This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

Automated Classification of Apoptosis in Phase Contrast Microscopy Using Capsule Network

Aryan Mobiny, *Member, IEEE,* Hengyang Lu, Hien V. Nguyen, *Member, IEEE,*, Badrinath Roysam, *Fellow, IEEE,* Navin Varadarajan

***Abstract*—Automatic and accurate classification of apoptosis, or programmed cell death, will facilitate cell biology research. State-of-the-art approaches in apoptosis classification use deep convolutional neural networks (CNNs). However, these networks are not efficient in encoding the part-whole relationships, thus requiring a large number of training samples to achieve robust generalization. This paper proposes an efficient variant of capsule networks (CapsNets) as an alternative to CNNs. Extensive experimental results demonstrate that the proposed CapsNets achieve competitive performances in target cell apoptosis classification, while significantly outperforming CNNs when the number of training samples is small. To utilize temporal information within microscopy videos, we propose a recurrent capsule network constructed by stacking a CapsNet and a bi-directional long short-term recurrent structure. Our experiments show that when considering temporal constraints, recurrent capsule network achieves 93.8% accuracy and makes significantly more consistent prediction compared to CNNs.**

***Index Terms*—Apoptosis, Capsule network, Cell classification**

I. Introduction

APOPTOSIS is programmed cell death, which occurs in a well-organized manner with a series of biochemical and morphological changes [1]. The process of apoptosis includes cell shrinking, membrane blebbing, deoxyribonucleic acid (DNA) degradation and in some cases depending on the size of the target cells, the formation of apoptotic bodies [2]. Automatic detection or classification of apoptosis has been in great need recently following the development of high-throughput screening assays [3]–[6]. More recently, with the advancements in immunotherapy for the treatment of cancer, single-cell methods like TIMING (time-lapse microscopy in nanowell grids) [7], [8] enable the monitoring of interactions between immune cells [9], [10] and cancer cells in hundreds of thousands of sub-nanoliter wells. These assays have been used to understand the biology of the immune cells, and their interaction with tumor cells, in a number of preclinical settings [11], [12]. Traditional methods using biochemical assays [13] to tag apoptotic cells are widely used, such as fluorophore-conjugated Annexin V [14]–[16] and SYTOX [17]. However, cytotoxicity of chemical assays and phototoxicity due to exposure to light in fluorescent microscopy could adversely affect cell behaviors and lead to cell death [18]. Non-destructive phase contrast microscopy provides a potential label-free solution to apoptosis classification.

A. Mobiny, Hengyang Lu, H. V. Nguyen and Badrinath Roysam are with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX, 77004 USA (e-mail: amobiny@uh.edu, hlu9@uh.edu, hienvnguyen@uh.edu, and broysam@central.uh.edu).

Navin Varadarajan is with the Department of Chemical and Biomolecular Engineering, University of Houston, Houston, TX, 77004 USA (e-mail: nvaradarajan@uh.edu).

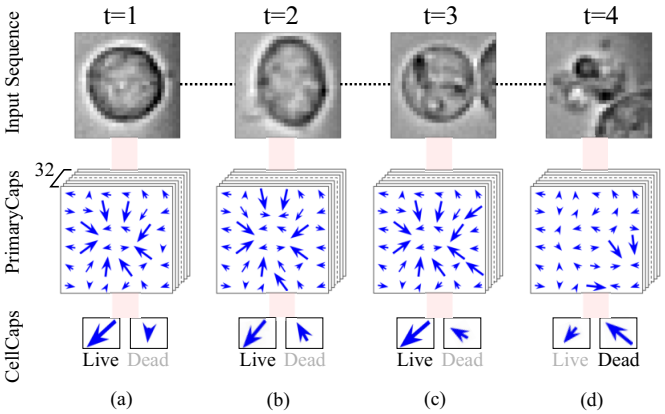

Fig. 1. A sample illustration of capsule activations at different layers for the network proposed in [19] **Top:** four sample images associated to life-span of a specific target cell when it is: (a) static, (b) moving horizontally, (c) approached by an effector cell, and (d) killed by the effector cell. **Middle:** The corresponding grids of child capsule activations in *PrimaryCaps* layer which includes 32 grid of capsules. Each grid is called a *Capsule Type* and contains 6×6 capsules. Each capsule at this layer is a group of 8 neurons, thus outputs an 8-dimensional vector shown by a blue arrow. Theoretically, each capsule *type* is responsible for detecting a specific feature in the input image. In this example, the front capsule type is recognizing the edges of cell body. **Bottom:** Final capsule layer called *CellCaps* which contains two capsule types; one for each class (dead vs. live). Prediction is made according to the length of output activation vectors.

While apoptotic patterns have visual saliency, label-free apoptosis classification is challenging from the algorithmic perspective. The disparity in appearance exists between apoptotic cells. This is due to either characteristic of individual cells or the different times the images are taken. For example, the apoptotic cells at the membrane blebbing stage may look quite different from the ones at DNA degradation stage with chromatin condensation. Previous works on label-free apoptosis classification or detection are mostly based on simple heuristics, e.g., a terminated cell tracks could possibly indicate an apoptosis event [18]. More advanced method models the imaging physics of phase contrast microscopy to restore the original images [20] and identifies apoptosis using handcrafted features including changes of brightness and contrast as well as local binary patterns [18]. All these previous methods are either too simple and subject to multiple sources of interferences or have a comparatively complex model which still has limitations dealing with large variances.

In the last few years, deep learning methods, such as convolutional neural networks (CNNs), have shown remarkable performance for a wide range of computer vision tasks.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

Specifically related to the classification problem, AlexNet [21], ResNet [22], and DenseNet [23] achieved the state-of-the-art recognition level and have been adopted as the desired models for various medical imaging tasks. Recently, many researchers work on designing deep networks which are more suitable for their tasks. This leads to more complex models with a huge number of parameter and hyper-parameters to be tuned which makes the overall network harder to be optimized. This motivates the development of new techniques to address the fundamental limitations of the current CNN structures.

One fundamental drawback of CNNs is the way they route information between layers. Routing is the mechanism of relaying information from one layer to the next layer in the network. CNNs currently perform routing via pooling layers, most commonly being max-pooling and average pooling. However, pooling is a naive way of routing as it discards information about the precise location and pose of the entity within the region which can be valuable for the classification purpose. This gives a limited translation invariance in which a feature can slightly move within the pooling window and still does not change the network's output.

Moreover, while CNNs are translation invariant (and partially rotational invariant while using data augmentation), they are unable to identify the position of one object *relative* to another [24], [25]. They can only identify if the object exists in a certain region or not. Therefore, it makes it difficult to correctly identify objects with spatial relationships between features. For example, a set of randomly assembled face parts might look like a face to a CNN as it sees all the key features.

Recently, Sabour et al. [19] introduced a new architecture called Capsule Network (CapsNet in short) to address CNNs shortcomings. The idea is to encode the relative relationships (e.g., locations, scales, orientations) between local parts and the whole object. Encoding these relationships equips the model with a built-in understanding of the 3-dimensional space. This makes CapsNets more *invariant to viewpoint* which enables them to recognize objects from different viewpoints not seen in the training data.

While CapsNets are shown to achieve promising performance in some tasks, they do not scale well to high-dimensional data. The required number of parameters grows rapidly which makes the routing computationally expensive and intractable. Moreover, it remains unclear whether CapsNets are appropriate for dealing with temporally varying data. Our paper makes the following contributions:

1) We propose an efficient variant of capsule networks. Our networks achieve higher computational efficiency by sharing the transformation matrices across capsules of the same type, locally-constraining the dynamic routing, and imposing a spatially consistent voting mechanism. These changes dramatically reduce the number of parameters and enable us to make capsule networks deeper to work with larger images. Extensive experimental results show that the proposed networks compare favorably to CNNs when the training set is large, and significantly outperform CNNs for small-size datasets.
2) We investigate the performance of capsule networks for temporal classification. The proposed architecture consists of a capsule network stacked with a bi-directional long short-term recurrent network. The proposed architecture extracts richer features than those from CNNs, illustrated by higher classification accuracy and temporal consistency.
3) The proposed CapsNets are extensively evaluated against CNNs on the apoptosis classification task. We provide visualization to compare important image features captured by CNNs and CapsNets. We observe that CNNs tend to make decisions based on a small region in the image while CapsNets collectively make predictions based on a larger image region.

The rest of this paper is organized as follows: works related to capsule networks and its variants are presented in Section II. Section III explains the original capsule network, its limitations, and the proposed network architecture. Section IV describes the dataset used in this study. Experimental results are presented in Section V. Section VI concludes the paper with future research directions.

II. Related work

Since CNNs are not efficient in capturing the hierarchical relationship between the entities in the image, CapsNets are introduced as a structure capable of encoding the part-whole relationship. CapsNets employ a dynamic routing mechanism to determine where to send the information. Sabour et al. [19] successfully used this algorithm for training the network on hand-written images of digits (MNIST) and achieved state-of-the-art performance. In [26], Hinton introduces matrix capsules with expected-maximization (EM) routing to encode the relationship between entities. Bahadori [27] proposes a novel variation of capsule networks, called spectral capsule networks, that is more stable than the capsule network with EM routing, and converges faster as well. Multi-scale CapsNet [28] is another variation of capsule networks which employs a two-stage processing for extracting and encoding the hierarchy of features. It also uses a modified dropout to improve the robustness of the network and achieves competitive performance on FashionMNIST and CIFAR-10 datasets.

CapsNets has also been adopted into various tasks and fields. CapsuleGAN [29] is proposed as a generative model which uses a CapsNet discriminator to replace the standard CNN. Siamese CapsNet [30] is introduced as a variant utilized in a pairwise learning task. Capsule networks have been recently used in medical image analysis tasks and achieved remarkable results. Our preliminary work proposed Fast CapsNets [31] which achieve promising results in the lung cancer screening task. Lalonde et al. [32] proposed a convolutional-deconvolutional capsule network, which expands capsule networks to segment pathological lungs from low dose CT scans. These studies show the potential of CapsNet to scale to large and volumetric images. However, the original CapsNets with dynamic routing experience unstable training when the number of layers increases. This has also been noticed in [33]. Our paper introduces a deeper capsule network in which, similar to CNNs, the hidden capsules are connected locally and trainable parameters are shared. The proposed network maintains the classification power of capsule networks while requiring a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

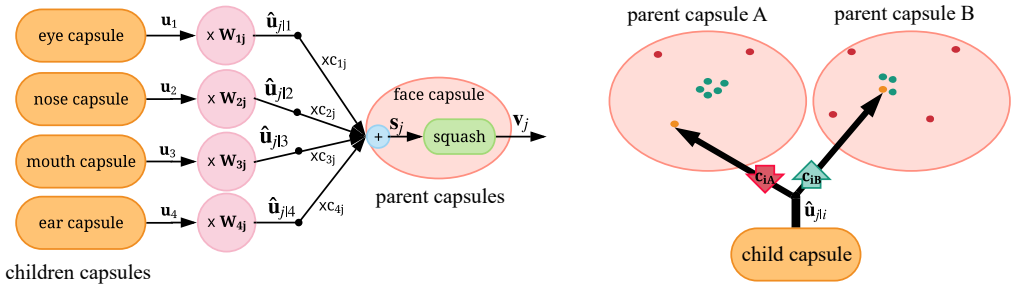

Fig. 2. Connections between the child and parent capsules in a face detection example. **Left:** Each child capsule encodes the detailed information about the lower-level entities (such as nose, mouth, etc.) in its output $\mathbf{u}_i$. Then every child predicts the output of the parent capsule. In our example, the mouth capsule tries to answer: "*how is the face going to look like given the mouth pose*". It makes this prediction by computing the dot-product of a transformation matrix ($\mathbf{W}_{3j}$) with its own activation vector, $\mathbf{u}_3$. **Right:** Routing by agreement: dynamic routing will ensure that a child capsule will send information to the parent capsules that *agree* with its prediction. In other words, if several children capsules point at the same pose of the face, then it must be a face there.

dramatically smaller number of parameters. It performs favorably on large images even when the number of training samples is small. Most of the existing work does not provide a deep understanding of how CNNs and CapsNets differ in making their decisions. This paper gives a visualization to help us understand this better. Finally, we extensively evaluate the performance of the proposed capsule network on a temporal classification task, which have not been sufficiently studied in the related prior works.

III. Methodology

A. Background on Capsule Network

Capsule Computation: A capsule is defined as a group of neurons whose outputs form an *activation vector*. They predict the presence and the pose parameters of a particular object at a given pixel location. The direction of an activation vector captures the object's pose information, such as location and orientation, while the length (a.k.a norm or magnitude) of the activation vector represents the probability that an object of interest exists. For instance, if we rotate an image, the activation vectors also change accordingly, but their lengths stay the same. This property is usually referred to as *equivariance*. Fig. 2 illustrates the way CapsNets route information from one layer to another layer, using face detection as an example. The length of the activation vector from a lower-level capsule $(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_I)$ encodes the existence probability of its corresponding entity (e.g. eyes, nose, and mouth). The direction of the vector encodes various properties of the entity, such as its size, orientation, position, etc.

The relationship between i-th capsule in a lower layer and j-th capsule in the next higher layer is encoded using a linear transformation matrix $\mathbf{W}_{ij}$. The information is propagated as: $\hat{\mathbf{u}}_{j|i} = \mathbf{W}_{ij}\mathbf{u}_i$. The vector $\hat{\mathbf{u}}_{ij}$ represents the belief of i-th capsule in a lower layer about j-th capsule in the higher layer. In our example, $\hat{\mathbf{u}}_{j|1}$ represents the predicted pose of the face according to the detected pose of the nose. During the training, the network will gradually learn a transformation matrix for each capsule pair to encode the corresponding part-whole relationship.

Dynamic Routing: Having computed the prediction vectors, the lower-level capsules then route their information to parent capsules that agree the most with their predictions. The mechanism that ensures that the outputs of the child capsules get sent to the proper parent capsules is named *dynamic routing*. Let c_{ij} denotes the routing coefficient from i-th capsule in the lower layer to j-th capsule in the higher layer, where $\sum_j c_{ij} = 1$ and $c_{ij} \geq 0, \ \ \forall j$. When $c_{ij} = 1$, all information from i-th capsule will be sent to j-th capsule, whereas when $c_{ij} = 0$, there is no information flowing between the two capsules. Dynamic routing method iteratively tunes the c_{ij} coefficients and routes the child capsules' outputs to the appropriate capsule in the next layer so that they get a cleaner input, thus determining the pose of the objects more accurately.

The right panel of Fig. 2 shows a lower-level capsule (e.g. nose capsule) making a decision to send its output to the parent capsules. This decision is made by adjusting the routing coefficients, c_{ij}, that will be multiplied by the prediction vectors before sending it to high-level capsules. CapsNets compute the parent capsules activation vector ($\mathbf{v}_j$) and routing coefficients as follows:

$$\mathbf{v}_j = \frac{||\mathbf{s}_j||^2}{1+||\mathbf{s}_j||^2}\frac{\mathbf{s}_j}{||\mathbf{s}_j||}, \quad \mathbf{s}_j = \sum_i c_{ij}\hat{\mathbf{u}}_{j|i}, \tag{1}$$

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_k \exp(b_{ik})}, \quad b_{ij} \leftarrow b_{ij} + \hat{\mathbf{u}}_{j|i}.\mathbf{v}_j. \tag{2}$$

The output of each parent capsule $\mathbf{v}_j$ is computed as the weighted sum of all predictions from child capsules (i.e. $\mathbf{s}_j$), then passed through a *squash* non-linearity. Squashing makes sure that the output vector has a length no more than 1 (so that its length can be interpreted as the probability that a given feature being detected by the capsule) without changing its direction. each parent capsule receives predictions ($\hat{\mathbf{u}}_{j|i}$) from all children capsules. These vectors are represented by points in Fig. 2. The dynamic routing mechanism will increase the routing coefficient to parent capsule-j by a factor of $\hat{\mathbf{u}}_{j|i}.\mathbf{v}_j$ whose value increases for the similar vectors. Thus a child capsule will send more information to the parent capsule whose output $\mathbf{v}_j$ is more similar to its prediction $\hat{\mathbf{u}}_{j|i}$.

Capsule Network Architecture: The original capsule network contains two main parts: *encoder* and *decoder*, depicted in the first two figures of [19]. The encoder contains three layers: two convolution layers and one fully-connected layer.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

The first layer is a standard convolution layer with 256 filters of size 9×9 and stride 1, followed by ReLU activation. The next layer is a convolutional capsule layer called the *PrimaryCaps* layer. Capsules are arranged in 32 channels (commonly referred to as 32 capsule types) where each primary capsule applies 8 convolutional filters of size 9×9 and stride 2 to the input volume. Therefore, each primary capsule sees the outputs of all 256 (the whole input depth) $\times$ 81 (that falls inside the 9×9 filter) units of the first convolutional layer. In this setting, all PrimaryCaps in each of the 32 channels share their weights with each other and each capsule outputs an 8-dimensional vector of activations. The last layer is called *DigitCaps* layer which has one 16D capsule per class. Routing takes place in between these capsules and all PrimaryCaps, encoding the input into 16-dimensional activation vector of instantiation parameters. The lengths of these prediction vectors are used to determine the predicted class.

The decoder tries to reconstruct the input from the final capsules, which will force the network to preserve as much information from the input as possible across the whole network. This effectively works as a regularizer that reduces the risk of over-fitting and helps generalize to new samples. In the decoder, the 16D outputs of the final capsules are all masked out (set to zero) except for the ones corresponding to the target (while training) or predicted (while testing) class. They proposed using a three-layer feed-forward neural network with 512, 1024, and 784 units to reconstruct the input image.

B. Capsule Network with Convolutional Routing

In the original capsule network described above, all PrimaryCaps are linked and route information to all DigitCaps. The routing coefficients are computed by the iterative dynamic routing process and the transformation matrices $\mathbf{W}_{ij}$ are trained through back-propagation. While dynamic routing has been shown to improve the classification accuracy of CapsNet [19], this operation is computationally expensive and does not scale well to high dimensional data. Technically, the number of *DigitCaps* is bounded by the number of classes. However, the number of PrimaryCaps increases with the size of input images. This will dramatically increase the required number of routing coefficients (c_{ij}), thus do not scale to large input images. Moreover, according to [19], each child-parent capsule pair requires a unique transformation matrix $\mathbf{W}_{ij}$. Therefore, both the number of non-trainable (c_{ij}) and trainable ($\mathbf{W}_{ij}$) parameters increases with the input size. This makes dynamic routing the *bottleneck* of the capsule networks.

A naive solution is to reduce the number of PrimaryCaps by changing the hyper-parameters of the preceding layers; e.g. increasing the strides of the convolutional layers or the number of strided convolutional layers. This practically results in more information loss and causes a significant drop in the classification accuracy. Instead, we propose the following modifications to the routing mechanism which dramatically reduces the number of parameters in the network while enhancing the information routing stability and the overall model prediction accuracy.

Convolutional Dynamic Routing Inspired by the local connectivity of the neurons in CNNs, [32] proposed locally-constraining the information routing; meaning that only child capsules within a user-defined kernel centered at position (x, y) are able to route to parent capsules placed at (x, y) position of the next layer. Therefore, we call capsule layers with locally-constrained routing *convolutional capsule layer*, compared with the *fully-connected capsule layer* proposed in the original structure. Similar to the idea of sharing weights in CNNs, we share the learnable transformation matrices across capsules of the same type. Here, capsule *type* is referred to a capsule (or a group of capsules) detecting a specific feature in the input. In the original capsule network proposed in [19], there exist 32 capsule types in the PrimaryCaps layer and one capsule type per class in the DigitCaps layer. Transformation matrix sharing will, therefore, reduce the required number of $\mathbf{W}_{ij}$ matrices by a factor equal to the number of members of each capsule type (e.g. we require only 32 matrices instead of $6 \times 6 \times 32$ matrices for the original CapsNet of Fig. 1).

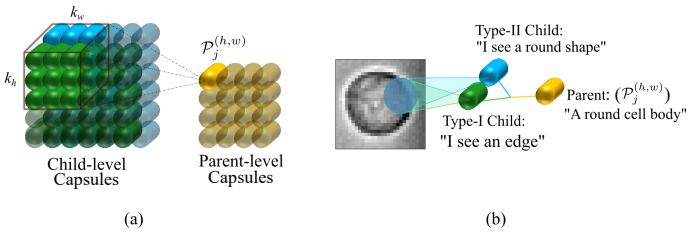

Fig. 3. Depicts the techniques incorporated to solve the memory burden and parameter explosion and make the capsule network deeper. Different capsule types are shown in different colors. (a) Convolutional dynamic routing in which the transformation matrices are shared across capsules of the same type, and a parent capsule positioned at point (h, w) receives information from a sub-grid of capsules bounded in a user-defined kernel of size $(k_h \times k_w)$ centered at (h, w). (b) The imposed consistent voting technique where children capsules at the same spatial location (and of different types) are restricted to agree on their votes to the parent capsule.

Assume having a $H^l \times W^l$ grid of d^l-dimensional capsules of T^l different types at layer l. These capsules form a $H^l \times W^l \times T^l$ grid. Likewise, we have a $H^{l+1} \times W^{l+1} \times T^{l+1}$ grid of d^{l+1}-dimensional capsules at layer $l+1$ where $H^{l+1} \times W^{l+1}$ is the spatial dimension and T^{l+1} the number of capsule types. Let $\mathcal{P}_j^{(h,w)}$ be a parent capsule of type $j \in \{1 : T^{l+1}\}$ located at (h, w) where $h \in \{1, ..., H^{l+1}\}$ and $w \in \{1, ..., W^{l+1}\}$. This capsule receives information from a sub-grid of child capsules of all types, confined in a user-defined kernel of size $k_h^l \times k_w^l$. This connection is illustrated in Fig. 3 where $\mathcal{C}_{1:T^l}^{(y,x)}$ is the sub-grid of children capsules of all types centered at (h, w) and $y \in \{(h - k_h), ..., (h + k_h)\}$ and $x \in \{(w - k_w), ..., (w + k_w)\}$. Each parent capsule $\mathcal{P}_j^{(h,w)}$ receives a prediction vector, $\hat{\mathbf{u}}_{j|i}^{(y,x)}$ per child capsule type:

$$\hat{\mathbf{u}}_{j|i}^{(y,x)} = \mathbf{W}_{ij} \times \mathbf{u}_i^{(y,x)}, \qquad \forall\, i \in \{1 : T^l\},\ x,\ y \tag{3}$$

where $\mathbf{W}_{ij}$ is the learned transformation matrix for type-i to type-j capsules and $\mathbf{u}_i^{(y,x)}$ the output of children capsules. As noted, $\mathbf{W}_{ij}$ is defined for each capsule type and does not depend on the spatial location. This is because it is shared across locations (similar to convolutional filters) and scans the whole capsule map. These transformation matrices are learned during training by back-propagation. Having computed the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

prediction vectors using eq. (3), the final input to each parent capsule $\mathcal{P}_j^{(h,w)}$ is the weighted sum over the predictions:

$$\mathbf{s}_j^{(h,w)} = \sum_i c_j^{(y,x)} \, \hat{\mathbf{u}}_{j|i}^{(y,x)} \,, \qquad \forall \, j = \{1 : T^{l+1}\} \tag{4}$$

which shows that only children capsules within the corresponding kernel are routing to the parent capsule. Here, $c_j^{(y,x)}$ is the routing coefficient of the child capsule positioned at (x, y) (which does not depend on the child capsule type i according to consistent voting mechanism explained below) to its corresponding parent of type j. Finally, the parent capsules activation is computed as:

$$\mathbf{v}_j^{(h,w)} = \frac{||\mathbf{s}_j^{(h,w)}||^2}{1 + ||\mathbf{s}_j^{(h,w)}||^2} \frac{\mathbf{s}_j^{(h,w)}}{||\mathbf{s}_j^{(h,w)}||} \tag{5}$$

which is similar to eq. (2) with the $\mathbf{s}_j$ in the fully-connected capsule layer replaced by $\mathbf{s}_j^{(h,w)}$ in the convolutional capsule layer.

Consistent Voting: While each child capsule can freely vote to a parent capsule, in [31] we proposed restraining capsules to agree with capsules of other types at the same spatial location.

$$c_{ij} = c_{kj}, \quad \forall i, k \in \mathcal{S} = \{i, k \mid loc(i) = loc(k)\} \tag{6}$$

where $loc()$ is the function converting a capsule index to its pixel location and j is the index of the parent capsule to route into. Intuitively, even though these capsules detect different features from the input, they look at the same entity (due to their same spatial location). Therefore, they should agree upon what they see and vote to parent capsules accordingly. Our experimental results show that incorporating this technique significantly boosts the stability of the routing mechanism performed in the convolutional capsule layers and improves the overall network convergence.

These coefficients are determined by the dynamic routing algorithm and computed as:

$$b_j^{(y,x)} \leftarrow \text{mean}_i(b_{ij}^{(y,x)} + \hat{\mathbf{u}}_{j|i}^{(y,x)} \cdot \mathbf{v}_j^{(h,w)}) \tag{7}$$

$$c_j^{(y,x)} = \frac{\exp(b_j^{(y,x)})}{\sum_j \exp(b_j^{(y,x)})} \tag{8}$$

where $c_j^{(y,x)}$ is the probability that the prediction vector $\hat{\mathbf{u}}_{j|i}^{(y,x)}$ to be routed to the parent capsule $\mathcal{P}_j^{(h,w)}$.

C. Deep Capsule Network Architecture

Applying the ideas mentioned above allow us to make the original capsule network deeper (with adding more convolutional capsule layers) and solve the memory burden and parameter explosion, thus makes the capsules work on large images. The final structure of the deep capsule network used in our cell classification task is presented in Fig. 4.

Similar to the original capsule network, this network contains two main parts: *encoder* and *decoder*. The encoder contains four layers. The first layer is a standard convolution layer with 128 filters of size 5×5 and stride 2, followed by ReLU activation. The next layer is a convolutional capsule layer in which capsules are arranged in 8 channels (i.e. 8 capsule types). Each capsule applies 16 convolutional filters

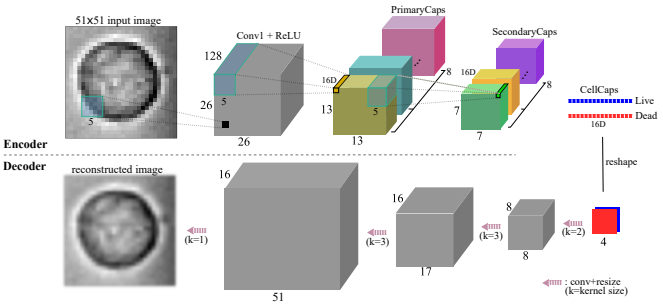

Fig. 4. Visual representation of the proposed deep capsule network for classifying live vs. dead cells. Different colors show different capsule types.

of size 5×5 and stride 2 to the whole input depth. In this setting, all capsules in each of the 8 channels share their weights with each other and each capsule outputs a 16-dimensional vector of activations. The third layer is another convolutional capsule layer where each of its 16-dimensional capsules receives information from only a $5 \times 5 \times 8$ grid of children capsules (locally-constrained routing). This is similar to the local connection of neurons in a CNN. The last layer is called *CellCaps* layer which has one 16D capsule per class. Routing also takes place between these capsules and all capsules in the previous layer, encoding the input into the 16D activation vector of instantiation parameters. The lengths of these vectors are used to determine the predicted class.

Capsule network uses an additional reconstruction loss as regularization to prevent over-fitting during learning the network's parameters [19]. This encourages the final capsules to encode as much information from the input as possible. The reconstruction is done by feeding 16D output of the final capsules to a three-layer feed-forward neural network. Clearly, given the larger input images, we need more neurons in the feed-forward network to reconstruct the input. This means that the required number of parameters also increases dramatically. For example, for reconstructing a 51×51 input image out of the two 16-dimensional *CellCaps* activations, we need 32, 1000, and 2601 (51×51) neurons, thus more than 2.6 million parameters (weights and biases) to be trained which is very large. To solve this, we propose using a convolutional decoder to serve as the training regularization. We first mask the 16-dimensional activity vector of the wrong *CellCap*, then reshape the activity vectors into two 4×4 feature maps which are to be fed into the convolutional layers. Here, we used four convolutional layers with 8, 16, 16, and 1 filters of sizes 2, 3, 3, and 3 respectively. After each convolution, the resulted maps are resized to double the size of the feature maps. This structure has much fewer parameters (about 4K) and significantly outperforms the feed-forward network in the reconstruction task.

Loss Function: We used the margin loss proposed in [19] to enforce the activation vector of the top-level capsule k ($k \in \{0, 1\}$) to have a large length if and only if the cell type (live vs. dead) is present in the image. The total margin loss is the sum of the losses for each *CellCap* and is computed as:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

$$L_{\text{margin}} = \sum_k \Big[T_k \max(0, m^+ - ||\mathbf{v}_k||)^2 + \lambda(1 - T_k) \max(0, ||\mathbf{v}_k|| - m^-)^2 \Big] \quad (9)$$

where $T_k = 1$ iff a cell of class k is present, and $m^+ = 0.9$, $m^- = 0.1$, and $\lambda = 0.5$ as set in [19]. Sum of squared differences between the input and reconstructed images is used as the reconstruction loss and the total loss is computed as:

$$L_{\text{Total}} = L_{\text{margin}} + \alpha L_{\text{reconstruction}} \quad (10)$$

where α is set to 0.0005 to scale down the reconstruction loss so that it does not dominate the margin loss during training.

D. Sequential data with temporal constraints

Dealing with the classification of time-lapse image frames, we must ideally consider the temporal dependencies between individual frames. Ignoring such dependencies and treating images as if they are independent yields noisy predictions. In the case of our time-lapse data, there exist at most one apoptosis event happening in a time-lapse sequence (i.e. a cell death occurs in the sequential frames). Thus the prediction should look like a step function in which the jump is the apoptotic transition moment (programmed cell death) which is irreversible; once a cell is dead, it will remain dead.

Given the time-lapse frames, we reformulate the temporal-constrained apoptosis classification as a sequential classification problem. There is a large number of methods for classification of sequential data, among which Hidden Markov Models (HMMs) [34], Conditional Random Fields (CRFs) [35], and Recurrent Neural Networks (RNNs) are the most popular ones. RNNs are equipped with an "internal memory" that captures the information about what has been seen so far. Long short-term memory (LSTM) [36] model is introduced as a modification of RNN with a hidden memory cell and dedicated gated logic. This makes LSTMs capable of learning to keep, update, or forget the memory cell state according to the context and incoming data. Moreover, unlike the vanilla RNNs, LSTMs are capable of encoding long-term dependencies.

Here, instead of feeding cell patches directly to LSTM, we use the pre-trained networks (both CNN and CapsNet) as feature extractors. However, instead of freezing their parameters, we let them get trained and fined-tuned along with the LSTM parameters. Other than accuracy, model predictions are also evaluated and compared according to the oscillations. We typically expect the predictions to look like a step function (same as the ground truth) and do not fluctuate much. This is quantified using the mean absolute ups and downs error (MAUDE) metric. For a sequence i (sequence of L frames), suppose $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i \in \mathbb{R}^L$ are the vectors of ground truth labels and model predictions respectively. Function $\mathcal{UD}(.) : \mathbb{R}^L \rightarrow \mathbb{R}$ is the function counting the number of ups and downs (from 0 to 1 and vice versa) in the sequence. Then the metric is defined as:

$$\text{MAUDE} = \frac{1}{N} \sum_{i=1}^{N} |\mathcal{UD}(\mathbf{y}_i) - \mathcal{UD}(\hat{\mathbf{y}}_i)| \quad (11)$$

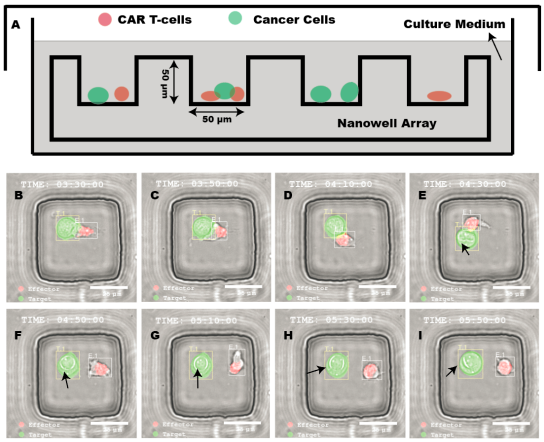

Fig. 5. Nanowell array demonstration and time-lapse microscopy illustrating a CAR T-cell **E1** killing a cancer cell **T1** in one nanoliter well. (B)-(D) showing the CAR T-cell started contacting the cancer cell; (E)-(I) indicating that the cancer cell was killed as we can see clear lumenal changes in cell nucleus pointed by black arrows. Scale bar: 25 μm.

where N is the total number of test sequences used in the evaluation phase. We also compare the quality of model prediction in terms of their accuracy in detecting the death time. This is measured by the mean absolute death-time error (MADTE) computed as:

$$\text{MADTE} = \frac{1}{N} \sum_{i=1}^{N} |\mathcal{DT}(\mathbf{y}_i) - \mathcal{DT}(\hat{\mathbf{y}}_i)| \quad (12)$$

where $\mathcal{DT}(.) : \mathbb{R}^L \rightarrow \mathbb{R}$ is the function that returns the death time, i.e. index of the first frame labeled '1'.

IV. Dataset

Cell Culture and Imaging: Cell images in this work are derived from 4 datasets in TIMING project. Specifically, CAR T-cells and NALM6 cell-line cancer cells were labeled with fluorescent bio-markers PKH67 and PKH26 respectively. CAR T-cells and cancer cells are then loaded to nanowells at a concentration of 106 cells/mL sequentially. The entire nanowells were immersed in the media containing fluorochrome (AlexaFluor 647, Invitrogen) conjugated Annexin V. The cell incubator was fixed at 37°C with 5% CO_2. Cells were monitored in both phase contrast and fluorescent modes using Carl Zeiss Observer Z1 fitted with Hamamatsu sCMOS camera using a 20x 0.8 NA objective for 6 hours at 5-min intervals. Fig. 5 depicts the structure of a nanowell array and frames captured from a nanowell demonstrating an apoptotic cell.

TIMING Pipeline and the Ground Truth: TIMING pipeline consists of a set of algorithms for nanowell detection, cell segmentation and cell tracking, with which, we are able to identify single-cell status at each time point. For example, we cropped cell patches around their centers with size 51×51 pixels. For the purpose of apoptosis classification, we only cropped images in phase contrast and Annexin V fluorescent channels. Instead of annotating each cell patch manually by looking at its phase contrast or fluorescent channels, we attained the binary labels ('0' for live and '1' for dead) using simple threshold method. Note that the Annexin V intensity threshold for apoptotic cells was generally consistent in a certain dataset

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

due to variations in imaging conditions and fluorescent biomarker concentrations across different experiments, we chose different threshold values for the 4 datasets before applying the threshold operation for all. Labels of all the cell patches are not always clean using simple threshold method. Multiple sources of annotation errors do exist, such as incorrect calculation of ANNEXIN V fluorescent intensity due to segmentation errors and spectral leakage between fluorescent channels.

We collected 92,000 cancer cell patches of live and apoptotic samples in total. For the sequential experiment, 9818 time-lapse sequences of cancer cell patches were collected where each sequence contains 72 consecutive crops from a delineated cell track. Performance is measured using 10-fold cross-validation where each train and test set is selected from different nanowells to ensure independence.

V. Experimental Results and Discussion

A. Apoptosis Classification

The proposed capsule network is used for classifying live and dead cells. We compared its performance with that of the original capsule network proposed in [19], along with some of the famous structures, namely AlexNet [21], ResNet-50 [22], and DenseNet-BC [23]. These networks are modified to make them compatible with our data and improve their performance. All networks are trained using images of size 51×51 pixels, except the capsule network which was trained on the same images down-sampled to 28×28 pixels. The final architectures are the result of random search over hyper-parameters such as filters' sizes, number of channels, and dense/residual blocks.

For all networks, training is done using ADAM optimizer [37] with an initial learning rate of 10^{-4} and mini-batches of size 16. We annealed the learning rate exponentially from its initial value to the minimum of 10^{-5} over the course of training. Cross-entropy loss function is used for CNNs, while capsule networks are trained using the margin-reconstruction loss combination. We perform data augmentation by randomly rotating images around the center of images with the maximum rotation degree of 180^o. Afterward, pepper noise was added to the rotated images. It is similar to applying dropout to the visible layer, i.e., the input. The dropout rate is set to 5% and all the model weights are initialized using Xavier initialization. Moreover, L2-regularization and dropout technique are applied to all weights and convolutional layers of the CNNs respectively to ensure that no over-fitting happens.

Test prediction results for the live vs. dead cell classification is presented in Table I for the various network architectures. As can be seen, the proposed deep capsule network achieves the best performance with almost one-third of parameters used in DenseNet-BC. Using the techniques explained in Section 3 and adding convolutional capsule layers, we were not only able to make the capsule network scale with larger input, but also to get significant classification improvement from 87.2% for the original capsule network (on input images of size 28×28) to 88.6% with more than 10 times reduction of the parameters.

TABLE I
THE DEAD/LIVE CLASSIFICATION PERFORMANCE OF THE CLASSIFIERS ON TEST DATA. WE PERFORMED 10-FOLD CROSS VALIDATION AND REPORTED MEAN AND STANDARD DEVIATION OF THE METRICS OVER ALL RUNS.

Model	#params.	Test accuracy	F1-score
AlexNet	4.1M	85.9 (±0.18)	0.855 (±1.7 e-3)
ResNet-50	5.0M	86.5 (±0.10)	0.863 (±1.0 e-3)
DenseNet-BC ($k = 24$)$^{(1)}$	1.8M	88.0 (±0.08)	0.878 (±1.1 e-3)
Capsule Network$^{(2)}$	6.9M	87.2 (±0.11)	0.870 (±1.6 e-3)
Deep Capsule Network	615K	**88.6** (±0.07)	**0.882 (±1.0 e-3)**

$^{(1)}$: k denotes the network's growth rate.
$^{(2)}$: The only network with 28×28 input size. Other models take in 51×51 input images.

We also compared the classification performance of the capsule networks with that of CNNs when fewer numbers of training examples are available. Models are trained 10 times on random subsets of the original training set and the average prediction accuracies (± std.) over the same test set are presented in the right panel of Fig. 6. It is observed that capsule networks are more robust when less training samples are available. Interestingly, while DenseNet performs better than the original CapsNet when trained on the whole data (88.0% compared with 87.2%), its prediction accuracy goes below capsule network when using only 20% of the training samples. The original and deeper proposed capsule networks are performing more robust and achieving almost similar performances when the number of samples is reduced to more than 10%. This emphasizes the intrinsic robustness of the capsule networks rooted in its viewpoint invariant matrix of weights, meaning that even the poses of test sample's parts change by a large degree, capsule networks are capable of getting back the pose of the whole object. This property possibly accounts for the proposed capsule network's ability to generalize well with the smaller number of training samples.

B. Temporal Apoptosis Classification

As mentioned, we use LSTM to capture and encode the temporal information from within the sequential time-lapse apoptotic frames. However, instead of directly feeding the raw image patches to LSTM, we use the models trained in the previous section as feature extractors. For the CNNs, the extracted features are the 512 features of the last fully-connected layer in AlexNet and the global average pooling layer in both ResNet-50 and DenseNet-BC. Note that the structure of these networks is modified to make them compatible with our data. In capsule networks, the two 16-dimensional activation vectors of the *CellCaps* are used as features. More importantly, the feature extractor model parameters are not frozen but are left free to get trained and fine-tuned along with the LSTM parameters. According to our experiments, fine-tuning the pre-trained models results in more informative features and higher overall prediction accuracy compared with initializing the model weights randomly.

Multiple variants of LSTMs were tested, including stacked (multi-layer) and bi-directional LSTMs, and the best performing models with the optimal configurations are reported in Table II. In the table, the frame-based accuracy is simply the percentages of images classified correctly and the sequence-based accuracy is computed on the single label generated for each sequence. To make the sequence label, we define a mapping from the frame labels to the sequence label which assigns one binary label to each movie sequence indicating live (label '0') or dead (label '1'). According to this mapping, a sequence is labeled dead or apoptotic ('1') if it contains either 5 consecutive dead frames or one-fifth of the frames are labeled dead. Otherwise, it is labeled live ('0').

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

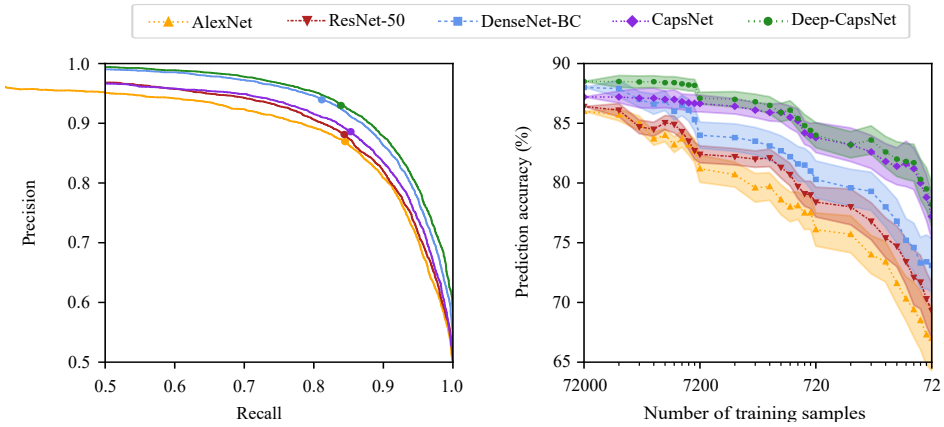

Fig. 6. Classification performance of various networks on the test data. **Left:** precision-recall curves for the networks trained on the whole training set. The points on the curves show the precision and recall values at the default threshold of 0.5. **Right:** prediction accuracy of models trained on different numbers of training samples. Training is conducted 10 times on random subsets of the training set, and the mean prediction accuracy (± std.) is reported.

Experimental results indicate that the best performances are achieved using bi-directional LSTMs. However, it should be noted that bi-direstional LSTMs are not causal and do not fit in the real-time classification systems as they need the whole sequence for making the inference. Capsule networks achieve the highest prediction performance with our proposed deep CapsNet model achieving 93.8% and 93.9% frame and sequence-based accuracies respectively. More importantly, CapsNets achieve such high prediction performances using a simple recurrent model with only one hidden layer and much fewer hidden units. This points out that the very few features extracted from CapsNets (32 features; 16 per Cell Capsule) are rich and informative enough to make an accurate inference. This is partially enforced by using the reconstruction network which reconstructs the whole input image out of these vectors.

Fig. 7 depicts the predictions for thirty sequences sampled randomly from the test data. The top left panel shows that the ground truth labels also contain some level of noise (labels changing from 'dead' to 'live') which is due to the thresholding method used for generating labels out of the death marker intensity curve (see Section IV). While DenseNet and CapsNet (with no recurrent structure stacked) achieve 88.0% and 88.6% prediction accuracies, stacking the recurrent networks improves the performance up to 93.3% and 93.8%. Incorporating a recurrent structure not only improves the prediction accuracy but also significantly reduces the fluctuations in the sequential predictions. It also helps to predict the cell death time more accurately. This is quantified using the MAUDE and MADTE metrics respectively whose values are presented in Table II for all models. While networks with no recurrence make unstable predictions (MAUDE of 9.82 and 7.11 for DenseNet and CapsNet respectively), stacking the recurrent model and end-to-end fine-tuning reduces the fluctuations with the MAUDE value decreasing to 0.80.

C. Visual Explanations of Deep Networks Inference

While the results support the superior prediction performance of capsule network, it is still unclear *How they predict what they predict*. Finding an explanation for the way these models make inferences helps to build trust in intelligent systems and to identify their failure mode. In the cases that AI works significantly better than humans, the goal of model transparency is *machine teaching*- i.e. a machine teaching a human how to make better decisions.

Grad-CAM [38] is a technique for providing such *visual explanation* of the CNN decisions. It uses the gradient of the predicted class to produce a localization heat-map highlighting the image regions that are most important in making a prediction. We adapted Grad-CAM to our trained deep CapsNet by flowing the gradient from the target output capsule (in CellCaps layer) into the 7×7 SecondaryCaps layer to understand the importance of each capsule for a decision of interest. The same method is applied to the last 6×6 layer of the DenseNet for comparison as depicted in Fig. 8.

As depicted in Fig. 8, CapsNet detects the whole cell and makes prediction according to the cell edges and body texture while CNN's region of interest is bounded to confined regions mostly around the cell edges. This is due to the routing mechanism employed in CapsNet which lets lower-level capsules (those capturing relevant information) to agree and consistently route information to higher-level capsules

TABLE II
TEST PREDICTION PERFORMANCE RESULTS FOR THE DEAD/LIVE CLASSIFICATION OF SEQUENTIAL APOPTOSIS DATA. EVALUATION IS CONDUCTED USING 10-FOLD CROSS VALIDATION AND AVERAGE (±STD.) OVER THE RUNS IS REPORTED FOR EACH OF THE METRICS.

L and H are the number of hidden layers and their corresponding number of hidden units of the recurrent structure.

Models	L	H	frame-based accuracy (%)	sequence-based accuracy (%)	F1-score	MAUDE	MADTE
AlexNet + Bi-LSTM	2	[256, 256]	92.4 (±0.06)	92.3 (±0.07)	0.894 (±8.0 e-4)	4.35 (±0.12)	3.12 (±0.21)
ResNet-50 + Bi-LSTM	2	[256, 128]	92.8 (±0.06)	92.6 (±0.06)	0.901 (±7.4 e-4)	2.30 (±0.10)	1.79 (±0.15)
DenseNet	-	-	88.0 (±0.08)	86.3 (±0.08)	0.875 (±1.1 e-3)	9.80 (±0.33)	6.15 (±0.42)
DenseNet + LSTM	2	[512, 256]	93.0 (±0.04)	92.8 (±0.05)	0.913 (±8.4 e-4)	1.82 (±0.05)	2.90 (±0.16)
DenseNet + Bi-LSTM	2	[256, 128]	93.2 (±0.03)	93.0 (±0.04)	0.921 (±6.8 e-4)	1.38 (±0.10)	2.12 (±0.13)
Deep CapsNet	-	-	88.6 (±0.07)	87.5 (±0.08)	0.882 (±1.0 e-3)	7.15 (±0.19)	4.28 (±0.23)
Deep CapsNet + LSTM	1	32	93.6 (±0.03)	93.6 (±0.02)	0.931 (±6.1 e-4)	1.25 (±0.06)	**1.34 (±0.08)**
Deep CapsNet + Bi-LSTM	1	32	**93.9** (±0.03)	**93.9** (±0.04)	**0.934 (±5.8 e-4)**	**0.77 (±0.04)**	1.39 (±0.08)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMI.2019.2918181, IEEE Transactions on Medical Imaging

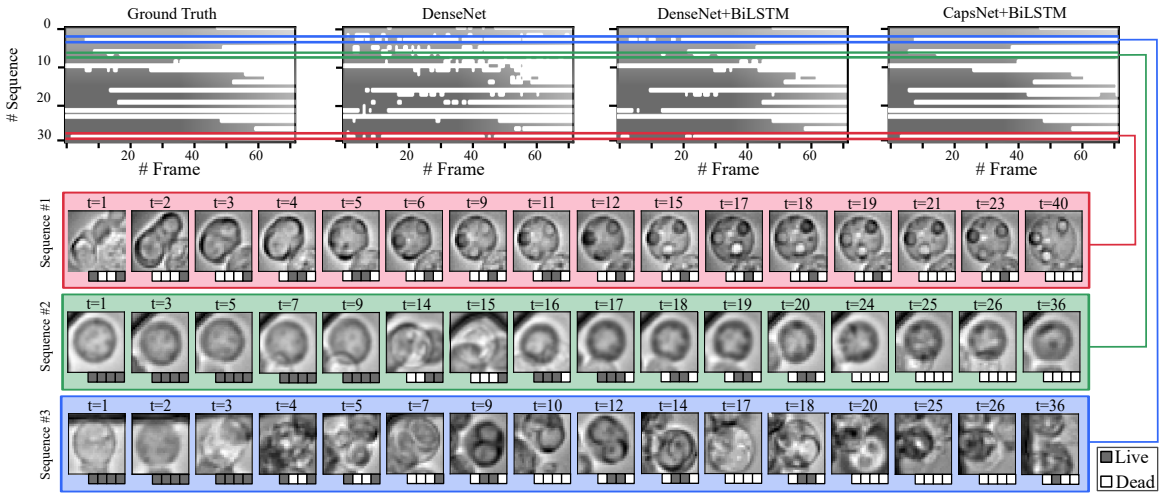

Fig. 7. Illustrating the sequential classification results on samples from test data. **Top:** the ground truth labels and deep networks predictions for 30 sequences (selected randomly) each of length 72 frames. CapsNet+BiLSTM gives the most accurate and stable prediction outputs compared to other structures. **Bottom:** Some of the frames of three example apoptotic sequences (marked with dashed lines on the top panel) and their associated prediction. The four squares at the bottom of each frame shows (from left to right) the ground truth, DenseNet, DenseNet+BiLSTM, and CapsNet+BiLSTM predictions respectively.

which can more effectively process it. It is superior to the max-pooling operation used in the CNNs which only attends to the most active neuron in the pool and ignores the rest. Although the CNN's localized region of interest observed to be sufficient for making inference in simpler cases (a-d), it failed in more complex scenarios. Examples are a cell with irregular shape mimicking the appearance of the other class (e), or existence of other objects such as effector cells or nanowell wall (f-k) which sometimes distracts the model. This happens because CNNs are unable to identify the position of one object relative to another. CapsNet, however, encodes the relative orientation and position of components, thus performs better in crowded scenes. Capsnet's failure cases were observed when the distraction is more severe (l-n). This is inevitable as we only labeled the cancer cells. Due to CapsNet's natural ability to detect multiple objects at once (as experimented in [19]), labeling the T-cells (effectors) along with the cancer cells and detecting both could potentially address this shortcoming. We observe that both CNN and CapsNet fail more often during the live-to-dead transition periods. This is understandable given the high visual uncertainty corresponding to these instances. An example is shown in Fig 7. (seq. #2), where DenseNet+BiLSTM model completely misses the death time (detects it at $t = 24$), while CapsNet+BiLSTM does fairly better and detects it at $t = 16$.

D. Scaling to Large Images

In principle, the proposed CapsNet can work with large images or volumetric data similar to CNNs. Converting from CNNs to CapsNets can be done by replacing the last few convolutional and fully-connected layers of a regular CNN with the proposed convolutional and fully-connected capsule layers, respectively. Dynamic routing mechanism is the computational and memory bottleneck of the original CapsNet [19]. However, the proposed convolutional dynamic routing significantly mitigates this computational issue by allowing only child capsules within a user-defined kernel, instead of all child capsules, to route information to parent capsules. The proposed architecture also reduces the memory footprint of CapsNet by sharing the transformation matrices $\mathbf{W}_{ij}$ among capsules of the same type. As an illustration, for CT scans of $512 \times 512 \times 256$ voxels, the proposed CapsNet has around 16.2 million parameters compared to 22.7 million parameters of a 3D ResNet with 50 layers.

E. Clinical Significance

Consistent with prior studies on tracking of individual cells within series of timelapse images, an accuracy of classification of least $90 - 95\%$ in any single image is essential for the biological user to accept the automated results without the need for manual proof-reading and editing [7], [39]. Unlike Annexin V based staining (accuracy of $>99\%$) for the detection of apoptosis, classification of apoptosis based on only

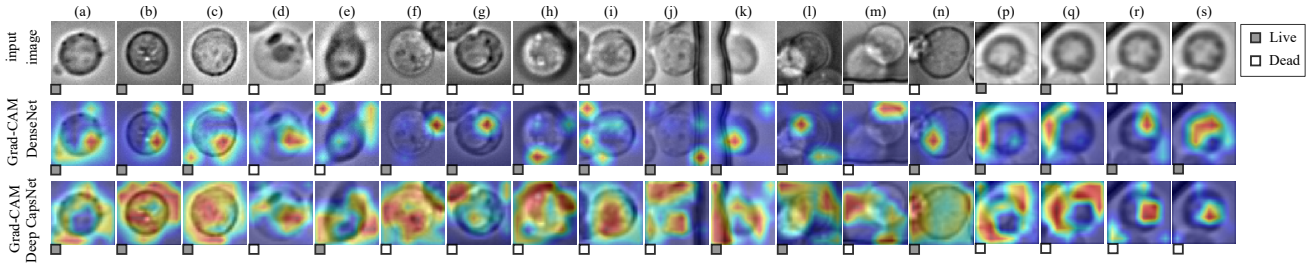

Fig. 8. Visualizing sample images (top) and their corresponding Grad-CAM localization of the region of interest for DenseNet (middle) and CapsNet (bottom).

the phase images is more challenging and the routine expert based classification accuracy is $\sim 95\%$. As we have illustrated with our results in Table II), the accuracy is greatly enhanced, from 88.6% to 93.9%, by taking advantage of the time-lapse data on the same cell. Since the accuracy is close to that of a routine expert, we expect the proposed apoptosis classifier to have high clinical significance. In addition, the ability to generalize well with fewer training examples make CapsNets an appealing choice for the clinical domains.

VI. Conclusion

Our work shows that CapsNet is a promising alternative to CNN. Experimental results demonstrate that CapsNets compare favorably to CNNs when the training size is large, but significantly outperform CNNs on small size datasets. This is especially helpful when the system must learn from small amounts of data, which is often the case in medical settings where data collection is either expensive (due to the requirement of highly-trained experts), or time-consuming (due to the multiple repetitions of an experiment). We showed that by modifying the routing mechanism and stacking a recurrent structure, we can reduce the number of trainable parameters more than 10 times while enhancing the prediction accuracy and stability at the same time. Future work will explore unsupervised methods for training capsule networks, as well as the possibility of employing the proposed capsule network in transfer learning and one-shot learning tasks.

References

[1] S. Elmore, "Apoptosis: a review of programmed cell death," *Toxicologic pathology*, vol. 35, no. 4, pp. 495–516, 2007.

[2] Y. Fuchs and H. Steller, "Programmed cell death in animal development and disease," *Cell*, vol. 147, no. 4, pp. 742–758, 2011.

[3] A.-L. Nieminen, G. J. Gores, J. M. Bond, R. Imberti, B. Herman, and J. J. Lemasters, "A novel cytotoxicity screening assay using a multiwell fluorescence scanner," *Toxicology and applied pharmacology*, vol. 115, no. 2, pp. 147–155, 1992.

[4] J. Kühn, E. Shaffer, J. Mena, B. Breton, J. Parent, B. Rappaz, M. Chambon, Y. Emery, P. Magistretti, C. Depeursinge *et al.*, "Label-free cytotoxicity screening assay by digital holographic microscopy," *Assay and drug development technologies*, vol. 11, no. 2, pp. 101–107, 2013.

[5] L. L. Chan, S. L. Gosangari, K. L. Watkin, and B. T. Cunningham, "Label-free imaging of cancer cells using photonic crystal biosensors and application to cytotoxicity screening of a natural compound library," *Sensors and Actuators B: Chemical*, vol. 132, no. 2, pp. 418–425, 2008.

[6] Z. Wang, M.-C. Kim, M. Marquez, and T. Thorsen, "High-density microfluidic arrays for cell cytotoxicity analysis," *Lab on a Chip*, vol. 7, no. 6, pp. 740–745, 2007.

[7] A. Merouane, N. Rey-Villamizar, Y. Lu, I. Liadi, G. Romain, J. Lu, H. Singh, L. J. Cooper, N. Varadarajan, and B. Roysam, "Automated profiling of individual cell–cell interactions from high-throughput time-lapse imaging microscopy in nanowell grids (timing)," *Bioinformatics*, vol. 31, no. 19, pp. 3189–3197, 2015.

[8] H. Lu, J. Li, M. A. Martinez Paniagua, I. N. Bandey, A. Amritkar, H. Singh, D. Mayerich, N. Varadarajan, B. Roysam, and R. Murphy, "Timing 2.0: High-throughput single-cell profiling of dynamic cell-cell interactions by time-lapse imaging microscopy in nanowell grids." *Bioinformatics*, vol. 1, p. 3, 2018.

[9] C. H. June and M. Sadelain, "Chimeric antigen receptor therapy," *New England Journal of Medicine*, vol. 379, no. 1, pp. 64–73, 2018.

[10] M. N. Androulla and P. C. Lefkothea, "Car t-cell therapy: A new era in cancer immunotherapy," *Current pharmaceutical biotechnology*, vol. 19, no. 1, pp. 5–18, 2018.

[11] D. Pischel, J. H. Buchbinder, K. Sundmacher, I. N. Lavrik, and R. J. Flassig, "A guide to automated apoptosis detection: How to make sense of imaging flow cytometry data," *PloS one*, vol. 13, no. 5, p. e0197208, 2018.

[12] P. Eulenberg, N. Köhler, T. Blasi, A. Filby, A. E. Carpenter, P. Rees, F. J. Theis, and F. A. Wolf, "Reconstructing cell cycle and disease progression using deep learning," *Nature communications*, vol. 8, no. 1, p. 463, 2017.

[13] T. H. Ward, J. Cummings, E. Dean, A. Greystoke, J.-M. Hou, A. Backen, M. Ranson, and C. Dive, "Biomarkers of apoptosis," *British journal of cancer*, vol. 99, no. 6, p. 841, 2008.

[14] I. Vermes, C. Haanen, H. Steffens-Nakken, and C. Reutellingsperger, "A novel assay for apoptosis flow cytometric detection of phosphatidylserine expression on early apoptotic cells using fluorescein labelled annexin v," *Journal of immunological methods*, vol. 184, no. 1, pp. 39–51, 1995.

[15] M. Van Engeland, L. J. Nieland, F. C. Ramaekers, B. Schutte, and C. P. Reutelingsperger, "Annexin v-affinity assay: a review on an apoptosis detection system based on phosphatidylserine exposure," *Cytometry: The Journal of the International Society for Analytical Cytology*, vol. 31, no. 1, pp. 1–9, 1998.

[16] G. Zhang, V. Gurtu, S. R. Kain, G. Yan *et al.*, "Early detection of apoptosis using a fluorescent conjugate of annexin v," *Biotechniques*, vol. 23, no. 3, pp. 525–531, 1997.

[17] D. Wlodkowic, J. Skommer, S. Faley, Z. Darzynkiewicz, and J. M. Cooper, "Dynamic analysis of apoptosis using cyanine syto probes: from classical to microfluidic cytometry," *Experimental cell research*, vol. 315, no. 10, pp. 1706–1714, 2009.

[18] S. Huh, H. Su, T. Kanade *et al.*, "Apoptosis detection for adherent cell populations in time-lapse phase-contrast microscopy images," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2012, pp. 331–339.

[19] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Advances in Neural Information Processing Systems*, 2017, pp. 3856–3866.

[20] Z. Yin, T. Kanade, and M. Chen, "Understanding the phase contrast optics to restore artifact-free microscopy images for segmentation," *Medical image analysis*, vol. 16, no. 5, pp. 1047–1062, 2012.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[23] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks." in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[24] T. Cohen and M. Welling, "Group equivariant convolutional networks," in *International conference on machine learning*, 2016, pp. 2990–2999.

[25] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow, "Harmonic networks: Deep translation and rotation equivariance," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017.

[26] G. E. Hinton, S. Sabour, and N. Frosst, "Matrix capsules with em routing," 2018.

[27] M. T. Bahadori, "Spectral capsule networks," 2018.

[28] C. Xiang, L. Zhang, W. Zou, Y. Tang, and C. Xu, "Ms-capsnet: A novel multi-scale capsule network," *IEEE Signal Processing Letters*, 2018.

[29] A. Jaiswal, W. AbdAlmageed, and P. Natarajan, "Capsulegan: Generative adversarial capsule network," *arXiv preprint arXiv:1802.06167*, 2018.

[30] J. O. Neill, "Siamese capsule networks," *arXiv preprint arXiv:1805.07242*, 2018.

[31] A. Mobiny and H. Van Nguyen, "Fast capsnet for lung cancer screening," *arXiv preprint arXiv:1806.07416*, 2018.

[32] R. LaLonde and U. Bagci, "Capsules for object segmentation," *arXiv preprint arXiv:1804.04241*, 2018.

[33] D. Rawlinson, A. Ahmed, and G. Kowadlo, "Sparse unsupervised capsules generalize better," *arXiv preprint arXiv:1804.06094*, 2018.

[34] S. R. Eddy, "Hidden markov models," *Current opinion in structural biology*, vol. 6, no. 3, pp. 361–365, 1996.

[35] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra *et al.*, "Grad-cam: Visual explanations from deep networks via gradient-based localization." in *ICCV*, 2017, pp. 618–626.

[39] M. Hejna, A. Jorapur, J. S. Song, and R. L. Judson, "High accuracy label-free classification of single-cell kinetic states from holographic cytometry of human melanoma cells," *Scientific reports*, vol. 7, no. 1, p. 11943, 2017.