Title

Protocol for vision transformer-based evaluation of drug potency using images processed by an optimized Sobel operator

Permalink
https://escholarship.org/uc/item/2rb190rq

Journal
STAR Protocols, 4(2)

ISSN
2666-1667

Authors
Wang, Yongheng
Zhang, Weidi
Wu, Yi
et al.

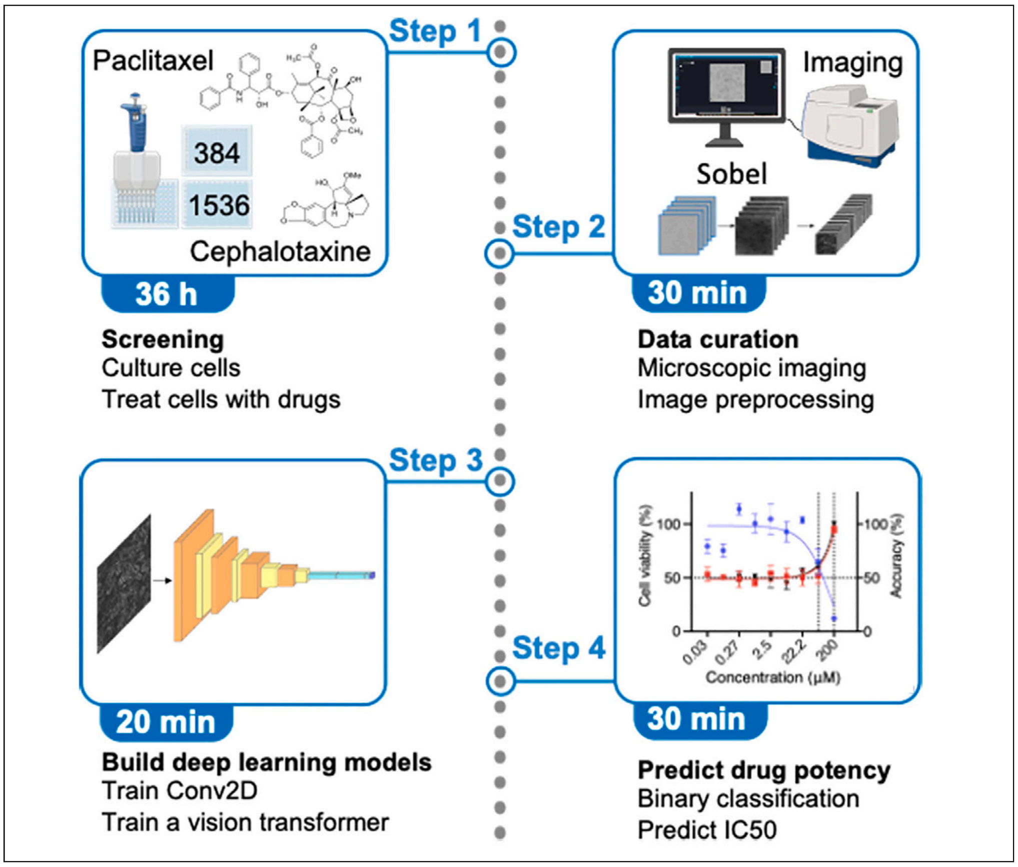
Publication Date
2023-06-01

DOI
10.1016/j.xpro.2023.102259

Peer reviewed

Protocol

Protocol for vision transformer-based evaluation of drug potency using images processed by an optimized Sobel operator



Yongheng Wang, Weidi Zhang, Yi Wu, ..., Jiawei Zhang, Kit S. Lam, Aijun Wang

yhgwang@ucdavis.edu (Y.W.)
jiwzhang@ucdavis.edu (J.Z.)
kslam@ucdavis.edu (K.S.L.)
aawang@ucdavis.edu (A.W.)

Highlights

Protocol for label-free evaluation of drug potency using a vision transformer and a Conv2D

A cost-effective method for high-throughput screening of anticancer drugs

An adaptable method for screening molecules that affect cell density and shape

Conventional approaches for screening anticancer drugs rely on chemical reactions, which are time consuming, labor intensive, and costly. Here, we present a protocol for label-free and high-throughput assessment of drug efficacy using a vision transformer and a Conv2D. We describe the steps for cell culture, drug treatment, data collection, and preprocessing. We then detail the building of deep learning models and their use to predict drug potency. This protocol can be adapted for screening chemicals that affect the density or morphological features of cells.

Wang et al., STAR Protocols 4, 102259
June 16, 2023 © 2023 The Authors.
https://doi.org/10.1016/j.xpro.2023.102259

Protocol

Protocol for vision transformer-based evaluation of drug potency using images processed by an optimized Sobel operator

Yongheng Wang,[1,9,10,*] Weidi Zhang,[2,9] Yi Wu,[2] Chuyuan Qu,[3] Hongru Hu,[4] Teresa Lee,[5] Siyu Lin,[2] Jiawei Zhang,[6,*] Kit S. Lam,[7,8,*] and Aijun Wang[1,2,11,*]

[1]Department of Biomedical Engineering, University of California, Davis, Davis, CA 95616, USA

[2]Center for Surgical Bioengineering, Department of Surgery, University of California, Davis, School of Medicine, Sacramento, CA 95817, USA

[3]Microsoft, Redmond, WA 98052, USA

[4]Integrative Genetics and Genomics, University of California, Davis, Davis, CA 95616, USA

[5]Department of Biomedical Engineering, Yale University, New Haven, CT 06511, USA

[6]Department of Computer Science, IFM Lab, University of California, Davis, Davis, CA 95616, USA

[7]Department of Biochemistry and Molecular Medicine, UC Davis NCI-designated Comprehensive Cancer Center, University of California, Davis, Sacramento, CA 95817, USA

[8]Division of Hematology and Oncology, Department of Internal Medicine, School of Medicine, University of California, Davis, Sacramento, CA 95817, USA

[9]These authors contributed equally

[10]Technical contact

[11]Lead contact

*Correspondence: yhgwang@ucdavis.edu (Y.W.), jiwzhang@ucdavis.edu (J.Z.), kslam@ucdavis.edu (K.S.L.), aawang@ucdavis.edu (A.W.)
https://doi.org/10.1016/j.xpro.2023.102259

SUMMARY

Conventional approaches for screening anticancer drugs rely on chemical reactions, which are time consuming, labor intensive, and costly. Here, we present a protocol for label-free and high-throughput assessment of drug efficacy using a vision transformer and a Conv2D. We describe the steps for cell culture, drug treatment, data collection, and preprocessing. We then detail the building of deep learning models and their use to predict drug potency. This protocol can be adapted for screening chemicals that affect the density or morphological features of cells.
For complete details on the use and execution of this protocol, please refer to Wang et al.[1]

BEFORE YOU BEGIN

Anticancer molecules typically inhibit the growth of tumor cells, resulting in a reduction of cell density, and sometimes cause changes in cellular shape. Both features (i.e., density and shape) can be identified by computer vision algorithms.[2–4] We develop approaches using a vision transformer and a Conv2D to evaluate drug potency in a fast and cost-effective manner, validate the methods using 4 different drugs and different multi-well plates, and describe the details of the equipment and reagents in key resources table. This protocol outlines the steps to culture cells and to treat them with different drugs at various concentrations, high-throughput image cells using Pico, preprocess the images using different platforms, calculate IC50s (i.e., the concentration at which anti-cancer drugs kill half of the cells[5]) using ImageJ and GraphPad, and compare the accuracies of different methods.

> *Note:* (1) If the B16-F10 melanoma cell line and anticancer drugs are not available, other cancer cell lines and oncology medicines can be used. (2) We collect images using ImageXpress

Pico in this study. However, this instrument can be replaced with other high-throughput imaging systems (e.g., BioTek Cytation 5 Cell Imaging Multimode Reader). (3) Although we quantify the number of cells using ImageJ, alternative software can be employed to count the nuclei (e.g., CellProfiler: https://cellprofiler.org/examples). (4) In this study, we stain cells with Hoechst to count the number of nuclei. To evaluate the cell viability, we could use other assays (e.g., MTT assays, CCK-8 assays, ATP assays, etc.) (5) We train and test the models locally using a Jupyter Notebook. Google Colab and Google Drive are good alternatives to train the models.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Critical commercial assays		
Tissue culture plate 96 wells	Fisherbrand	Cat# FB012931
Assay plate, 384 well	Corning	Cat# 29706027
Assay plate, 1536 well	Corning	Cat# 23615005
Cell culture flask	Corning	Cat# 430825
Centrifuge tube	VWR	Cat# 525-0606
Serological Pipette	VWR	Cat# 490019-704
Sterile aerosol pipet tips	VWR	Cat# 76322-150
Reagent reservoir	CELLTREAT	Cat# 2202252
Self-standing cryovial	Olympus	Cat# 24-202P
Chemicals, Peptides, and Recombinant Proteins		
Paclitaxel	LC Laboratories	Cat# P-9600-250mg
Cephalotaxine	Toronto Research Chemicals	Cat# C261050
Fasudil	TAZCHEM (Pty) Ltd	Cat# R1036
Irinotecan	APExBIO	Cat# B2293
Fetal bovine serum	Atlanta Biologicals	Cat# S11150
Dimethyl sulfoxide	Sigma-Aldrich	Cat# 472301
TrypLE™ Select Enzyme (1×)	Gibco	Cat# 12563029
Penicillin-streptomycin (10,000 U/mL)	Gibco	Cat# 15140122
DMEM, high glucose, pyruvate	Gibco	Cat# 11995065
DPBS, no calcium, no magnesium	Gibco	Cat# 14190144
Hoechst 33342	Thermo Fisher Scientific	Cat# H3570
Reagent	Final concentration	Amount
Fetal bovine serum	10% in DMEM	N/A
Penicillin-streptomycin	1% in DMEM	N/A
DMEM	N/A	N/A
Experimental models: organisms/strains		
B16-F10	Wang et al.[1]	N/A
Deposited data		
Cellular images	Zenodo	https://doi.org/10.5281/zenodo.7509014
Software and algorithms		
ImageJ bundled with Java8	National Institution of Health	imagej.nih.gov/ij/index.html
PhotoScape X	N/A	http://x.photoscape.org/
Python 3.7.13	Van Rossum and Darke[6]	python.org
Tensorflow 2.9.1	Abadi et al.[7]	tensorflow.org
Jupyter Notebook 6.4.8	Jupyter	jupyter.org
Numpy 1.22.4	Oliphant[8]	Numpy.org
Matplotlib 3.5.1	Hunter[9]	Matplotlib.org
Anaconda 3-2022.10	Anaconda	Anaconda.com
Tensorflow-estimator	TensorFlow	tensorflow.org

(Continued on next page)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Tensorflow_addons	TensorFlow	tensorflow.org
Osobel_edge_code.ipynb	Zenodo	https://doi.org/10.5281/zenodo.7509014
High_pass_code.ipynb	Zenodo	https://doi.org/10.5281/zenodo.7509014
Conv2D_code.ipynb	Zenodo	https://doi.org/10.5281/zenodo.7509014
Vision_Transformer_code.ipynb	Zenodo	https://doi.org/10.5281/zenodo.7509014
Other		
TSX Series Ultra-Low Freezers	Thermo Fisher	TSX70086A
ImageXpress® Pico	Molecular Devices	ImageXpress Pico system
Cryo 1°C Freezing Container	NALGENE™	Cat# 5100-0001
Pipet-aid® XL	Drummond Scientific Co.	Cat# 4-000-105
Research® plus pipette	Eppendorf®	Cat# EP2231300010-6EA
Sanyo MCO-19AIC(UV) CO_2 Incubator	Sanyo	Cat# SA-MCO19
SterilGARD Hood II Type A/B3	The Baker Company, Inc.	Cat# SG600
Bright-Line™ Counting Chamber	Hausser Scientific™	Cat# 02-671-51B
Inverted Phase Contrast Digital Microscope	Leica	DMi1
Refrigerated Centrifuge	Thermo Sorvall	Cat# RT-7 plus
12-Channel Multichannel Pipettes	Eppendorf®	Cat# 22453980

STEP-BY-STEP METHOD DETAILS

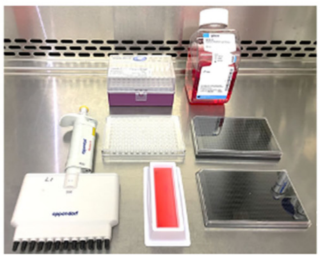
Cell culture

Timing: 24 h

This describes the detailed protocol for preparing the cells before the drug treatment.

1. Take a vial of cancer cells from the liquid nitrogen tank, thaw it in a water bath at 37°C, and centrifuge the vial at 800 *g* for 4 min.
2. Discard the supernatant and resuspend the cell pellet in 10 mL of DMEM complete culture medium.
3. Culture the cells in a T75 flask in an incubator at 37°C and 5% CO2 for 24 h.
4. Collect the cells after removal of culture medium, rinse the flask with 3 mL PBS, digest the cells with the TrypLe Select enzyme, followed by centrifugation at 800 *g* for 4 min.
5. Resuspend and count the cells.
6. Choose the plate (e.g., 96-well plate, 384-well plate, 1536-well plate, Figure 1A) based on the number of drugs that need to be tested.
7. Seed the cells at a concentration of 6, 000 cells per well for 96-well plates.

A



B

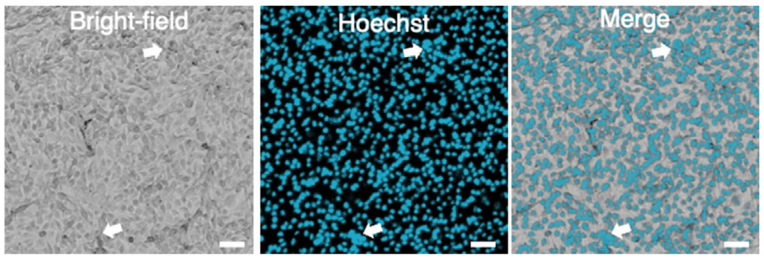


Figure 1. Cell culture and seeding cells in microplates
(A) Settings for seeding cells.
(B) Images of cells from an overpopulated well. Scale bars are 30 μm.

Note: If the concentration of the cells is too high at the beginning, the control groups will have too many cells 48 h later, presenting challenges in accurately counting cells stained with Hoechst, as demonstrated in Figure 1B.

Treating cells with different drugs at various concentrations

Timing: 25 h

The cells are incubated with different drugs. Detailed steps are described below.

8. Prepare stock solutions by dissolving drugs in DMSO.
9. Dilute the stock solutions with DMEM complete culture medium and prepare the first concentration for the drug (e.g., 200 μM Cephalotaxine and Fasudil).
10. Prepare a serial dilution by mixing 1 volume of the previous solution with 2 volumes of the DMEM complete culture medium, i.e., 1:3 dilution.
11. Remove the old media and replace them with 200 μL solution containing drugs and fresh culture media.
12. Keep the microplates in the incubator at 37°C and 5% CO_2 for 24 h.

Note: If bubbles were created during pipetting, they will show up in the final images, creating new shapes and introducing artifacts. To prevent bubbles, new pipette tips should be used for every well. Do not push the plunger of the pipette over the first stop. Otherwise, air will be injected into the solution, forming bubbles.

Imaging the cells treated with drugs using Pico

Timing: 25 h

Microscopic images are collected using the ImageXpress Pico imaging system. The procedures are described below in detail.

13. To image multi-well plates, turn on the ImageXpress Pico imaging system installed in Windows 10 (Note) and double-click click CRX App (CRX-2.6.130 for Windows 10) icon to open the application.
14. Click "ACQUISITION" on the home page (Figure 2A), and click "ADD PROTOCOL" (Figure 2B).
15. Click the second icon below "STEPS" , and select "96 Corning 3603" (Figure 2C).
16. Click the second icon under "TOOLS" , turn off all the "STAINS" and set "TL" to "Capture first" (Figure 2D).
17. Click the third icon under "TOOLS" , and select 4× objective (3.464 mm × 3.464 mm, PL FLUOTAR 4 × 0.13 objective, Figure 2E).
18. Click the third icon under "STEPS" , and select 29.8% (Figure 2F).
19. Click the sixth icon under "STEPS" , and select the wells of interest (Figure 2G).
20. Click the ninth icon under "STEPS" , and fill in the protocol name and save it (Figure 2H).
21. Click the tenth icon under "STEPS" , and fill in the experiment name and start imaging (Figure 2I).
22. Return to the home page and click "MONITOR", if the imaging is progressing, the experiment will show under "IN PROGRESS"; if the imaging is completed, it will show under "SUCCEEDED". If the experiment is completed, click on it, and then click "Thumbnail View" (Figure 2J).
23. Select the wells, click the fifth icon on the right , and select "Export full size" and export the images (Figure 2K).

Note: The ImageXpress Pico is an all-in-one imaging system and a complete solution containing all the hardware (camera, stage, monitor, etc.) and software (CRX App) needed for our

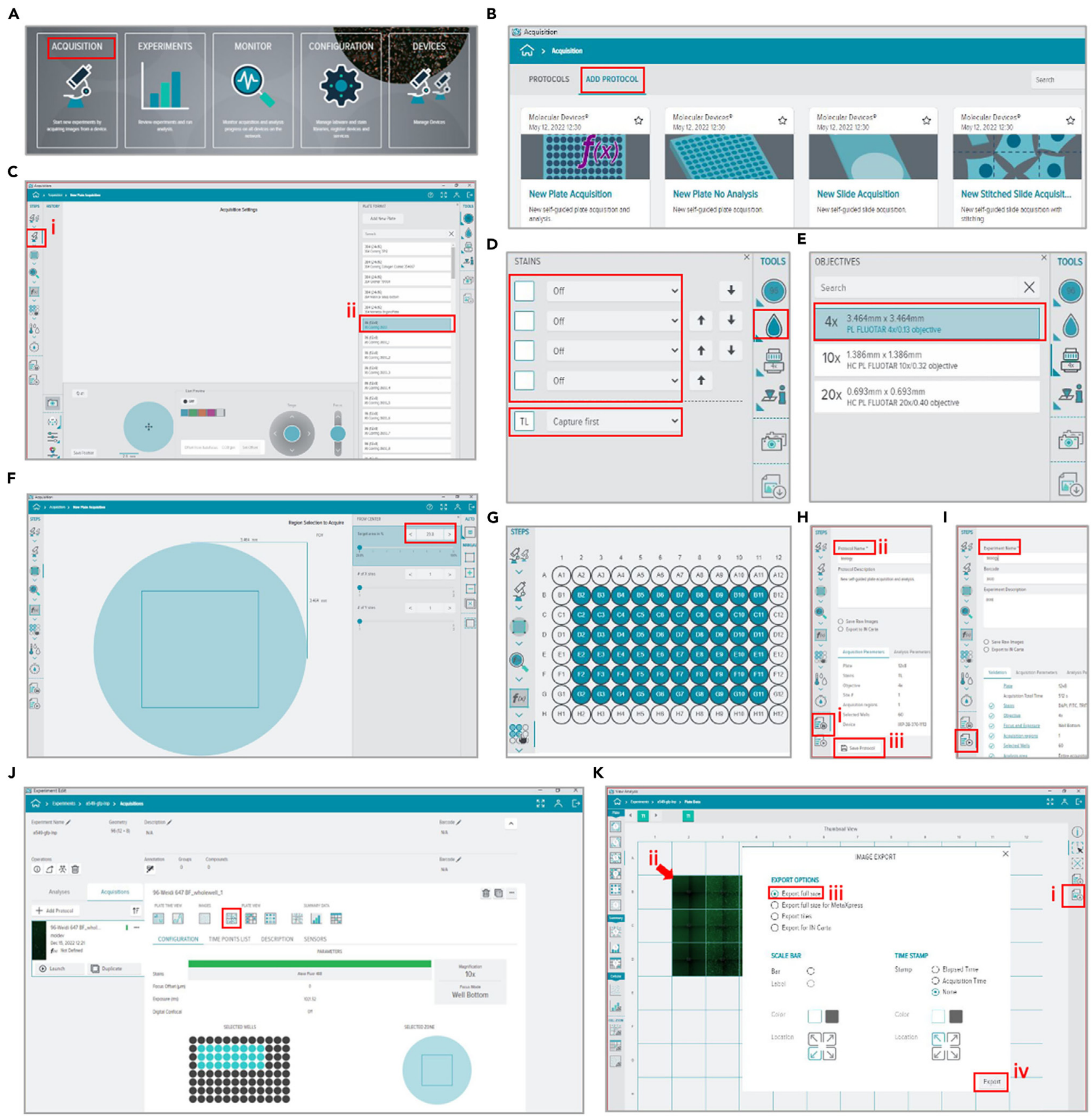
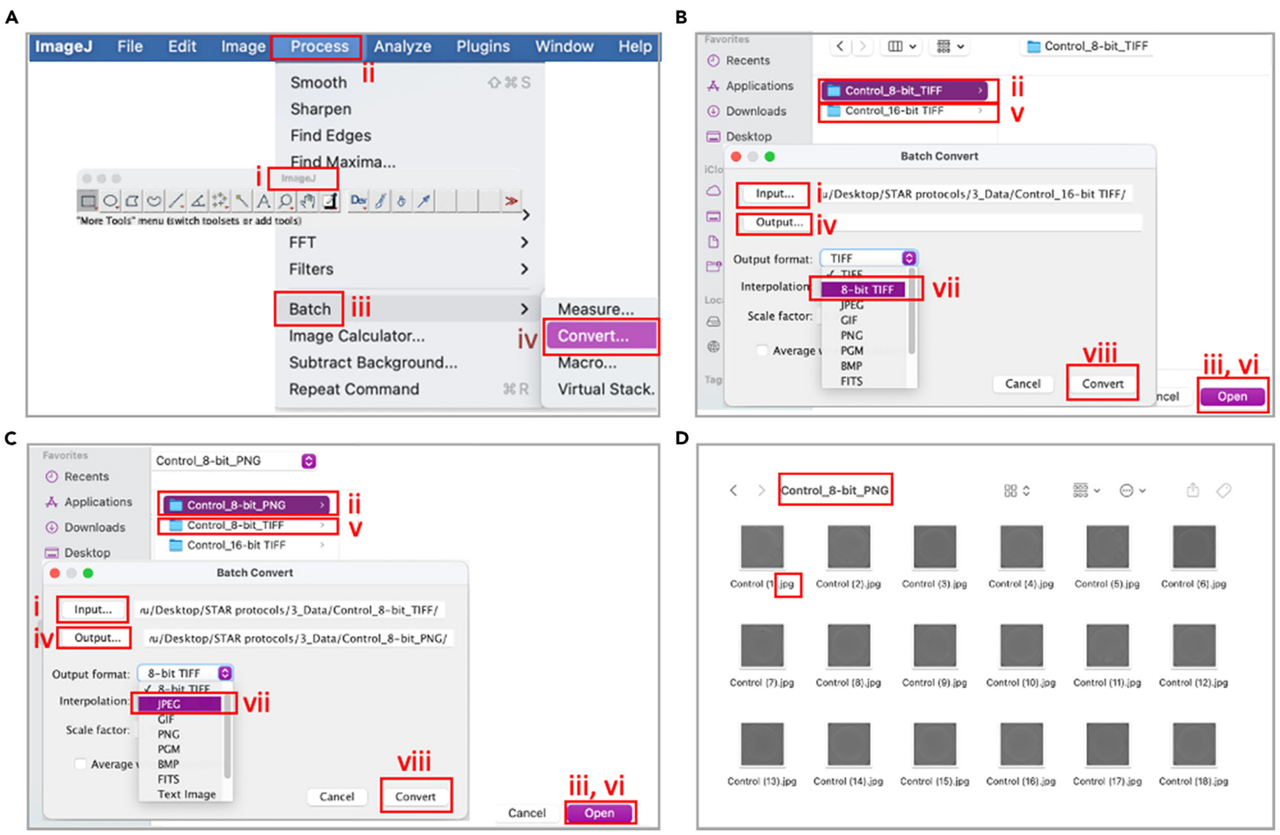


Figure 2. Pico interface with a full pipeline
(A) Pico CRX home screen.
(B–I) Creation of a new protocol (B–H) and start imaging (I).
(J) Analyze the pictures.
(K) Select images and export.

experiments. After purchasing this imaging system from Molecular Devices, the company helped set up the CRX App and provided on-site training.

https://www.moleculardevices.com/products/cellular-imaging-systems/high-content-imaging/imagexpress-pico.



B: Input... u/Desktop/STAR protocols/3_Data/Control_16-bit TIFF/ — Output format: TIFF

C: Input... ru/Desktop/STAR protocols/3_Data/Control_8-bit_TIFF/ — Output... ru/Desktop/STAR protocols/3_Data/Control_8-bit_PNG/ — Output format: 8-bit TIFF

Figure 3. Batch-convert files with ImageJ
(A–C) Procedures to change the format of images.
(D) Output folder contains PNG files.

For other general questions related to Pico, please refer to the "ImageXpress Pico Automated Cell Imaging System User Guide" (https://www.moleculardevices.com/sites/default/files/en/assets/user-guide/dd/img/imagexpress-pico-automated-cell-imaging-system-with-cellreporterxpress-software-v2-5.pdf).

Modify the file formats using ImageJ

Timing: 10 min

This describes the detailed procedures for changing the format of the images.

24. Create two new folders and name them "8-bit TIFF" and "8-bit PNG", respectively.
25. Open ImageJ (bundled with 64-bit Java 8 for Windows 10 can be downloaded from https://imagej.nih.gov/ij/download.html) and select "Process > Batch > Convert" (Figure 3A).
26. Select the input and output folders, change the output format to "8-bit TIFF," and click "convert" (Figure 3B).
27. Select the input and output folders, change the output format to "8-bit PNG," and click "Convert" (Figure 3C).
28. Check the format of the files in the final output folder (Figure 3D).

Process the images with Jupyter Notebook (6.4.8 for Windows 10)

Timing: 30 min

The images are prepared using Jupyter Notebook. Detailed steps and codes are provided below.

29. Download Anaconda 3-2022.10 for Windows 10, open Anaconda from "Anaconda Navigator".
30. Click "Environments" on the left and "Create" on the bottom.
31. Name the environment, select Python 3.7.13 and click "Create".
32. After the environment is set up, select "Not installed".
33. Search and select TensorFlow (2.9.1), Keras (2.9.0), NumPy (1.22.4), and matplotlib(3.5.1). Click "Apply" on the bottom right.
34. Return to the Anaconda home page and select "All applications" "on" to the environment.
35. Install Jupyter Notebook. The codes for image processing and training can be found on Zenodo:https://doi.org/10.5281/zenodo.7509014.
36. Launch Jupyter Notebook from Anaconda, open the file "High_pass_code.ipynb", and run the following codes to import packages for array calculation, data visualization, and operating system interface.

```
>%matplotlib inline
>import numpy as np
>import matplotlib.pyplot as plt
>import matplotlib.image as mpimg
>import os
```

37. Load the 8-bit PNG images.

```
>#8bit png dirctory
>RAW_PNG_DIR = ('E:\\raw')
```

38. Create a folder for the output files and set the directory by running the next cell.

```
>#find edge png save dirctory
>EDGE_PNG_SAVE_DIR = ('E:\\Highpass')
```

39. The following codes help locate the files.

```
>def prepend(list, str):
>    str += '{0}'
```

```
>    list = [str.format(i) for i in list]
>    return(list)
>png_files = os.listdir(RAW_PNG_DIR)
>png_prepend = RAW_PNG_DIR+"\\"
>png_dir = prepend(png_files,png_prepend)
```

40. Define the vertical filter and horizontal filter as arrays.

```
>vertical_filter = [[-1,-2,-1], [0,0,0], [1,2,1]]

>horizontal_filter = [[-1,0,1], [-2,0,2], [-1,0,1]]

>file_count = 0
```

41. Run the following codes to apply a Sobel operator.

```
>for i in png_dir:
>   img = mpimg.imread(i)
>   n,m = img.shape # n=number of pixels in the row of the image
             # m=number of pixels in the column of the image
>   edges_img = np.zeros_like(img)
>   for row in range(3,n-2):
>     for col in range(3,m-2):
>       local_pixels = img[row-1:row+2, col-1:col+2]
>       vertical_transformed_pixels = vertical_filter*local_pixels
>       vertical_score = vertical_transformed_pixels.sum()
>       horizontal_transformed_pixels = horizontal_filter*local_pixels
>       horizontal_score = horizontal_transformed_pixels.sum()
>       edge_score = (vertical_score**2 + horizontal_score**2)**.5
>       #print("edge score",(edge_score)*2)
>       #print("edge row col",edges_img[row,col] )
>       edge_score = (edge_score)**0.8
>       if edge_score >= 0.2:
>         edge_score=edge_score**0.6
>       edges_img[row,col]= edge_score
>     plt.imsave(EDGE_PNG_SAVE_DIR + "\\edge_"
>         + png_files[file_count],
edges_img, cmap = 'gray')
>     file_count = file_count+1
>     print(file_count, "/", len(png_dir))
>print("Done")
```

Split each image into 100 patches of sub-images with PhotoScape X

Timing: 10 min

This describes the detailed steps for splitting images using PhotoScape X.

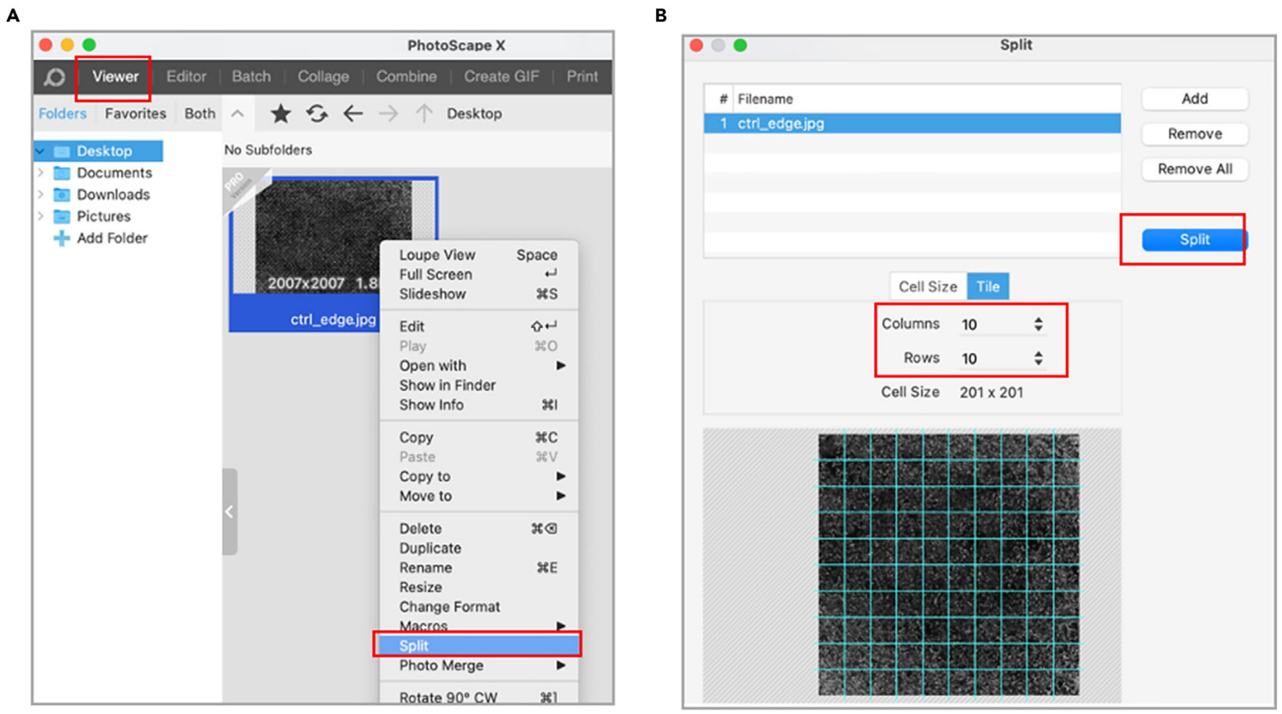


Figure 4. Split the images with Photoscape X
(A) Home screen of Photoscape X.
(B) Specify the number of columns and rows.

42. In PhotoScape X (Windows 10), open the file under "Viewer" (Figure 4A).
43. Right-click the image and select "Split" (Figure 4A).
44. Change the numbers of "Columns" and "Rows" and click "Split" (Figure 4B).

Train a Conv2D model

Timing: 30 min

A convolutional neural network model is trained in this step. Detailed procedures are provided below.

45. To randomly split the data into training and testing sets, we run the following script and assign the files into corresponding subfolders (Figure 5).

```
>import pandas as pd
>import numpy as np
># set up random state seed for reproducibility.
>seed = 12
>training_frac = 0.9
># load meta data
>meta_data = pd.read_csv('meta_data.csv', index_col=0)
```

```
>meta_data["ids"] = list(meta_data.index)
># the splitting is based on 'drug','concentration','treatment'
># training set
>meta_train = meta_data.groupby(['drug','concentration','treatment']).apply(
lambda x: x.sample(frac=training_frac, random_state=seed))
>meta_train.index = meta_train["ids"]
>meta_train.sort_index(inplace=True)
>meta_train.to_csv('meta_data_training.csv')
># testing set
>meta_test = meta_data.loc[set(meta_data.index) - set(meta_train.index)]
>meta_test.sort_index(inplace=True)
>meta_test.to_csv('meta_data_testing.csv')
```

46. Launch Jupyter Notebook from Anaconda and open the following file "Conv2D_code.ipynb".

```
>import tensorflow as tf
>import keras_preprocessing
>from keras_preprocessing import image
>from keras_preprocessing.image import ImageDataGenerator
```

47. Data augmentation.[10]

```
>TRAINING_DIR = ('D:\\Fasudil\\con1ctrl\\train')
>training_datagen = ImageDataGenerator(
>    rescale = 1./255,
>    rotation_range=40,
>    width_shift_range=0.2,
>    height_shift_range=0.2,
>    shear_range=0.2,
>    zoom_range=0.2,
>    horizontal_flip=True,
>    fill_mode='nearest')
```

48. Load the test set and rescale the data.

```
>VALIDATION_DIR = ('D:\\Fasudil\\con1ctrl\\test')
>validation_datagen = ImageDataGenerator(rescale = 1./255)
```

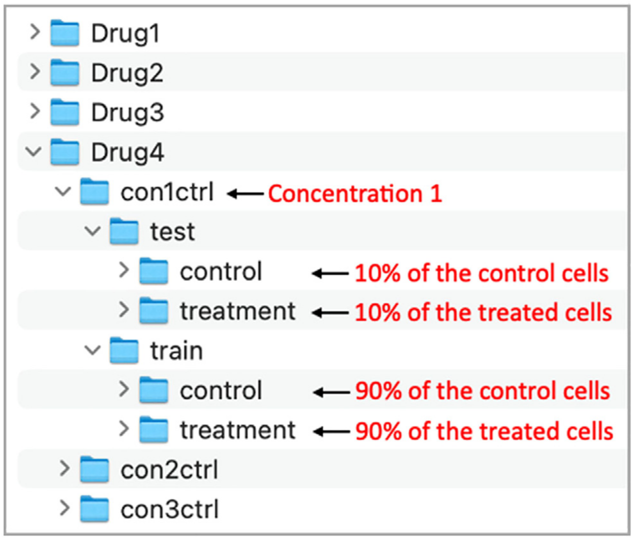


Figure 5. Create folders for training and testing

49. Set the image size in the train to "(201, 201)", choose "binary classification" and set the batch size for training.

```
>train_generator = training_datagen.flow_from_directory(
>    TRAINING_DIR,
>    target_size=(201,201),
>    class_mode='binary',
>    batch_size=10)
```

50. Resize the images in the test set, choose "binary classification" and set the batch size for testing.

```
>validation_generator = validation_datagen.flow_from_directory(
>    VALIDATION_DIR,
>    target_size=(201,201),
>    class_mode='binary',
>    batch_size=10)
```

51. Train Conv2D.

```
>model = tf.keras.models.Sequential([
>    # This is the first convolution
>    tf.keras.layers.Conv2D(64, (3,3), activation='relu',
>    input_shape=(196, 196, 3)),
>    tf.keras.layers.MaxPooling2D(2, 2),
>    # The second convolution
>    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
>    tf.keras.layers.MaxPooling2D(2,2),
>    # The third convolution
>    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
>    tf.keras.layers.MaxPooling2D(2,2),
>    # The fourth convolution
>    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
>    tf.keras.layers.MaxPooling2D(2,2),
>    # Flatten the results to feed into a DNN
>    tf.keras.layers.Flatten(),
>    tf.keras.layers.Dropout(0.5),
>    # 512 neuron hidden layer
>    tf.keras.layers.Dense(512, activation='relu'),
>    tf.keras.layers.Dense(1, activation='sigmoid')])
>model.summary()
```

52. Generate a confusion matrix.

```
>model.compile(loss = tf.keras.losses.BinaryCrossentropy(),
>       optimizer = tf.keras.optimizers.Adam(learning_rate=1e-3),
>       metrics = [tf.keras.metrics.BinaryAccuracy(),
>            tf.keras.metrics.TruePositives(),
>            tf.keras.metrics.TrueNegatives(),
>            tf.keras.metrics.FalsePositives(),
>            tf.keras.metrics.FalseNegatives()])
```

53. Test the model.

```
>history = model.fit(train_generator,
>         epochs=4,
>         steps_per_epoch=504,
>         validation_data = validation_generator,
>         verbose = 1,
>         validation_steps=56)
```

54. After the training is completed, use "val_accuracy" to plot the curve.

Train a vision transformer model

Timing: 30 min

A vision transformer model is trained in this step. Details are provided below.

55. Launch Jupyter Notebook from Anaconda and open the following file"Vision_Transformer_code.ipynb". Import packages.

```
>import numpy as np
>import tensorflow as tf
>from tensorflow import keras
>from tensorflow.keras import layers
>import tensorflow_addons as tfa
>from keras_preprocessing import image
>from keras_preprocessing.image import ImageDataGenerator
```

56. Load the images.

```
>num_classes = 2
>input_shape = (201, 201, 3)
>TRAINING_DIR = ('D:\\all-con1ctrl\\c-con1ctrl\\train')
>training_datagen = ImageDataGenerator(rescale = 1./255)
>VALIDATION_DIR = ('D:\\all-con1ctrl\\c-con1ctrl\\test')
>validation_datagen = ImageDataGenerator(rescale = 1./255)
```

57. Choose binary classification and specify batch sizes.

```
>train_generator = training_datagen.flow_from_directory(
>    TRAINING_DIR,
>    target_size=(201,201),
>    class_mode='binary',
>    batch_size=200)
>validation_generator = validation_datagen.flow_from_directory(
```

```
>    VALIDATION_DIR,
>    target_size=(201,201),
>    class_mode='binary',
>    batch_size=200)
```

58. Check matrix shape.

```
>x_train, y_train = train_generator.next()
>x_test, y_test = validation_generator.next()
>print(f"x_train shape: {x_train.shape} - y_train shape: {y_train.shape}")
>print(f"x_test shape: {x_test.shape} - y_test shape: {y_test.shape}")
```

59. Set training parameters.

```
>learning_rate = 0.001
>weight_decay = 0.0001
>batch_size = 200
>num_epochs = 100
>image_size = 72
>patch_size = 6
>num_patches = (image_size // patch_size) ** 2
>projection_dim = 64
>num_heads = 4
>transformer_units = [
>    projection_dim * 2,
>    projection_dim,]
>transformer_layers = 4
>mlp_head_units = [2048, 1024]
```

60. Data augmentation.

```
>data_augmentation = keras.Sequential([
>         layers.Normalization(),
>         layers.Resizing(image_size, image_size),
>         layers.RandomFlip("horizontal"),
>         layers.RandomRotation(factor=0.02),
>         layers.RandomZoom(
>           height_factor=0.2, width_factor=0.2),],
>  name="data_augmentation",)
>data_augmentation.layers[0].adapt(x_train)
```

61. Define multilayer perceptron and patches.

```
>def mlp(x, hidden_units, dropout_rate):
>    for units in hidden_units:
>      x = layers.Dense(units, activation=tf.nn.gelu)(x)
>      x = layers.Dropout(dropout_rate)(x)
>    return x
>class Patches(layers.Layer):
>    def __init__(self, patch_size):
>      super(Patches, self).__init__()
>      self.patch_size = patch_size
>    def call(self, images):
>      batch_size = tf.shape(images)[0]
>      patches = tf.image.extract_patches(
>          images=images,
>          sizes=[1, self.patch_size, self.patch_size, 1],
>          strides=[1, self.patch_size, self.patch_size, 1],
>          rates=[1, 1, 1, 1],
>          padding="VALID",)
>      patch_dims = patches.shape[-1]
>      patches = tf.reshape(patches, [batch_size, -1, patch_dims])
>      return patches
```

62. Resize images.

```
>import matplotlib.pyplot as plt
>plt.figure(figsize=(4, 4))
>image = x_train[np.random.choice(range(x_train.shape[0]))]
>plt.imshow(image.astype("uint8"))
>plt.axis("off")
>resized_image = tf.image.resize(
     tf.convert_to_tensor([image]), size=(image_size, image_size))
>patches = Patches(patch_size)(resized_image)
>print(f"Image size: {image_size} X {image_size}")
>print(f"Patch size: {patch_size} X {patch_size}")
>print(f"Patches per image: {patches.shape[1]}")
>print(f"Elements per patch: {patches.shape[-1]}")
>n = int(np.sqrt(patches.shape[1]))
```

```
>plt.figure(figsize=(4, 4))
>for i, patch in enumerate(patches[0]):
>    ax = plt.subplot(n, n, i + 1)
>    patch_img = tf.reshape(patch, (patch_size, patch_size, 3))
>    plt.imshow(patch_img.numpy().astype("uint8"))
>    plt.axis("off")
```

63. Encode patches and positions.

```
>class PatchEncoder(layers.Layer):
>    def __init__(self, num_patches, projection_dim):
>        super(PatchEncoder, self).__init__()
>        self.num_patches = num_patches
>            self.projection = layers.Dense(units=projection_dim)
>        self.position_embedding = layers.Embedding(
>            input_dim=num_patches, output_dim=projection_dim)
>    def call(self, patch):
>        positions = tf.range(start=0, limit=self.num_patches, delta=1)
>        encoded = self.projection(patch) + self.position_embedding(pos > itions)
>        return encoded
```

64. Set the parameters of the model.

```
>def create_vit_classifier():
>    inputs = layers.Input(shape=input_shape)
>    # Augment data.
>    augmented = data_augmentation(inputs)
>    # Create patches.
>    patches = Patches(patch_size)(augmented)
>    # Encode patches.
>    encoded_patches = PatchEncoder(num_patches, projection_dim)(patches)
>    # Create multiple layers of the Transformer block.
>    for _ in range(transformer_layers):
>            # Layer normalization 1.
>        x1 = layers.LayerNormalization(epsilon=1e-6)(encoded_patches)
>        # Create a multi-head attention layer.
>        attention_output = layers.MultiHeadAttention(
>        num_heads=num_heads, key_dim=projection_dim, dropout=0.1)
>        (x1, x1)
>        # Skip connection 1.
>        x2 = layers.Add()([attention_output, encoded_patches])
>        # Layer normalization 2.
>        x3 = layers.LayerNormalization(epsilon=1e-6)(x2)
>        # MLP.
>        x3 = mlp(x3, hidden_units=transformer_units, dropout_rate=0.1)
>        # Skip connection 2.
>        encoded_patches = layers.Add()([x3, x2])
>    # Create a [batch_size, projection_dim] tensor.
>    representation = layers.LayerNormalization(epsilon=1e- > 6)(encoded_patches)
>    representation = layers.Flatten()(representation)
>    representation = layers.Dropout(0.5)(representation)
>    # Add MLP.
>    features = mlp(representation, hidden_units=mlp_head_units, dropout_rat > e=0.5)
>    # Classify outputs.
>    logits = layers.Dense(1,activation='sigmoid')(features)
>    # Create the Keras model.
>    model = keras.Model(inputs=inputs, outputs=logits)
>    return model
```

65. Train the transformer model.

```
>def run_experiment(model):
>    optimizer = tfa.optimizers.AdamW(
>        learning_rate=learning_rate, weight_decay=weight_decay)
>    model.compile(
>        optimizer=optimizer,
>        loss=keras.losses.BinaryCrossentropy(from_logits=True),
>        metrics = [ tf.keras.metrics.BinaryAccuracy(),
>              tf.keras.metrics.TruePositives(),
>              tf.keras.metrics.TrueNegatives(),
>              tf.keras.metrics.FalsePositives(),
>              tf.keras.metrics.FalseNegatives()])
>    history = model.fit(
>        x=x_train,
>        y=y_train,
>        batch_size=batch_size,
>        epochs=num_epochs,
>        validation_split=0.1,)
>    return history
>vit_classifier = create_vit_classifier()
>history = run_experiment(vit_classifier)
```

66. Check the accuracy of the model using the whole test set.

```
>score = vit_classifier.evaluate(x_test, y_test, verbose = 0)
```

Quantify cell numbers using ImageJ

Timing: 30 min

This describes the detailed procedures for the quantifications of cells with ImageJ.

67. Open the file (Figure 6A).
68. Select "Process > Subtract Background", adjust the parameters in the popup window, and click "OK" (Figure 6B).
69. Select "Image > Adjust > Threshold", adjust the parameters in the popup window, and click "Apply" (Figure 6B).
70. Select "Process > Binary > Fill Holes" (Figure 6C).
71. Select "Process > Binary > Convert to Mask" (Figure 6C).
72. Select "Process > Binary > Watershed" (Figure 6C).
73. Select "Analyze > Analyze Particles", adjust the parameters in the popup window, and click "OK" (Figure 6D).
74. Check the result (Figure 6D).

Calculate IC50s using GraphPad Prism 9.3.1 for Windows 10.

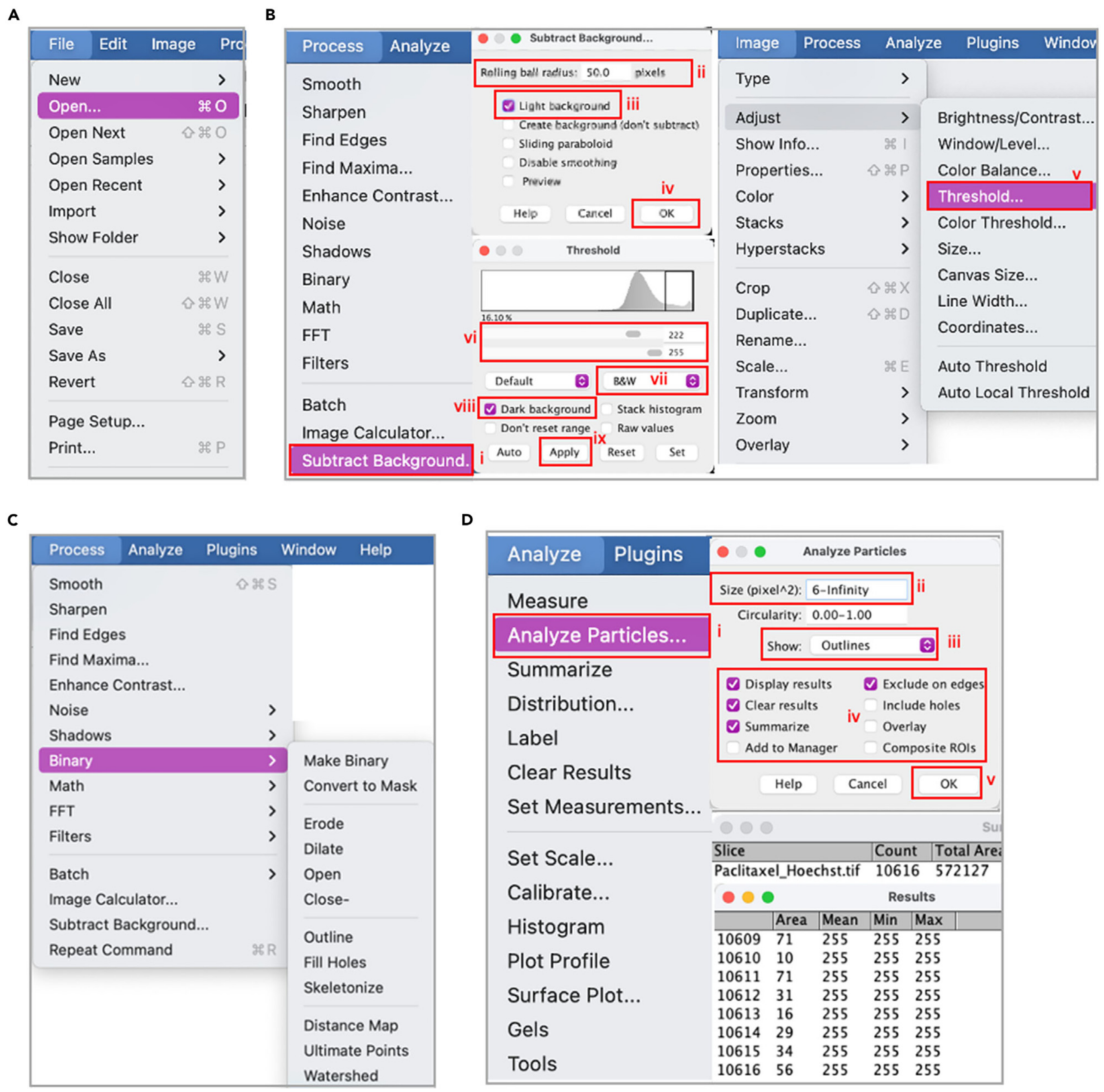
Timing: 10 min

The IC50s can be determined using GraphPad. Detailed steps are described below.

75. Select "File > New Project File", select "XY", "enter 3 replicates values in side-by-side subcolumns" and click "Create" (Figure 7A).
76. Paste data and click "Data 1" under "Graphs" (Figure 7B).
77. Select "Graph Type" in the popup window and click "OK" (Figure 7C).
78. Double-click the X-axis on the graph, select "Log 10" for "Scale", "Short" for "Ticks length", "Power of 10" for "Format" and click "OK" (Figure 7D).
79. Click "Analyze", select "Nonlinear regression (curve fit)" under "XY analyses" and click "OK" (Figure 7E).
80. Select "[Inhibitor] vs. response – Variable slope (four parameters) [2]" and click "OK" (Figure 7F).
81. Check the IC50 value (Figure 7G).

EXPECTED OUTCOMES

The main outcomes of the workflows are two deep learning models, producing confusion matrices for binary classification. We expect to see a sharp increase in the prediction accuracy at concentrations close to the IC50 of the drug because the model can better distinguish the untreated cells from



Rolling ball radius: 50.0 pixels
[✓] Light background
[ ] Create background (don't subtract)
[ ] Sliding paraboloid
[ ] Disable smoothing
[ ] Preview

16.10 %
222
255
Default
B&W
[✓] Dark background
[ ] Don't reset range
[ ] Stack histogram
[ ] Raw values

Size (pixel^2): 6-Infinity
Circularity: 0.00-1.00
Show: Outlines
[✓] Display results
[✓] Clear results
[✓] Summarize
[ ] Add to Manager
[✓] Exclude on edges
[ ] Include holes
[ ] Overlay
[ ] Composite ROIs

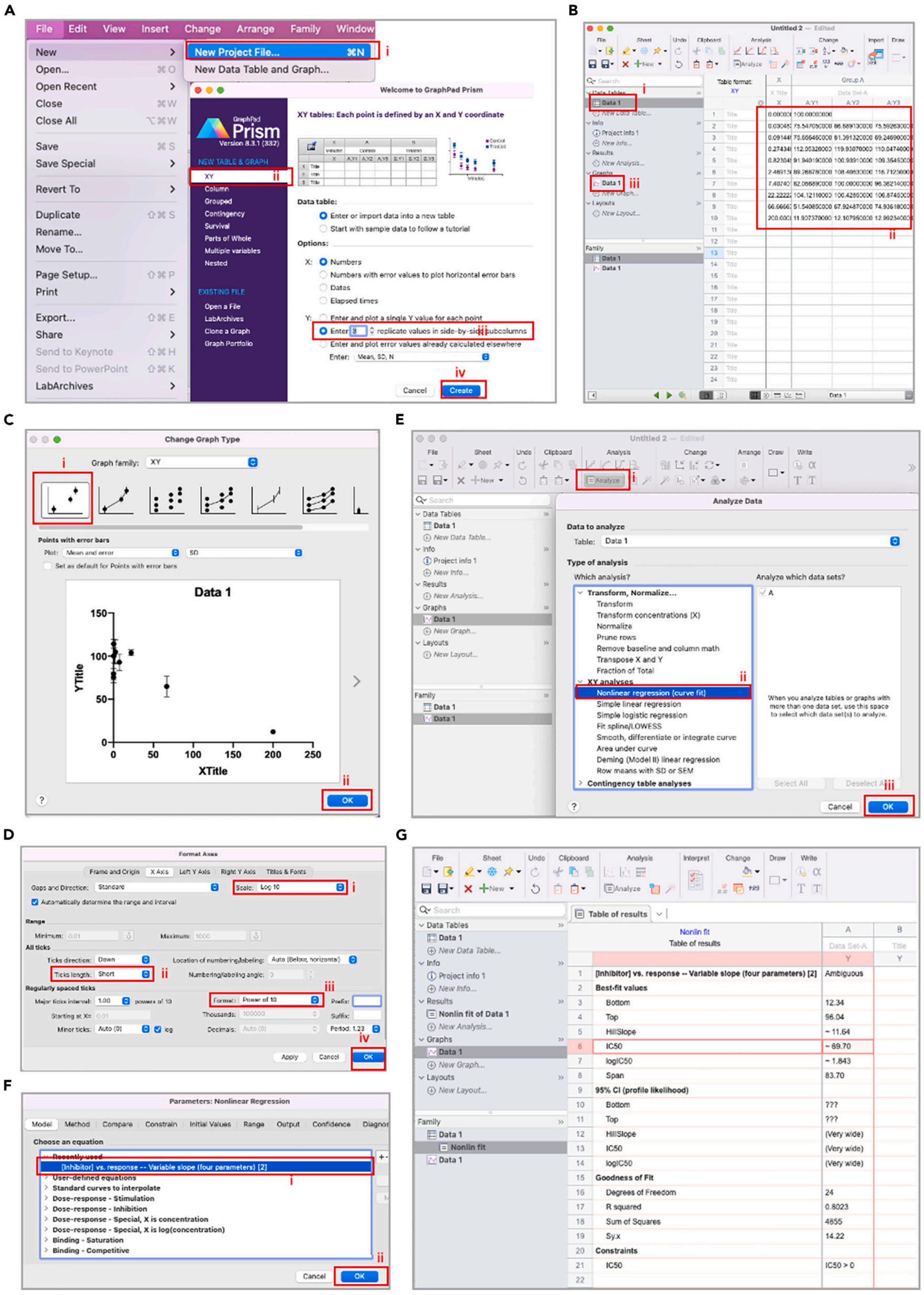
Slice	Count	Total Area
Paclitaxel_Hoechst.tif	10616	572127

	Area	Mean	Min	Max
10609	71	255	255	255
10610	10	255	255	255
10611	71	255	255	255
10612	31	255	255	255
10613	16	255	255	255
10614	29	255	255	255
10615	34	255	255	255
10616	56	255	255	255

Figure 6. Count nuclei with ImageJ
(A–C) Open the file and modify the image.
(D) Set the parameters for "Analyze Particles".

treated cells at a high concentration. This method allows for multi-time-point evaluations of drug potency and requires approximately 20 min for each round of imaging and prediction. By contrast, conventional cell viability assays are performed at the experimental endpoint when the cells are lysed. In addition, chemicals with absorbance around 450 nm or 570 nm may interfere with the readouts for WST-8 assays and MTT assays, respectively. Their effects on cell viability cannot be assessed using these assays but can be evaluable with our method.

Although the step-by-step methodology is intended to provide an alternative for high-throughput screening of anticancer drugs, we expect this protocol to be adapted for other purposes such as


A
B
C
D
E
F
G
New Project File...
New Data Table and Graph...
Welcome to GraphPad Prism
XY
Create
Change Graph Type
Graph family: XY
Data 1
YTitle
XTitle
OK
Format Axes
Scale: Log 10
Ticks length: Short
Format: Power of 10
Analyze Data
Nonlinear regression (curve fit)
Parameters: Nonlinear Regression
[Inhibitor] vs. response -- Variable slope (four parameters) [2]
i
ii
iii
iv

	Nonlin fit Table of results	A Data Set-A Y	B Title Y
1	[Inhibitor] vs. response -- Variable slope (four parameters) [2]	Ambiguous	
2	Best-fit values		
3	Bottom	12.34	
4	Top	96.04	
5	HillSlope	~ 11.64	
6	IC50	~ 69.70	
7	logIC50	~ 1.843	
8	Span	83.70	
9	95% CI (profile likelihood)		
10	Bottom	???	
11	Top	???	
12	HillSlope	(Very wide)	
13	IC50	(Very wide)	
14	logIC50	(Very wide)	
15	Goodness of Fit		
16	Degrees of Freedom	24	
17	R squared	0.8023	
18	Sum of Squares	4855	
19	Sy.x	14.22	
20	Constraints		
21	IC50	IC50 > 0	
22			

Figure 7. Calculate IC50s using GraphPad Prism

(A–D) Create a new project and fill in the table. Select the graph type and settings for the x-axis.

(E–G) Select analysis and equation to generate the IC50 values.

screening drugs that affect the differentiation of stem cells and evaluating the effects of chemicals on the density and morphology of cells.

LIMITATIONS

Although many companies and laboratories specializing in drug development have access to high-throughput imaging systems such as Pico or Cytation 5, one constraint of this workflow is that it relies on such imaging systems, which might not be readily available for every lab. Another drawback is that our method for IC50 prediction can be less accurate when a larger dilution ratio is used because we define the IC50 as the average of two adjacent concentrations between which a sharp increase in classification accuracy is observed. The accuracy of IC50 prediction depends on the ratio at which the drug is diluted. For example, if the dilution ratio (DR) is 5, the predicted IC50 can be less accurate than when the ratio is 3. However, mapping the IC50 from a large range of concentrations (0.03 μM–200 μM, 6666-fold) to a 2-fold variation (DR = 3) using our method can be helpful and sufficient for many purposes. In addition, 1.5-fold or 1.25-fold dilutions could be adopted in the second round to further improve the prediction accuracy. This third limitation is that this method is primarily developed to evaluate the efficacy of molecules or treatments that can change cell density or morphology. For treatments that do not alter these properties, other methods should be considered.

TROUBLESHOOTING

Problem 1

The cells in the outermost layer of the multi-well plate do not proliferate normally, probably because the culture media evaporate faster. Consequently, the concentration of the drug can be changed. Using the cell counts from these wells can make it challenging to interpret the data (Related to step 19).

Potential solution

Cells can still be seeded in the outermost layer. However, the cell count data from these wells should be excluded from downstream analysis. Having media in these wells can help the adjacent wells maintain the appropriate humidity.

Problem 2

In general, the cells in all the wells are evenly distributed, regardless they are treated with drugs or not. However, in some cases, large empty spots (regions without any cells) are observed, which can be due to vigorous pipetting or contacting the cells with pipette tips, resulting in inaccurate downstream analyses (Related to step 11).

Potential solution

This problem can be avoided by positioning the pipette tips on the inner wall of the wells and pushing out the solution gently and slowly.

Problem 3

To compare the IC50 values generated using conventional methods and deep learning methods, ImageJ is needed to count the cells whose nuclei are stained with Hoechst. Different cell lines may vary in size. In some cases, the settings described in Figure 6 can generate inaccurate cell counts (Related to step 73).

Potential solution

The following parameters can be modified for accurate cell count, the "pixels" option under "Rolling ball radius", "Threshold" (Figure 6B), and "6-infinity" under "Size(pixel^2)" (Figure 6D).

Problem 4

Error messages may pop up when training the models for the first time (Related to step 31).

Potential solution

Proper versions of software and packages are essential for using image processing and model training (key resources table).

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, Aijun Wang (aawang@ucdavis.edu).

Materials availability

The study did not generate new unique reagents or other materials.

Data and code availability

The codes and data are available at Zenodo:https://doi.org/10.5281/zenodo.7509014. Any additional information can be provided by the lead contact upon reasonable request. The link to the website is biochemml.com/image/.

ACKNOWLEDGMENTS

This work was supported by an NIH grant R01CA247685 awarded to K.S.L. The graphical abstract figures (steps 1 and 2) were created with BioRender.com. The Conv2D model was generated on app.diagrams.net.

AUTHOR CONTRIBUTIONS

Conceptualization, Y.W., W.Z., J.Z., K.S.L., A.W.; methodology, Y.W., W.Z.; software, Y.W.; validation, Y.W., W.Z., C.Q., H.H.; investigation, Y.W., W.Z.; resources, A.W., K.S.L.; data curation, Y.W., W.Z.; writing - original draft, Y.W., W.Z., Y.W.; writing - review & editing, Y.W., W.Z., Y.W., C.Q., H.H., T.L., S.L., J.Z., K.S.L., A.W.; visualization, Y.W., W.Z., Y.W., C.Q.; supervision, J.Z., A.W., K.S.L.; project administration, Y.W.; funding acquisition, K.S.L.

DECLARATION OF INTERESTS

The authors, on behalf of the University of California, Davis, are preparing a patent application covering aspects of the work presented here.

REFERENCES

1. Wang, Y., Zhang, W., Yip, H., Qu, C., Hu, H., Chen, X.,., and Lam, K.S. (2023). SIC50: determining drug inhibitory concentrations using a vision transformer and an optimized Sobel operator. Patterns. https://doi.org/10.1016/j.patter.2023.100686.
2. Lecun, Y., and Bengio, Y. (1995). Convolutional networks for images, speech, and time-series. In Conference: The Handbook of Brain Theory and Neural Networks. https://doi.org/10.5555/303568.303704.
3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. Adv. Neural Inf. Proces. Syst. 5998–6008. https://doi.org/10.48550/arXiv.1706.03762.
4. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An image is worth 16x16 words: transformers for image recognition at scale. Preprint at arXiv. https://doi.org/10.48550/arXiv.2010.11929.
5. Florento, I., Matias, R., Tuaño, E., Santiago, K., Dela Cruz, F., and Tuazon, A. (2012). Comparison of cytotoxic activity of anticancer drugs against various human tumor cell lines using in vitro cell-based approach. Int. J. Biomed. Sci. *8*, 76–80.
6. Van Rossum, G., and Drake, F.L. (2009). Python 3 Reference Manual (CreateSpace). https://doi.org/10.5555/1593511.
7. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., and Zheng, X. (2016). {TensorFlow}: a system for {Large-Scale} machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283. https://doi.org/10.48550/arXiv.1605.08695.
8. Oliphant, T.E. (2006). A Guide to NumPy, *1* (Trelgol Publishing), p. 85.
9. Hunter, J.D. (2007). Matplotlib: a 2D graphics environment. Comput. Sci. Eng. *9*, 90–95. https://doi.org/10.1109/MCSE.2007.55.
10. Shorten, C., and Khoshgoftaar, T.M. (2019). A survey on image data augmentation for deep learning. J. Big Data *6*, 60. https://doi.org/10.1186/s40537-019-0197-0.