the state vector of the encoder is $2d_{\text{dyn}}$, composed of both μ_w, and σ_w, such that $w \sim \mathcal{N}(\mu_w, \text{diag}(\sigma_w))$. The prior loss then becomes $\text{KL}(\mathcal{N}(\mu_w, \text{diag}(\sigma_w)) \,||\, \mathcal{N}(0, I))$ and is minimized throughout the training.

It is important to note that the appearance module and the dynamics module are decoupled, such that sampling a different vector w from the dynamics latent space results in different motion dynamics in the extrapolated sequence of video frames despite starting from the same initial supplied frame. As is the case when supplying a different initial output frame as well. To reinforce that, after encoding an input sequence into a dynamics latent vector w, multiple sequences of $k+1$ frames are sampled uniformly from the same training video, where each generated sequence is set to extrapolate from its first frame, and the same dynamics vector w. The $L2$ loss between the extrapolated frames and the remaining k frames in each sequence is minimized with backpropagation.

To summarize, encoder of the dynamics module is trained to extract the motion dynamics from the appearance vectors of

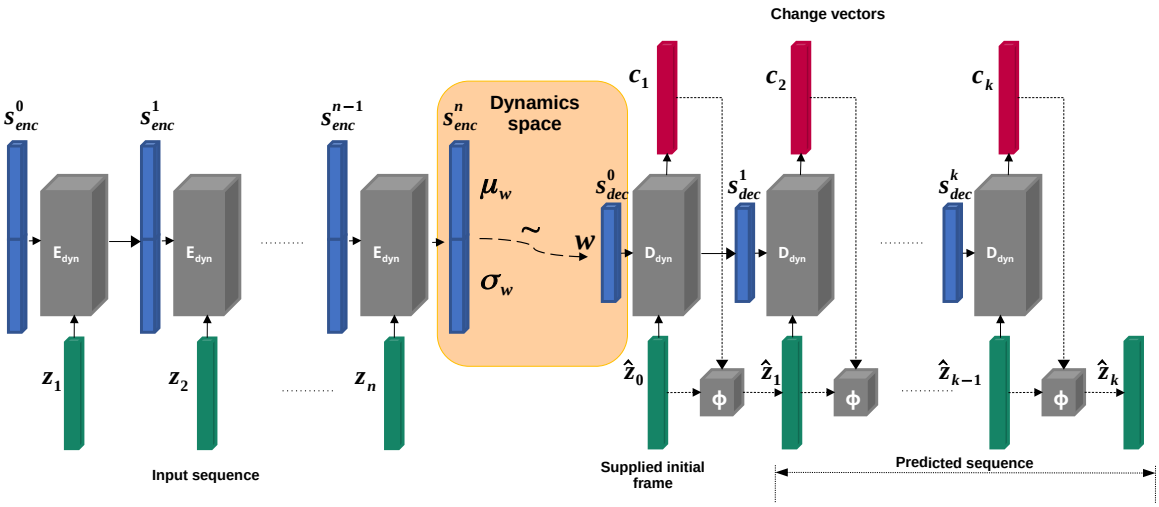

Fig. 5: The Dynamics Module pipeline

a sequence of video frames, and embeds them in a semantic space representing all possible cilia motion patterns. The decoder applies motion patterns from a sampled dynamics vector to a given starting frame, and predicts the appearance vectors to future frames.

Results

The data used for the segmentation task consists of 223 corresponding sets of ground truth masks and high-speed digital microscopy video data. The ground truth masks were manually generated to represent regions of cilia, and the video contains time-series differential image contrast grayscale frames. Each model trained is evaluated by testing intersection over union (IoU), testing precision, and testing accuracy. For every mask generated by FCDN-103, IoU computes the region of overlap between predicted pixels containing cilia and ground truth pixels containing cilia over the joint regions of either prediction or ground truth that contain cilia. Although IoU is typically a superior metric for segmentation evaluation, FCDN-103 is optimized with the goal of minimizing type II error or the presence of false positives because the output masks will be used to narrow representation learning to our region of interest. Thus, we aim to produce segmentation masks with high precision that exclusively identify regions of cilia containing minimal background scene.

We train our FCDN-103 model, written in PyTorch [PGM+19], with an Adam optimizer and cross-entropy loss on one NVIDIA Titan X GPU card. We split our data to consist of 1785 training patches and 190 testing patches. Throughout training and tuning, we experiment with several parameters: standard parameters such as batch size, learning rate, and regularization parameters such as learning rate decay, weight decay, and dropout. We observe optimal performance after 50 epochs, 14 patches per batch, learning rate of 0.0001, learning rate decay of 0.0, and weight decay of 0.0001. This model achieves 33.06% average testing IOU, and 53.26% precision. Figure 6 shows two examples of 128 x 128 test patches with their corresponding ground truth mask (middle) and FCDN-103 generated mask (right); the predicted masks cover more concise areas of cilia than the ground

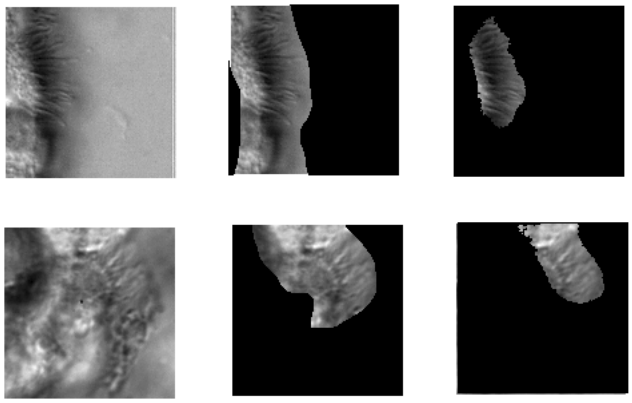

Fig. 6: Segmentation examples from left to right: raw test frame, frame overlain with ground truth segmentation mask, frame overlain with FCDN-103 predicted segmentation maskline

truths and ignore the background in entirety. Previously, Lu 2018 implement a Fully Convolutional DenseNet with 109 layers in a tiramisu architecture trained on ciliary data [LMZ+18]; FCDN-103 achieves an average of 88.3% testing accuracy, outperforming Lu 2018's FCDN-109 by two percentage points.

Curl and deformation fields are extracted from the generated optical flow fields using SciPy's signal and ndimage packages [VGO+20]. Figure 7 shows an example of healthy cilia and its mid-cycle optical flow where vector magnitude corresponds to color saturation; we can reasonably assume that the primary region of movement within optical flow fields will contain healthy cilia. While optical flow fields can potentially provide information on cilia location, we avoid solely using optical flow fields to generate segmentation masks due to the presence of dyskinetic cilia. Identifying stationary cilia is a crucial step in learning ciliary motion phenotype. However, it is possible that optical flow provides insight into both ciliary location and temporal behavior.

During optimization of the appearance module, we observe that cilia do not tend to exhibit a large degree of spatial differences

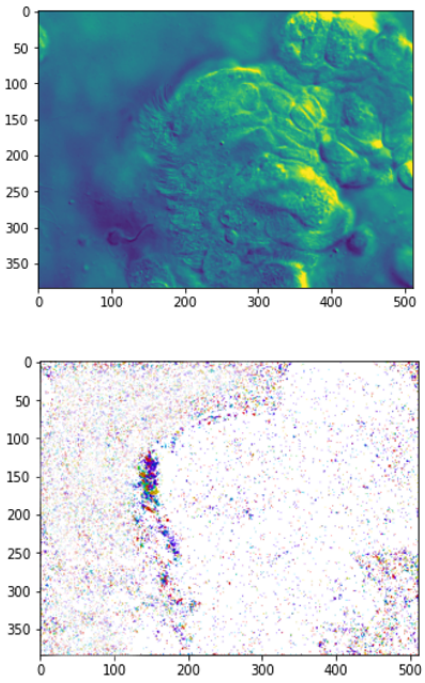

Fig. 7: *Raw imagery and corresponding optical flow visualization*

over time, thus rather than processing every frame of the dataset, we used the NumPy [vdWCV11] library to efficiently sample a fixed number of frames from each video. For testing purposes, we set the number of sampled frames to 40. We sample these frames uniformly throughout the video to ensure that both high-frequency (e.g. cilia beats) and low-frequency (e.g. cell locomotion) spatial changes are represented to ensure that we train on a sufficiently varied base of spatial features.

The entirety of the appearance module's architecture was written using PyTorch. The encoder is a composition of residual blocks, with pixel-wise convolutions and maxpooling operations between them to facilitate channel shuffling and dimensionality reduction respectively, connecting to a fully-connected layer which represents the means and log-variances along each axis of the latent space. We use log-variances instead of the usual standard deviation, or even variance, to guarantee numerical stability, make subsequent calculations such as KL divergence easier, and reduce the propensity for degenerate distributions with variances that approach 0. Since we use a modified VampPrior, the KL Divergence is between a single Gaussian, the posterior, and a mixture of Gaussians, the prior, and thus intractable. In order to estimate this, we employ a Monte Carlo estimation technique, manually calculate the difference in log-probabilities for each distribution at every pass of the loss function, asserting that throughout training these values approximate the ideal KL Divergence. All figures were generated using Matplotlib [Hun07]. The current project can be found at our github repository.

Conclusion

While the initial task of this model was to represent cilia, it also serves as a general framework that is readily extensible to almost any task that involves the simultaneous, yet separate, representation of spatial and temporal components. The specific aim of this project was to develop separate, usable tools which sufficiently accomplish their narrow roles and to integrate them together to offer a more meaningful understanding of the overall problem. While we are still in the early phases of evaluating the entire integrated pipeline as a singular solution, we have demonstrated early successes with the preprocessing module, and have situated the appearance and dynamics modules in the context of modern machine learning approaches well enough to justify further exploration.

Further Research

This generative framework is a foundational element of a much larger project: the construction of a complete library of ciliary motion phenotypes for large-scale genomic screens, and the development of a comprehensive and sophisticated analytics toolbox. The analytics toolbox is intended to be used by developmental and molecular biologists in research settings, as well as clinicians in biomedical and diagnostic settings. By packaging this framework in an easy-to-use open source toolbox, we aim to make sophisticated generative modeling of ciliary motion waveforms available to researchers who do not share our machine learning backgrounds. This pipeline will also serve as a basis and back-end for an exploration into the realm of collaborative, crowd-driven data acquisition and processing in the form of a user-friendly web tool.

More research should also be done to the implementations of each module, and namely their codependencies. For example, how do the quality of segmentation masks in the preprocessing module affect the quality of spatial representation, and consequently dynamic representation? Is there virtue in allowing partial entanglement between the appearance and dynamics module to optimize their joint representation? Can the learned spatial representation influence and inform the preprocessing module in a meaningful way? We hope to explore these questions, and many more, in the near future.

We also encourage the expansion and application of this framework to various other problem contexts. The modular approach to its design ensures portability and adaptability to other projects. The fact that the dynamics module is designed to operate within the abstract latent space of the appearance module means that the appearance module acts as a buffer or converter between the concrete data and the temporal analysis. Consequently, when applying the framework to new projects, only the appearance module need be altered, while the preprocessing module may optionally be adapted or entirely dropped, and the dynamics module preserved.

One example task this pipeline could be adapted to is that of RNA folding analysis. The study of RNA folding patterns is essential in areas such as drug development. One way to model RNA folding is to consider a strand of RNA as a partially-connected point cloud, tracked through time. In this case, the preprocessing module may be forgone, and altering the appearance encoder/decoder to a generic architecture compatible with point clouds, e.g. a geometric neural network or GCNN is all that is necessary. The dynamics module could be readily applied without significant changes.

Acknowledgements

This study was supported in part by Google Cloud. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU used for this research. This study was supported in part by NSF CAREER #1845915.

REFERENCES

[BB95] S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Computing Surveys (CSUR)*, 27(3):433–466, September 1995. URL: http://dl.acm.org/doi/10.1145/212094.212141, doi:10.1145/212094.212141.

[BMO17] Ximena M. Bustamante-Marin and Lawrence E. Ostrowski. Cilia and mucociliary clearance. *Cold Spring Harbor perspectives in biology*, 9(4), 2017. doi:10.1101/cshperspect.a028241.

[CA17] ANDREEA CATANA and ADINA PATRICIA APOSTU. The determination factors of left-right asymmetry disorders- a short review. *Clujul Medical*, 90(2):139–146, 2017. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5433564/, doi:10.15386/cjmed-701.

[CvMG+14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014. arXiv:1406.1078, doi:10.3115/v1/D14-1179.

[DCWS03] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, Feb 2003. URL: https://doi.org/10.1023/A:1021669406132, doi:10.1023/A:1021669406132.

[Doe16] Carl Doersch. Tutorial on variational autoencoders, 2016. arXiv:1606.05908.

[Far03] Gunnar Farnebäck. Two-Frame Motion Estimation Based on Polynomial Expansion. In Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Josef Bigun, and Tomas Gustavsson, editors, *Image Analysis*, volume 2749, pages 363–370. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. Series Title: Lecture Notes in Computer Science. URL: http://link.springer.com/10.1007/3-540-45103-X_50, doi:10.1007/3-540-45103-X_50.

[FK04] Shih Ching Fu and Peter Kovesi. Extracting Differential Invariants of Motion Directly From Optical Flow. In *13th School of Computer Science & Software Engineering Research Conference*, 2004. doi:10.1.1.185.1179.

[FL12] Thomas W Ferkol and Margaret W Leigh. Ciliopathies: the central role of cilia in a spectrum of pediatric disorders. *The Journal of pediatrics*, 160(3):366, 2012. doi:10.1016/j.jpeds.2011.11.024.

[GSC99] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999. doi:10.1162/089976600300015015.

[GZT+14] Andrea S Garrod, Maliha Zahid, Xin Tian, Richard J Francis, Omar Khalifa, William Devine, George C Gabriel, Linda Leatherbury, and Cecilia W Lo. Airway ciliary dysfunction and sinopulmonary symptoms in patients with congenital heart disease. *Annals of the American Thoracic Society*, 11(9):1426–1432, 2014. doi:10.1513/AnnalsATS.201405-222OC.

[HGGRD99] Els Houtmeyers, Rik Gosselink, Ghislaine Gayan-Ramirez, and Marc Decramer. Regulation of mucociliary clearance in health and disease. *European Respiratory Journal*, 13(5):1177–1188, 1999. doi:10.1034/j.1399-3003.1999.13e39.x.

[HLW16] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL: http://arxiv.org/abs/1608.06993, arXiv:1608.06993, doi:10.1109/CVPR.2017.243.

[HS81] Berthold K P Horn and Brian G Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 08 1981. doi:10.1016/0004-3702(81)90024-2.

[Hun07] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science Engineering*, 9(3):90–95, 2007.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL: http://arxiv.org/abs/1512.03385, arXiv:1512.03385, doi:10.1109/CVPR.2016.90.

[JDV+17] Simon Jegou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation. *arXiv:1611.09326 [cs]*, October 2017. arXiv: 1611.09326. URL: http://arxiv.org/abs/1611.09326, doi:10.1109/CVPRW.2017.156.

[KSC17] Celine Kempeneers, Claire Seaton, and Mark A Chilvers. Variation of ciliary beat pattern in three different beating planes in healthy subjects. *Chest*, 151(5):993–1001, 2017. doi:10.1016/j.chest.2016.09.015.

[KW19] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *CoRR*, abs/1906.02691, 2019. URL: http://arxiv.org/abs/1906.02691, arXiv:1906.02691, doi:10.1561/2200000056.

[LK81] Bruce Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision (ijcai). volume 81, 04 1981.

[LMZ+18] Charles Lu, M. Marx, M. Zahid, C. W. Lo, Chakra Chennubhotla, and Shannon P. Quinn. Stacked neural networks for end-to-end ciliary motion analysis. *CoRR*, abs/1803.07534, 2018. URL: http://arxiv.org/abs/1803.07534, arXiv:1803.07534.

[OCH+07] Christopher O'Callaghan, Mark Chilvers, Claire Hogg, Andrew Bush, and Jane Lucas. Diagnosing primary ciliary dyskinesia, 2007. doi:10.1136/thx.2007.083147.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[QFLC11] Shannon Quinn, Richard Francis, Cecilia Lo, and Chakra Chennubhotla. Novel use of differential image velocity invariants to categorize ciliary motion defects. In *Proceedings of the 2011 Biomedical Sciences and Engineering Conference: Image Informatics and Analytics in Biomedicine*, pages 1–4, Knoxville, TN, USA, March 2011. IEEE. URL: http://ieeexplore.ieee.org/document/5872328/, doi:10.1109/BSEC.2011.5872328.

[QZD+15] Shannon P Quinn, Maliha J Zahid, John R Durkin, Richard J Francis, Cecilia W Lo, and S Chakra Chennubhotla. Automated identification of abnormal respiratory ciliary motion in nasal biopsies. *Science translational medicine*, 7(299):299ra124–299ra124, 2015. doi:10.1126/scitranslmed.aaa1233.

[RWH+14] Johanna Raidt, Julia Wallmeier, Rim Hjeij, Jörg Große Onnebrink, Petra Pennekamp, Niki T Loges, Heike Olbrich, Karsten Häffner, Gerard W Dougherty, Heymut Omran, et al. Ciliary beat pattern and frequency in genetic variants of primary ciliary dyskinesia. *European Respiratory Journal*, 44(6):1579–1588, 2014. doi:10.1183/09031936.00052014.

[SPMLF13] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, 3:137–150, July 2013. URL: http://www.ipol.im/pub/art/2013/26/?utm_source=doi, doi:10.5201/ipol.2013.26.

[SS90] Peter Satir and Michael A Sleigh. The physiology of cilia and mucociliary interactions. *Annual review of physiology*, 52(1):137–155, 1990. doi:10.1146/annurev.ph.52.030190.001033.

[TW17] Jakub M. Tomczak and Max Welling. VAE with a vampprior. *CoRR*, abs/1705.07120, 2017. URL: http://arxiv.org/abs/1705.07120, arXiv:1705.07120.

[vdWCV11] Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: a structure for efficient numerical computation. *CoRR*, abs/1102.1523, 2011. URL: http://arxiv.org/abs/1102.1523, arXiv:1102.1523, doi:10.1109/MCSE.2011.37.

[VGO+20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake Vand erPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1. 0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python.

Nature Methods, 17:261–272, 2020. doi:https://doi.org/10.1038/s41592-019-0686-2.

[WMBL19] Nicholas Watters, Loic Matthey, Christopher P. Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes, 2019. URL: http://arxiv.org/abs/1901.07017, arXiv:1901.07017.