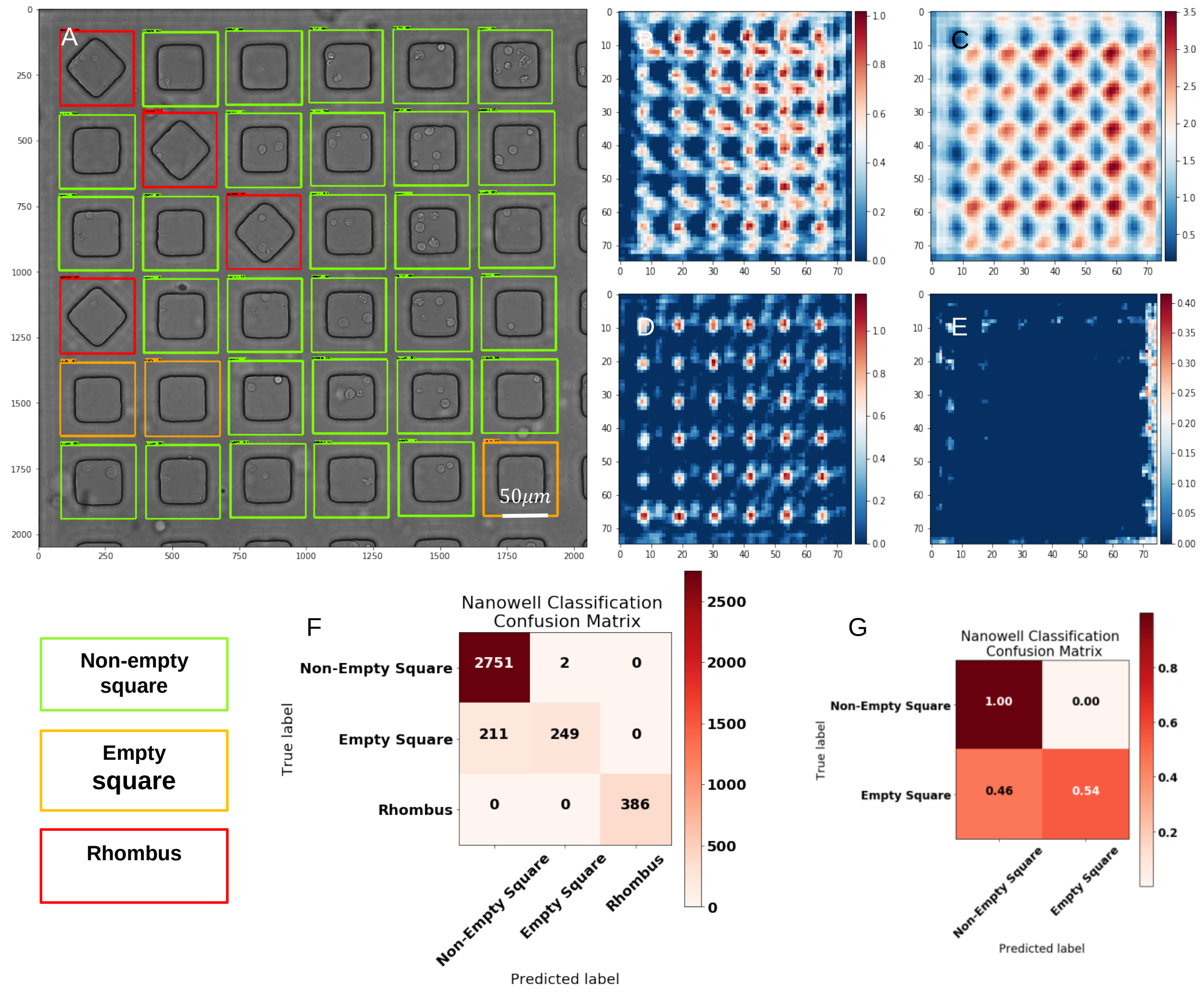
Supplementary Materials

Automated cell detection, tracking and apoptosis classification using deep neural networks from high-throughput time-lapse microscopy in nanowell grids(TIMING)

A1. NANOWELL DETECTION USING CONVOLUTIONAL NEURAL NETWORKS

In addition to cell detection, we also explored the opportunity of nanowell detection using convolutional neural networks. The benefits of using convolutional neural networks for nanowell detection include, but not limited to: (1) fully automatic operation without the need of providing sample nanowell templates as explained in the original method using normal cross correlation; (2) classification of empty nanowells which could avoid around 20% unnecessary computation in cell detection for empty nanowells; (3) robustness against variance in brightness, nanowell deformation, etc; (4) a unified solution using deep neural networks from end to end. In Supplementary Fig 1, nanowell detection



Supplementary Fig 1. Nanowell detection example in “Deep TIMING”. (A) nanowell detection results of a typical block; (B)-(E) show selected feature maps extracted from first stage feature extractor of Faster R-CNN Inception ResNet v2; (F)-(G) are nanowell classification confusion matrix.

results of one typical block are shown in Panel (A); characteristic patterns in selected features maps in panel (B)-(E) indicate how deep convolutional features help nanowell detection; currently, the nanowell classification between empty and non-empty ones still has large room to improve.

B1. Proof of Theorem 1 in Cell Tracking Section

The original cell tracking of K cells over T frames is formulated as a globally optimal edge selection problem on a directed graph. A node n_j^t in the graph represents cell j at frame t, and is described by an attribute vector $d_j^t = \{c_j^t, a_j^t, r_j^t\}$, where $c_j^t = \{x_j^t, y_j^t\}$ is the centroid; a_j^t is the area of the cell; and r_j^t denotes the pixels. An edge $e_{i,j}^t = \{n_i^{t-1}, n_j^t\}$ associates cell i at frame t-1 to cell j at frame t, and we compute an association cost $\varphi_{i,j}^t$ that measures the dissimilarity between cell regions i and j. An edge selection variable $\gamma_{i,j}^t \in \{0,1\}$ indicates if a given edge is selected in the final solution. Using integer programming, we seek the solution $\gamma \in \{0,1\}^N$, where N=(T-1) x K x K that minimizes the following sum of association costs over each nanowell:

$$\boldsymbol{\Gamma = arg_{\gamma \in \{0,1\}^N}\ min \sum_{t=2}^{T} \sum_{j=2}^{K} \sum_{i=1}^{K} \varphi_{i,j}^t \gamma_{i,j}^t} \qquad (1)$$

$$\boldsymbol{s.t.} \begin{cases} \displaystyle\sum_{i=1}^{K} \boldsymbol{\gamma_{i,j}^t \le 1\ for\ j = \{1,\dots,K\},\ t = \{2,\dots,T\}} \\ \displaystyle\sum_{k=1}^{K} \boldsymbol{\gamma_{i,k}^{t+1} \le 1\ for\ j = \{1,\dots,K\},\ t = \{2,\dots,T-1\}} \end{cases} \qquad (2)$$

Note: the original condition defined in (2) is inappropriate since it does not exclude the trivial solution when all $\gamma_{i,j}^t$ equal 0 or some sparse connection. We replace it by $(2')$ which results in a 1 on 1 mapping

$$\boldsymbol{s.t.} \begin{cases} \displaystyle\sum_{i=1}^{K} \boldsymbol{\gamma_{i,j}^t = 1\ for\ j = \{1,\dots,K\},\ t = \{2,\dots,T\}} \\ \displaystyle\sum_{k=1}^{K} \boldsymbol{\gamma_{i,k}^{t+1} = 1\ for\ j = \{1,\dots,K\},\ t = \{2,\dots,T-1\}} \end{cases} \qquad (2')$$

between detected cells in any consecutive frames.

Theorem 1: Under the confinement constraint which is constant cell counts K over T frames and condition defined in (2’), the problem defined in (1) is equivalent to the following minimization problem as defined in (3):

$$\boldsymbol{\Gamma = arg_{\gamma \in \{0,1\}^N} \min \sum_{j=2}^{K} \sum_{i=1}^{K} \varphi_{i,j}^t \gamma_{i,j}^t \ \ \forall\, t} \qquad (3)$$

Proof:

(i) if ${}^{*}\gamma$ is the solution of (3) subject to the conditions, then ${}^{*}\gamma$ is also solution of (1); since ${}^{*}\gamma$ is the solution to (3), then we have the following relationship defined in (4)

$$\sum_{j=2}^{K}\sum_{i=1}^{K}\boldsymbol{\varphi}_{i,j}^{t}\,{}^{*}\boldsymbol{\gamma}_{i,j}^{t} \le \sum_{j=2}^{K}\sum_{i=1}^{K}\boldsymbol{\varphi}_{i,j}^{t}\boldsymbol{\gamma}_{i,j}^{t} \quad \forall\, \mathbf{t} \in \{\mathbf{2},\mathbf{3},\dots,\mathbf{T}\}, \forall \boldsymbol{\gamma}^{\mathbf{t}} \in \{\mathbf{0},\mathbf{1}\}^{\mathbf{K}\times\mathbf{K}} \qquad (4)$$

Summation of both L.H.S and R.H.S of expression (4) w.r.t t from 2 to T will lead to (5)

$$\sum_{t=2}^{T}\sum_{j=2}^{K}\sum_{i=1}^{K}\boldsymbol{\varphi}_{i,j}^{t}\,{}^{*}\boldsymbol{\gamma}_{i,j}^{t} \le \sum_{t=2}^{T}\sum_{j=2}^{K}\sum_{i=1}^{K}\boldsymbol{\varphi}_{i,j}^{t}\boldsymbol{\gamma}_{i,j}^{t} \quad \forall \boldsymbol{\gamma} \in \{\mathbf{0},\mathbf{1}\}^{(\mathbf{N}-\mathbf{1})\times\mathbf{K}\times\mathbf{K}} \qquad (5)$$

So ${}^{*}\gamma$ is the solution of (1);

(ii) if ${}^{**}\gamma$ is the solution of (1) subject to the conditions, then ${}^{**}\gamma$ is also solution of (3); we prove this by contradiction.

Suppose ${}^{**}\gamma$ is solution of (1) while not the solution of (3), and ${}^{***}\gamma$ is the solution of (3), then

$$\exists\, \boldsymbol{t'} \in \{\mathbf{2},\mathbf{3},\dots,\boldsymbol{T}\} \;\; \boldsymbol{s.t.} \; \sum_{j=2}^{K}\sum_{i=1}^{K}\boldsymbol{\varphi}_{i,j}^{t'}\,{}^{***}\boldsymbol{\gamma}_{i,j}^{t'} \le \sum_{j=2}^{K}\sum_{i=1}^{K}\boldsymbol{\varphi}_{i,j}^{t'}\,{}^{**}\boldsymbol{\gamma}_{i,j}^{t'} \qquad (6)$$

And

$${}^{****}\boldsymbol{\gamma}^{t} = \begin{cases} {}^{**}\boldsymbol{\gamma}^{t} & \boldsymbol{if\ t \neq t'} \\ {}^{***}\boldsymbol{\gamma}^{t} & \boldsymbol{if\ t = t'} \end{cases} \qquad (7)$$

will satisfy

$$\sum_{t=2}^{T}\sum_{j=2}^{K}\sum_{i=1}^{K}\boldsymbol{\varphi}_{i,j}^{t}\,{}^{****}\boldsymbol{\gamma}_{i,j}^{t} \le \sum_{t=2}^{T}\sum_{j=2}^{K}\sum_{i=1}^{K}\boldsymbol{\varphi}_{i,j}^{t}\,{}^{**}\boldsymbol{\gamma}_{i,j}^{t} \qquad (8)$$

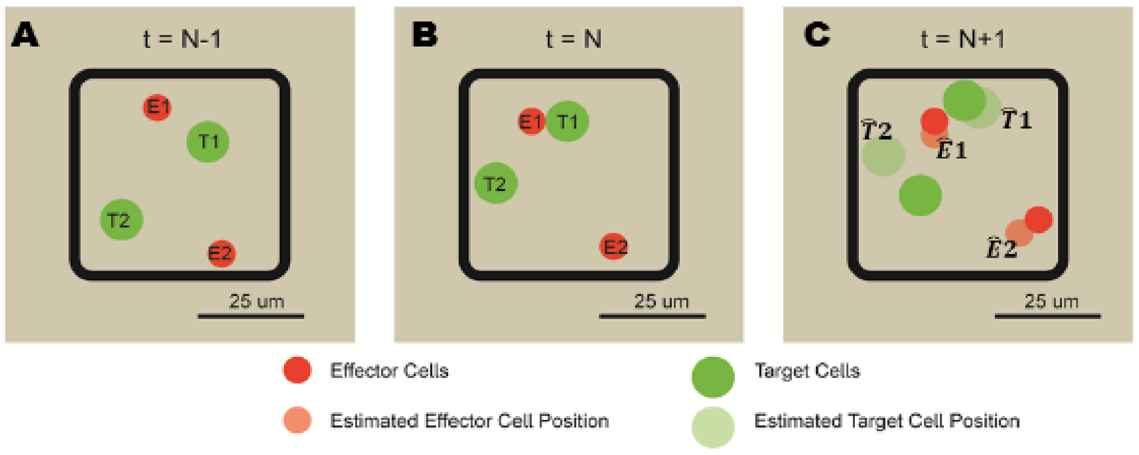
Which contradicts that ${}^{**}\gamma$ is the solution of (1).

Proved.

Cell Tracker Description: The immediate benefit of Theorem 1 would be that we could track the cell frame-by-frame, which is equivalent to the optimization of a global cost function under the confinement-constraint assumption. Considering the instability of area difference and set-theoretic distance defined in [9], a simplified cell tracker, EZ Cell Tracker, only considers the centroid information based on the detection outputs from previous section. The design of EZ cell tracker follows the basic ideas of [10]. As shown in figure 3. The estimated centroid coordinates of cell i in frame t is given by equation (9). The estimated displacement function f could be as simple as the velocity of cell i at time t or other expression considering non-linearity and longer history. Cell tracking from frame t to frame t+1 is a bipartite mapping procedure from the estimated centroid sets $\{\widehat{C_1^t},\widehat{C_2^t},\dots,\widehat{C_K^t}\}$ to the detected centroid sets $\{C_1^{t+1},C_2^{t+1},\dots,C_K^{t+1}\}$, which could be solved by plugging equation (9) and (10) back into equation (3).

$$\widehat{C_i^t} = C_i^t + f\left(C_i^t, C_i^{t-1}, \dots, C_i^{t-M}\right) \tag{9}$$

$$\varphi_{i,j}^t = \left\| \widehat{C_i^t} - C_j^{t+1} \right\|_{L2} \tag{10}$$



Supplementary Fig 2. Demonstration of cell tracker with estimated dynamics.

C1. Metrics of Apoptosis Classification Accuracy

We define two kinds of classification accuracy.

The first one is a frame-based accuracy while the second one is a sequence-based accuracy.

For a movie sequence, each frame is defined in (11):

$$Movie\ Sequence: \quad X[t_i], \quad t_i \in \{1, 2, \dots, N\} \tag{11}$$

The ground truth for each frame is defined in (12):

$$Movie\ Sequence\ Apoptosis\ Label: \quad y[t_i] \in \{0, 1\}, \ t_i \in \{1, 2, \dots, N\} \tag{12}$$

The prediction result for each frame is defined in (13):

$$Movie\ Sequence\ Apoptosis\ Classification: \quad \varphi(X[t_i]) = \hat{y}[t_i] \in \{0, 1\}, \ t_i \in \{1, 2, \dots, N\} \tag{13}$$

We define a comparison function in (14):

$$\delta(p, q) = \begin{cases} 0 & if\ p \neq q \\ 1 & if\ p = q \end{cases} \tag{14}$$

The frame-based classification result is

$$\eta_{frame-based} = \frac{1}{N} \sum_{i=1}^{N} \delta(y[t_i], \hat{y}[t_i]) \tag{15}$$

In the experiment, we randomly select K sequences and average the frame-based classification accuracy, shown in (16).

$$\overline{\boldsymbol{\eta}_{frame-based}} = \frac{\mathbf{1}}{\boldsymbol{KN}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{K}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{N}}\boldsymbol{\delta}(\boldsymbol{y_j}[\boldsymbol{t_i}], \widehat{\boldsymbol{y}}_{\boldsymbol{j}}[\boldsymbol{t_i}]) \qquad (16)$$

The other classification accuracy is sequence-based. In this metric, each movie sequence is assigned a binary label indicating live ('0') or dead ('1'). We define a Mapping from a frame-based label to a sequence-based label:

$$\boldsymbol{\Psi}(\boldsymbol{y}) = \boldsymbol{Y} \in \{\mathbf{0},\mathbf{1}\}, \boldsymbol{y} \in \{\mathbf{0},\mathbf{1}\}^{\mathbf{1}\times\boldsymbol{N}} \qquad (17)$$

$$\boldsymbol{\Psi}(\boldsymbol{y}) = \mathbf{1}\ \boldsymbol{iff.y\ satisfy\ condition}\ (\boldsymbol{i})\ \boldsymbol{or}\ (\boldsymbol{ii})$$

$$(\boldsymbol{i})\ \exists \boldsymbol{t} \in [\mathbf{1},\boldsymbol{N}],\ \boldsymbol{s.t.}\ \ \boldsymbol{y}[\boldsymbol{t}+\boldsymbol{i}] = \mathbf{1}\ \forall \boldsymbol{i} \in \{\mathbf{0},\mathbf{1},\mathbf{2},\mathbf{3},\mathbf{4}\};$$

$$(ii)|\boldsymbol{Apop\_Set}(\boldsymbol{y})| \geq \frac{\boldsymbol{N}}{5}\ , Apop\_Set(y) is\ the\ set\ of\ index\ of\ non-zero\ elements\ in\ y.$$

For a given movie sequence, if $\Psi(y)$ *equals* $\Psi(\hat{y})$, then the classification for this sequence is considered good. We randomly sample K sequences and average the sequence-based classification accuracy, shown in (18).

$$\overline{\boldsymbol{\eta}_{sequence-based}} = \frac{\mathbf{1}}{\boldsymbol{K}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{K}}\boldsymbol{\delta}(\boldsymbol{\Psi}(\boldsymbol{y_j}), \boldsymbol{\Psi}(\ \widehat{\boldsymbol{y}}_{\boldsymbol{j}})) \qquad (18)$$

To quantify the quality of the sequence y, we calculate two additional metrics: sequence standard deviation $\bar{\sigma}(y)$ and counts of ups and downs $\bar{\Lambda}(y)$.

$$\bar{\boldsymbol{\sigma}}(\mathbf{y}) = \frac{\mathbf{1}}{\mathbf{K}}\sum_{\mathbf{i=1}}^{\mathbf{K}}\sqrt{\frac{\mathbf{1}}{\mathbf{N}}\sum_{\mathbf{j=1}}^{\mathbf{N}}(\mathbf{y_i}[\mathbf{t_j}] - \bar{\mathbf{y}})^{\mathbf{2}}} \qquad (\mathbf{19})$$

$$\bar{\boldsymbol{\Lambda}}(\mathbf{y}) = \frac{\mathbf{1}}{\mathbf{K}}\sum_{\mathbf{i=1}}^{\mathbf{K}}\mathbf{Sum}(\mathbf{Diff}(\mathbf{Zero\_Pad}(\mathbf{y}))) \qquad (\mathbf{20})$$

where Zero_Pad(y) add y[0] = 0 to the sequence y and Diff(y) = y[t] – y[t-1] for t in {1,2,3,…,N}.

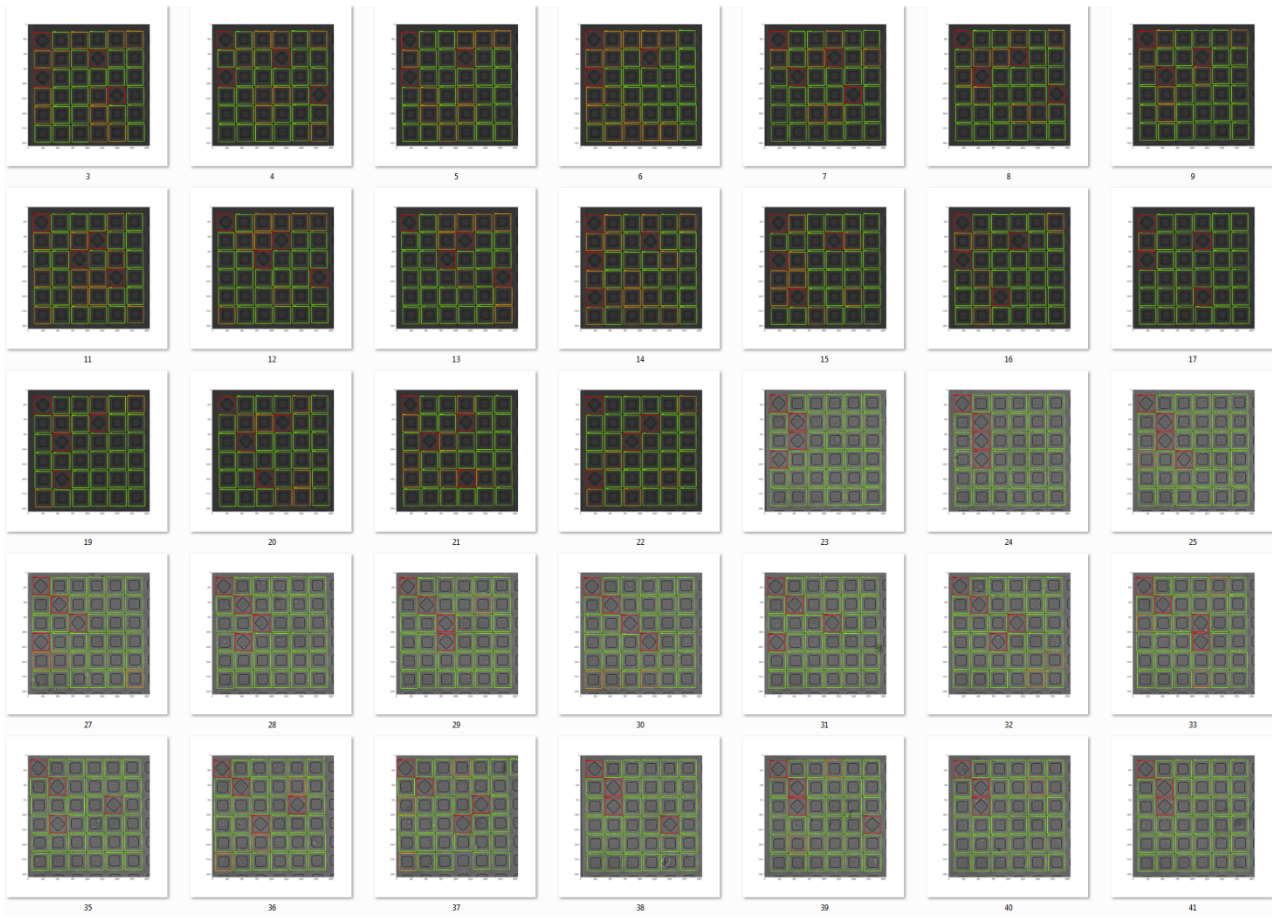
D1. Sample Datasets

GT-100: Sample datasets for nanowell detection testing (Supplementary Fig 3). Includes:

[1] 100 nanowell block images (phase contrast images) randomly sampled from 4 datasets;

[2] Nanowell bounding box ground-truth annotation;

[3] Download link:



Supplementary Fig 3. Selected nanowell block images in GT-100 test set.

GT-500: Sample datasets for cell detection testing (Supplementary Fig. 4). Includes:

[1] 500 nanowell images randomly sampled from 4 TIMING datasets;

[2] Phase contrast (CH0) as well as fluorescent (CH1, CH2) channel images;

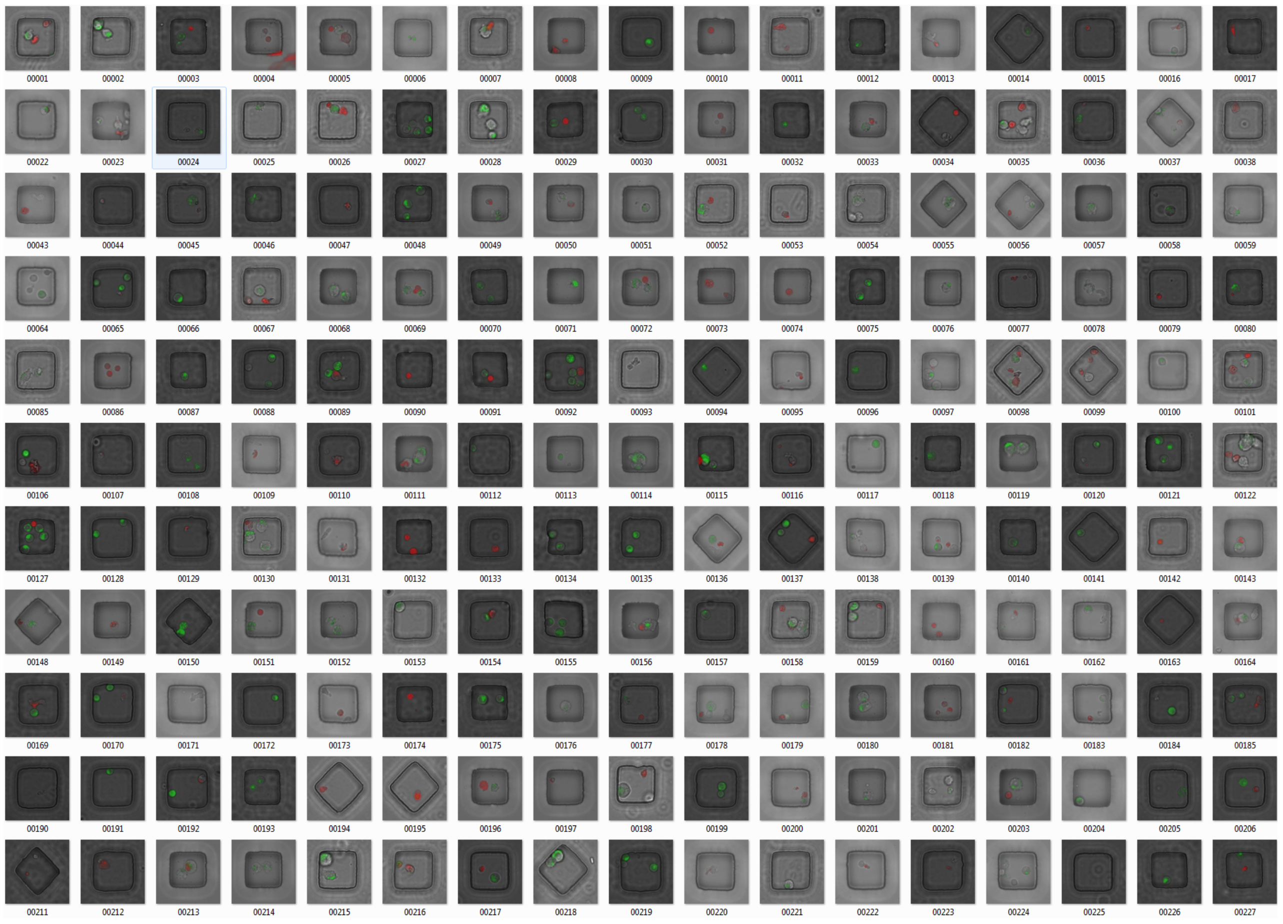
[3] Cell detection ground-truth bounding boxes (containing position, width, height and type information);

GTS-180: Sample datasets for cell tracking testing. It includes 180 annotated time-lapse nanowell movie sequences. Each Sequence contains 97 time-lapse frames. The annotation includes cell bounding boxes as well as cell track IDs, shown at the top-left corner of each bounding box. The ground-truth annotations and images can be loaded into our visualization tool labelTrack (Supplementary Fig 5).

Apoptosis training and test sets: training and test sets for CNN and CNN-LSTM apoptosis classification. All the data are saved as numpy arrays, including:

[1] X_train_Alex_72000.npy: (72000,51,51,1) numpy array containing 72000 cell crops of 51 by 51 pixels;

[2] y_train: (72000,) numpy array containing live/dead labels for corresponding cell crops in x_train_Alex_72000.npy;



Supplementary Fig 4. Selected nanowell images in GT-500 test set.

[3] X_test_Alex.npy: [20000, 51,51,1] numpy array of cell crops for testing;

[4] y_test.npy:[20000,] numpy array containing live/dead labels for corresponding cell crops in x_test_Alex.npy

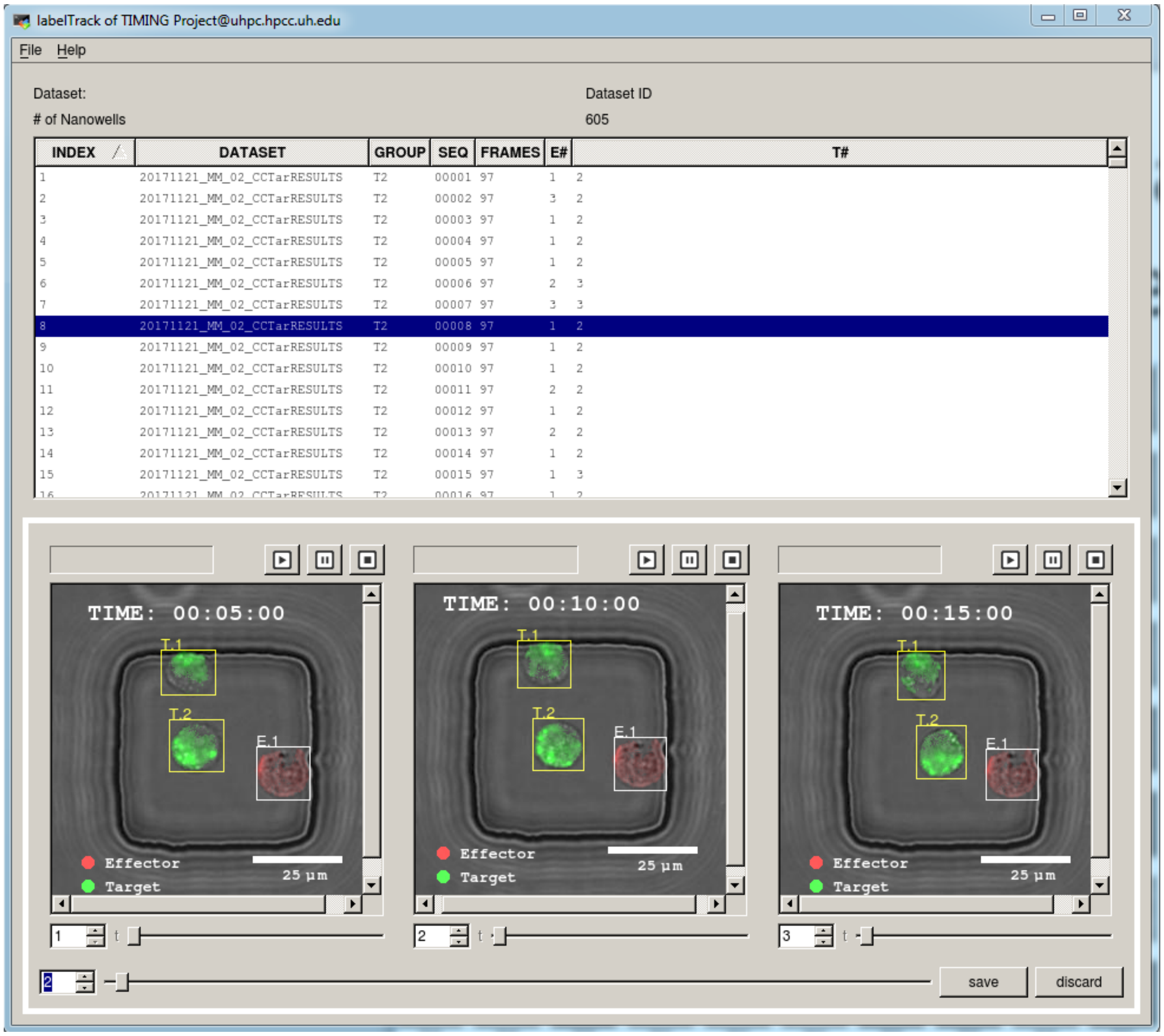
[5] Tx28_Combine_train.npy: [, 72,28,28,1] numpy array containing x sequences of time-lapse cell crop images. Each crop is of size 28 by 28 pixels. Each sequence contains 72 time-lapse crops;

[6] Ty28_Combine_train.npy: [,72,1] numpy array containing live/dead labels for corresponding sequences in Tx28_Combine_train.npy. Labels for each sequence is a 72 element 1-D array.

[7] Tx28_Combine_test.npy: [, 72,28,28,1] numpy array containing x sequences of time-lapse cell crop images. Each crop is of size 28 by 28 pixels. Each sequence contains 72 time-lapse crops;

[8] Ty28_Combine_test.npy: [,72,1] numpy array containing live/dead labels for corresponding sequences in Tx28_Combine_test.npy. Labels for each sequence is a 72 element 1-D array.

Training and testing data could be loaded into the memory through numpy.load() function.



Dataset:

Dataset ID

# of Nanowells

605

INDEX	DATASET	GROUP	SEQ	FRAMES	E#	T#
1	20171121_MM_02_CCTarRESULTS	T2	00001	97	1	2
2	20171121_MM_02_CCTarRESULTS	T2	00002	97	3	2
3	20171121_MM_02_CCTarRESULTS	T2	00003	97	1	2
4	20171121_MM_02_CCTarRESULTS	T2	00004	97	1	2
5	20171121_MM_02_CCTarRESULTS	T2	00005	97	1	2
6	20171121_MM_02_CCTarRESULTS	T2	00006	97	2	3
7	20171121_MM_02_CCTarRESULTS	T2	00007	97	3	3
8	20171121_MM_02_CCTarRESULTS	T2	00008	97	1	2
9	20171121_MM_02_CCTarRESULTS	T2	00009	97	1	2
10	20171121_MM_02_CCTarRESULTS	T2	00010	97	1	2
11	20171121_MM_02_CCTarRESULTS	T2	00011	97	2	2
12	20171121_MM_02_CCTarRESULTS	T2	00012	97	1	2
13	20171121_MM_02_CCTarRESULTS	T2	00013	97	2	2
14	20171121_MM_02_CCTarRESULTS	T2	00014	97	1	2
15	20171121_MM_02_CCTarRESULTS	T2	00015	97	1	3
16	20171121_MM_02_CCTarRESULTS	T2	00016	97	1	2

Supplementary Fig 5. labelTrack visualization tool. Annotated nanowells are shown in the table. Clicked nanowell will be highlighted with images and annotations shown in the image panels below. The image panel could show three frames of the selected nanowell at the same time. Cell track IDs are shown at the top-left corner of each bounding box.

E1. SAMPLE CODES

Recommended System Requirements:

[1] Multi-thread CPU (e.g. INTEL Core i7)

[2] NVIDIA CUDA-enabled GPU: (e.g. Tesla K80 in our experiment);

[3] Hard-drive: 1TB solid state hard drive;

[4] Operating system: Linux (Redhat or Ubuntu);

[5] Anaconda Python environment;

[6] Tensorflow v1.2 and object detection APIs;

[7] Keras;

[8]PyQt 4.11 for visualization;

Model Training:

[1] Create training data tfrecord files:

Follow example in /data/1_Nanowell_Detection/tfRecord_converter.py to create new training data for cell detectors;

[2] Train Faster R-CNN Model:

Follow example in official tensorflow object detection APIs: https://github.com/tensorflow/models/tree/master/research/object_detection; sample training data and pipeline configuration files are in folder /data/2_Cell_Detection/train/;

[3] Train apoptosis CNN classifier:

Use CNN training scripts to train LeNet, AlexNet and ResNet50 apoptosis classifiers respectively; make sure the training data path are correctly specified.

/code/4_Apoptosis_Classification/CNN/Apoptosis_Classifier_LeNet.py

/code/4_Apoptosis_Classification/CNN/Apoptosis_Classifier_AlexNet.py

/code/4_Apoptosis_Classification/CNN/Apoptosis_Classifier_ResNet50.py

[4] Train apoptosis CNN-LSTM classifier:

Use CNN-LSTM training scripts to train apoptosis classifiers using LeNet CNN feature extractor and LSTM as well as its variants:

/code/4_Apoptosis_Classification/CNN-LSTM/LeNet-LSTM-T.py → LSTM

/code/4_Apoptosis_Classification/CNN-LSTM/LeNet-LSTM-Stack-T.py → Stacked LSTM

/code/4_Apoptosis_Classification/CNN-LSTM/LeNet-bi-LSTM-T.py → bi-directional LSTM

/code/4_Apoptosis_Classification/CNN-LSTM/LeNet-LSTM-Sbi-T.py→stacked bi-directional LSTM

Model Testing:

[1] Deep-TIMING-Nanowell-Det:

/code/1_Nanowell_Detection/ Deep-TIMING-Nanowell-Detection-demo.ipynb, run demo using jupyter notebook;

[2] Deep-TIMING-Cell-Det:

Evaluate trained cell detectors over GT-500 test set using /code/2_Cell_Detection/ GT-500-Cell-Detection-demo-eval.ipynb; pre-trained model weights are in folder: folder /data/2_Cell_Detection/weights/

[3] Deep-TIMING-Cell-Track:

Follow the demo IPython notebook: /code/3_Cell_Tracking/GTS-180-Tracker-EVAL.ipynb;

[4] Deep-TIMING-Apoptosis-Classification:

Follow the demo IPython notebook: /code/4_Apoptosis_Classification/Deep-TIMING-Apoptosis-Classification-EVAL.ipynb;

[5] Deep-TIMING-viewer:

Visualization tool labelTrack is included in folder /data/3_Cell_Tracking/GTS-200-viewer/. Make sure required packages are installed successfully (Follow the same steps as setting up TIMING2-board in TIMING 2: https://github.com/troylhy1991/TIMING2). Start the tool by typing "python labelTrack.py", load the dataset by selecting the target folder (e.g. /data/3_Cell_Tracking/GTS-200/), see Supplementary Fig 5.