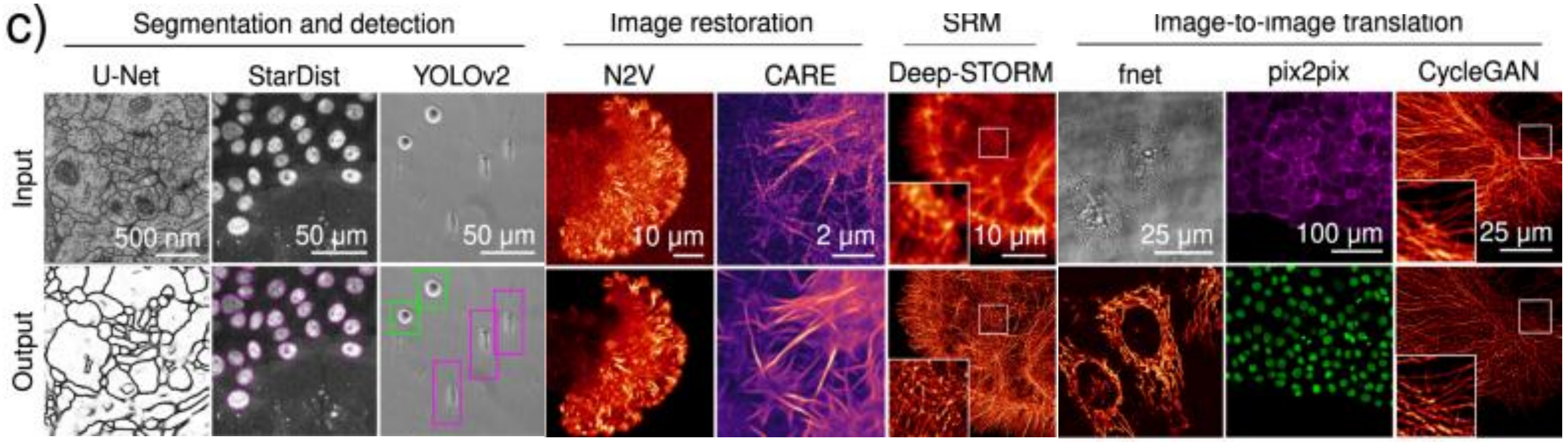
Deep learning in microscopy

27/08/2021

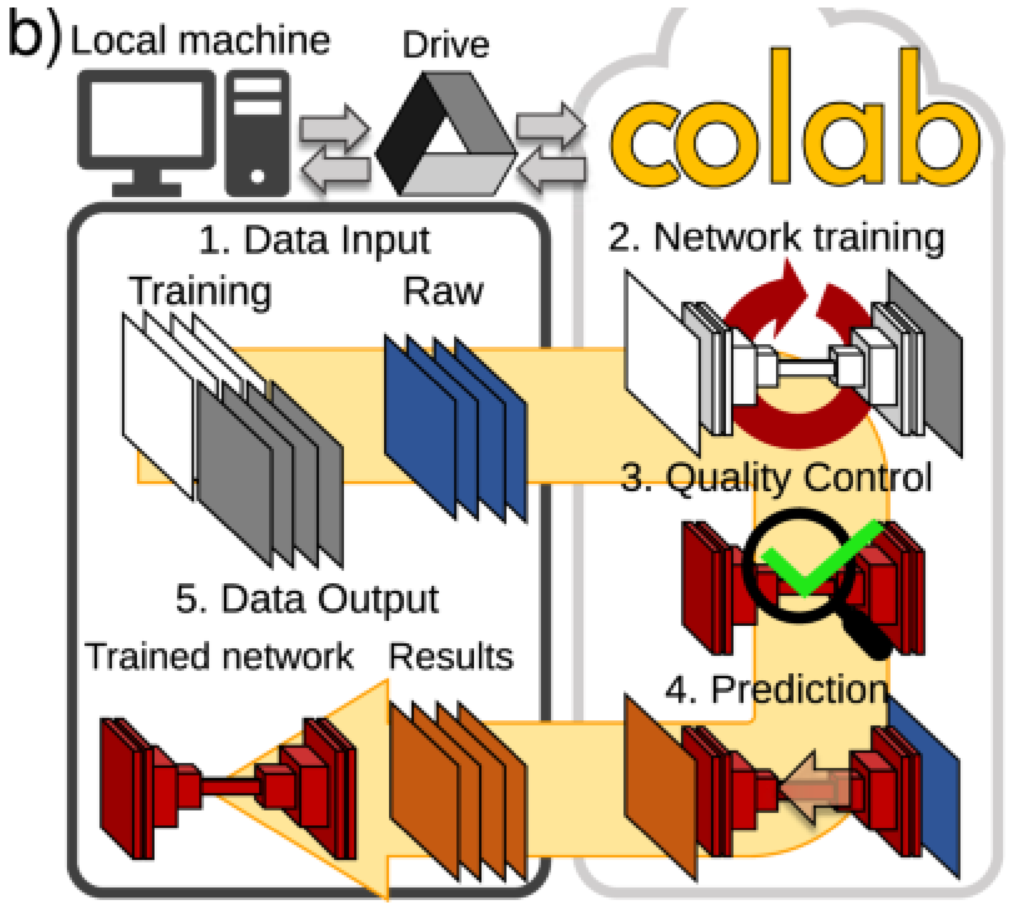
GULBENKIAN SCIENCE

Deep learning in microscopy



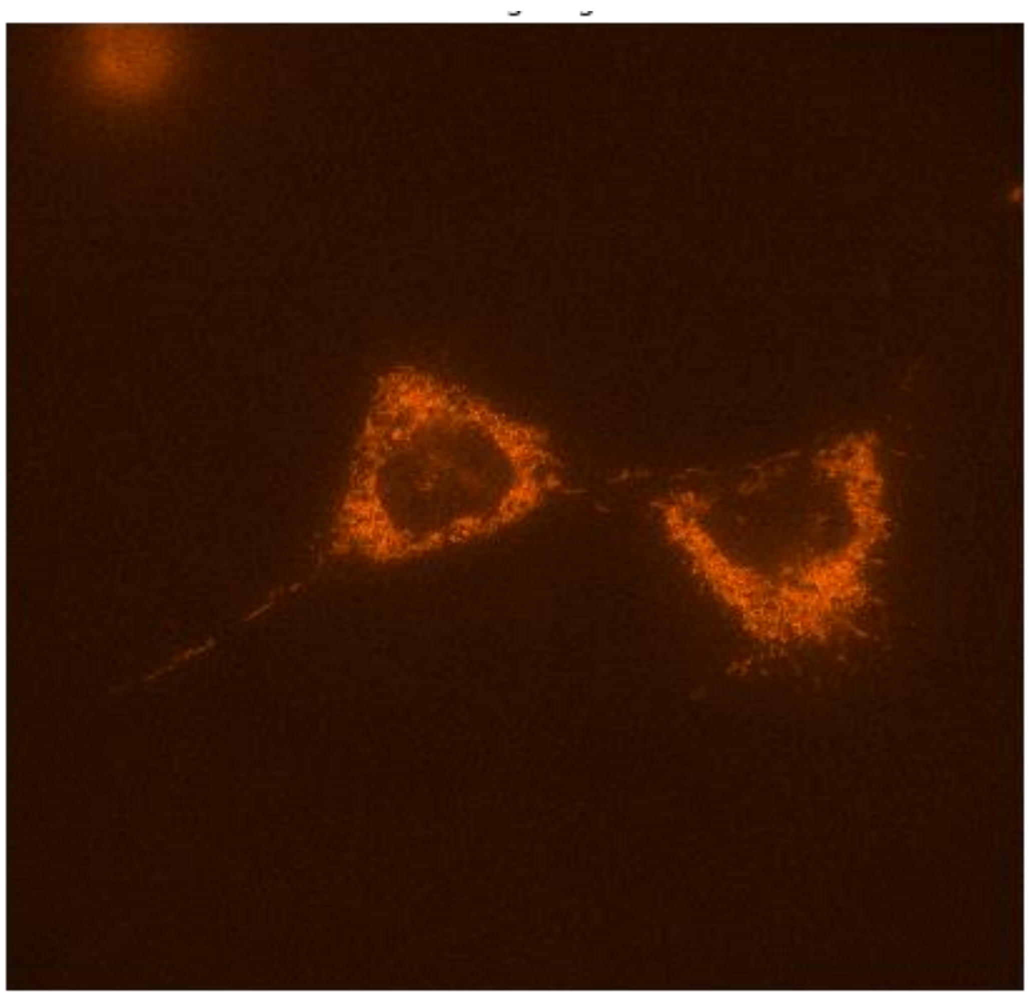
von Chamier & Laine et al., 2020

ZeroCostDL4Mic

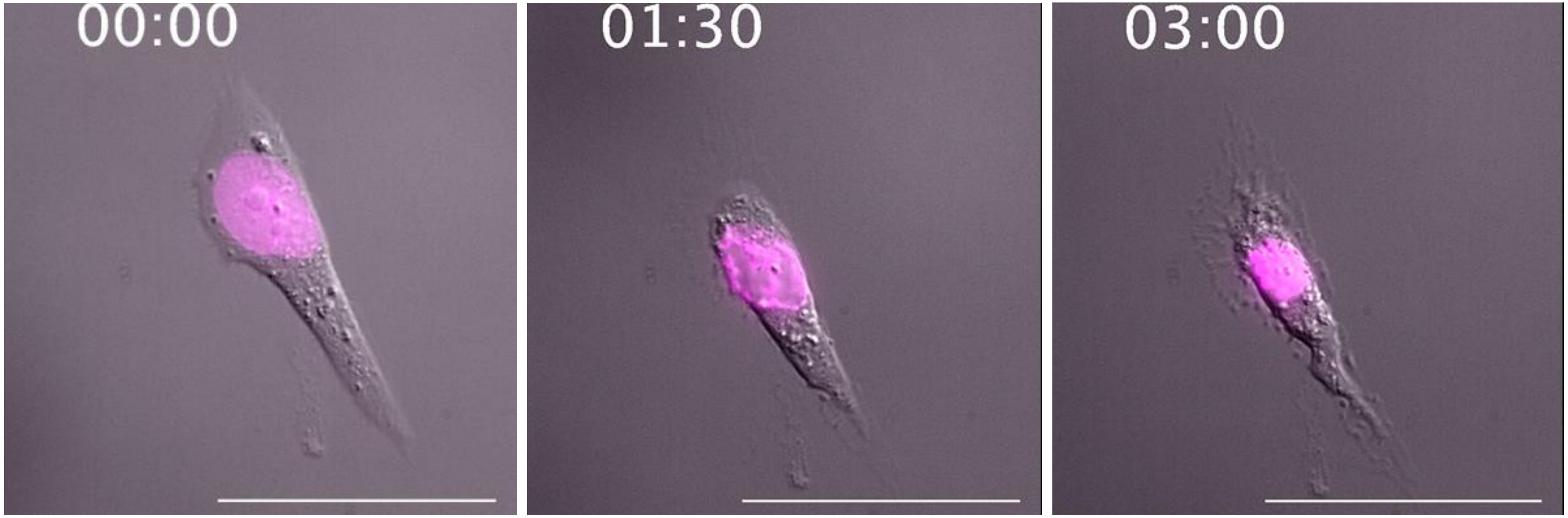


von Chamier & Laine et al., 2020

Fluorescence microscopy



Prolonged light interaction with cellular components lead to damage.



Use of ML approaches to assess photodamage cell features .

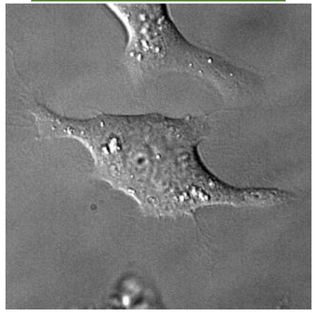
Machine learning

Classifier

- Cell morphology
- Cell division state
- Organelle damage

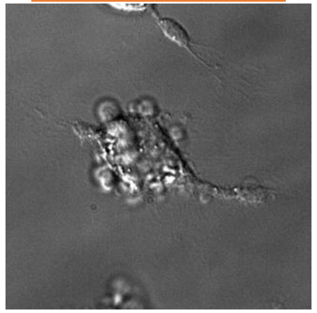
Cell state

No damage



Continue experiment

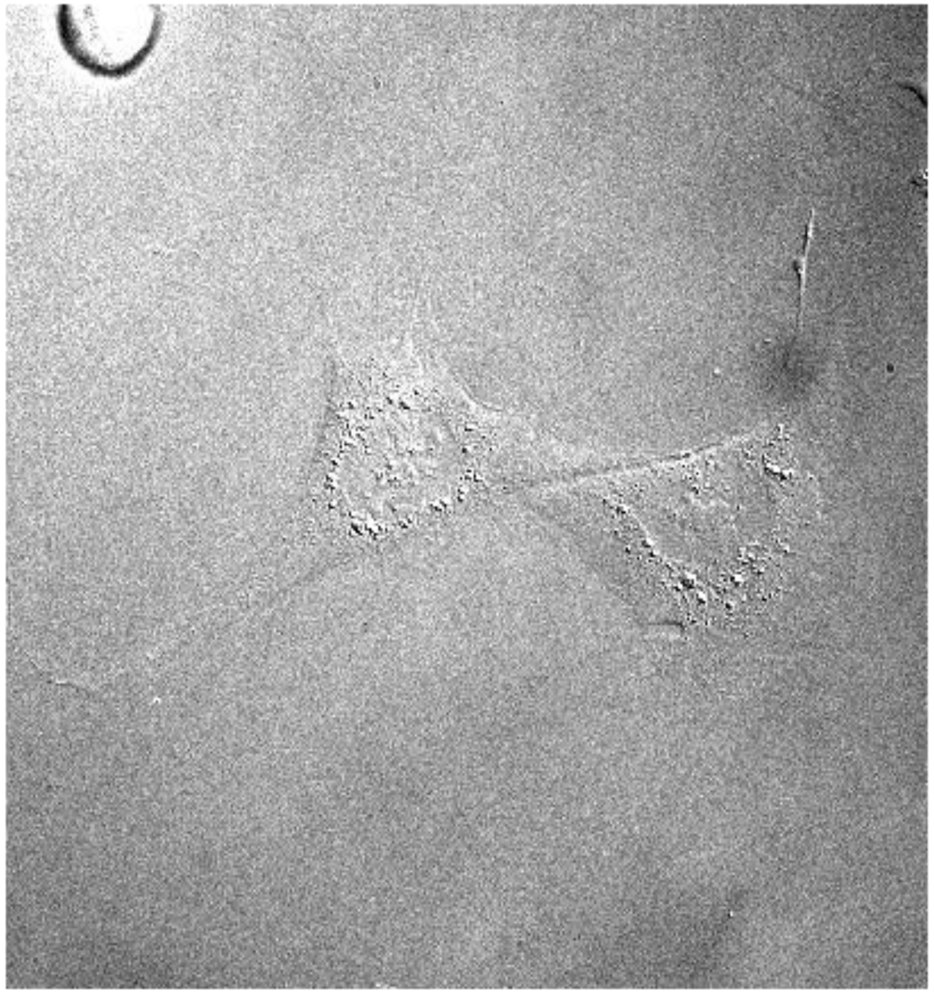
Damage



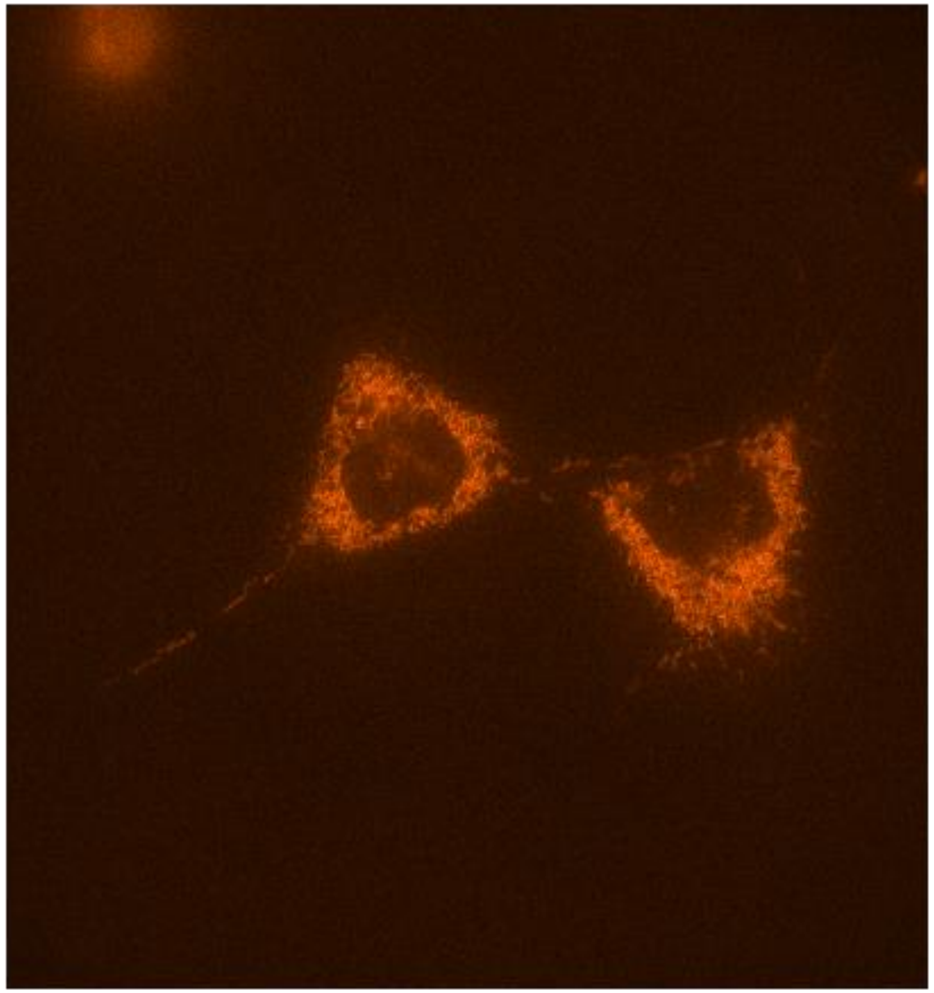
- Alter experimental settings to avoid cell death
- Remove individual cell from data analysis
- Test different conditions to determine cell fitness

Mitochondria segmentation

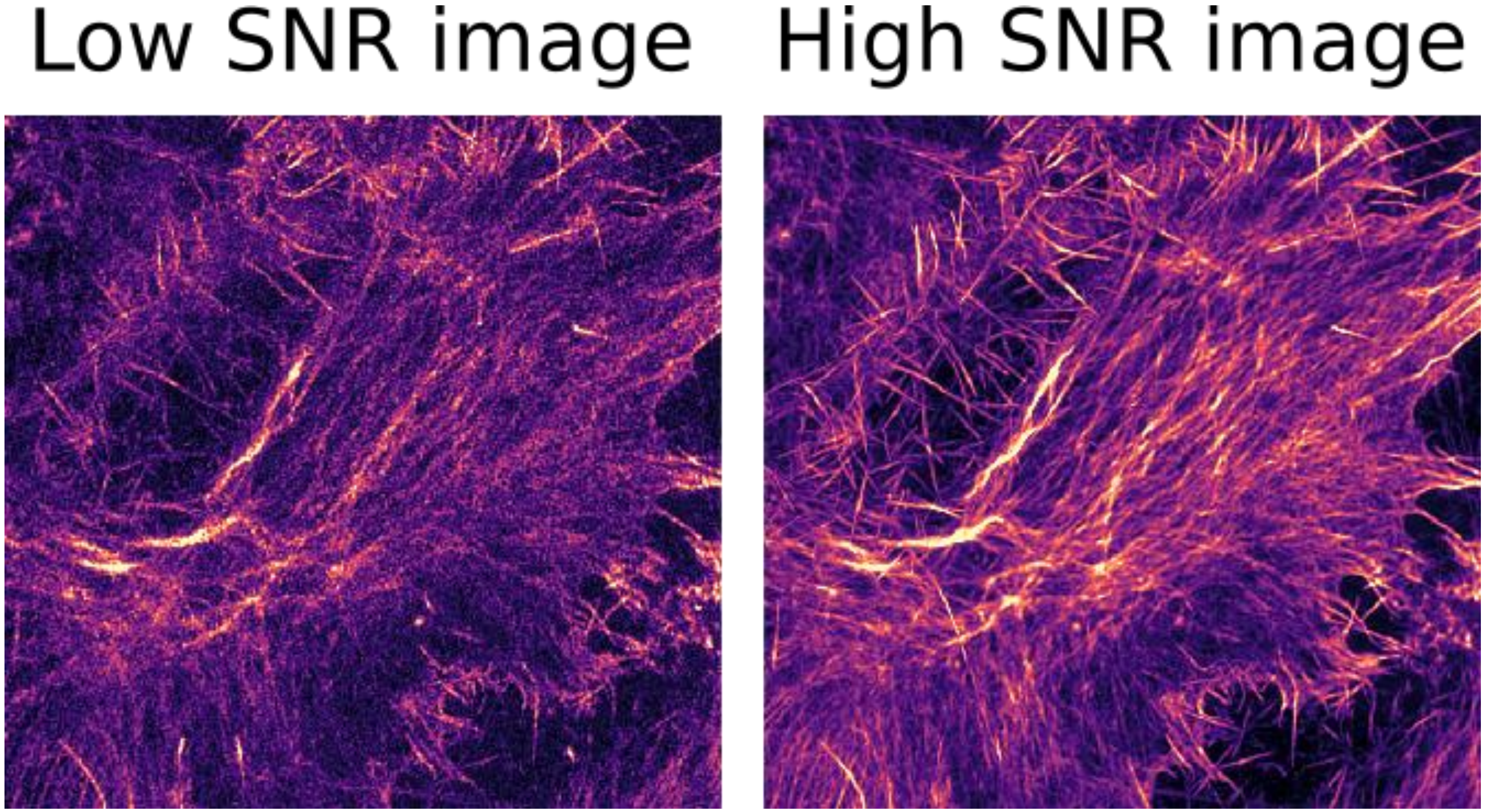
Training source

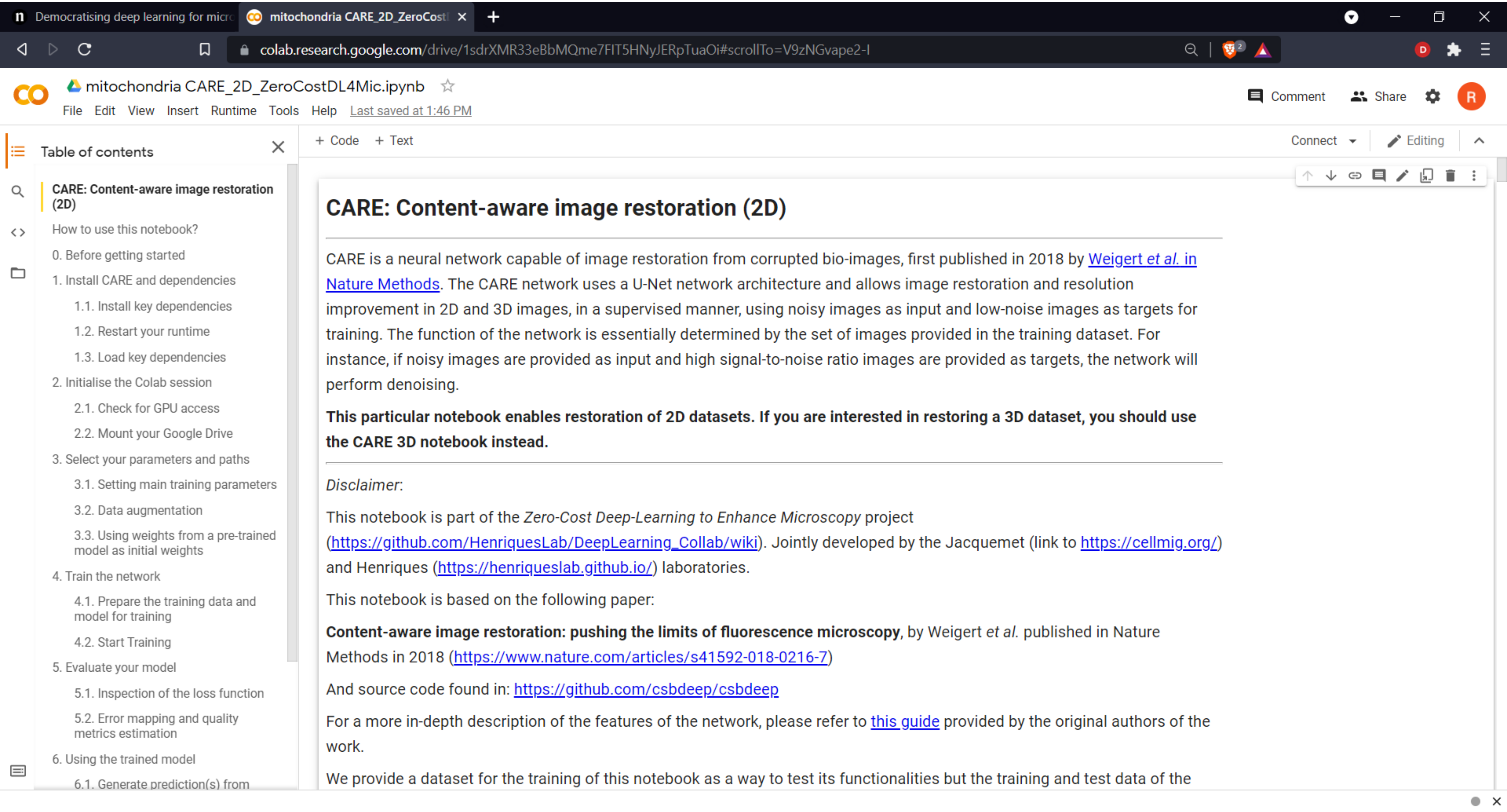


Training target



CARE




Democratising deep learning for micro
mitochondria CARE_2D_ZeroCost
colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=V9zNGvape2-I
mitochondria CARE_2D_ZeroCostDL4Mic.ipynb
File
Edit
View
Insert
Runtime
Tools
Help
Last saved at 1:46 PM
Comment
Share
+ Code
+ Text
Connect
Editing
Table of contents

CARE: Content-aware image restoration (2D)
How to use this notebook?
0. Before getting started
1. Install CARE and dependencies
1.1. Install key dependencies
1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from

CARE: Content-aware image restoration (2D)

CARE is a neural network capable of image restoration from corrupted bio-images, first published in 2018 by Weigert *et al.* in Nature Methods. The CARE network uses a U-Net network architecture and allows image restoration and resolution improvement in 2D and 3D images, in a supervised manner, using noisy images as input and low-noise images as targets for training. The function of the network is essentially determined by the set of images provided in the training dataset. For instance, if noisy images are provided as input and high signal-to-noise ratio images are provided as targets, the network will perform denoising.

This particular notebook enables restoration of 2D datasets. If you are interested in restoring a 3D dataset, you should use the CARE 3D notebook instead.

Disclaimer:

This notebook is part of the *Zero-Cost Deep-Learning to Enhance Microscopy* project (https://github.com/HenriquesLab/DeepLearning_Collab/wiki). Jointly developed by the Jacquemet (link to https://cellmig.org/) and Henriques (https://henriqueslab.github.io/) laboratories.

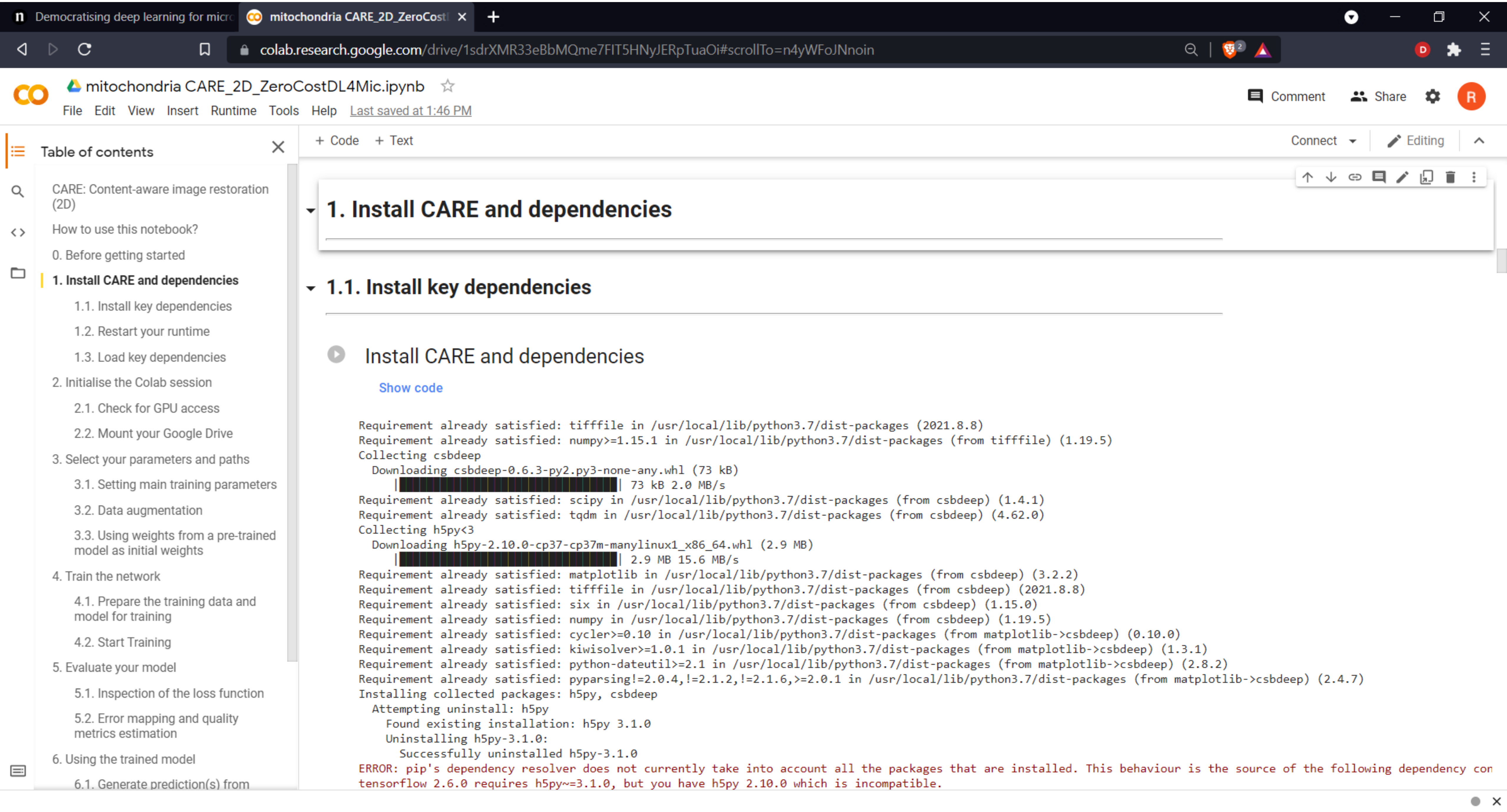
This notebook is based on the following paper:

Content-aware image restoration: pushing the limits of fluorescence microscopy, by Weigert *et al.* published in Nature Methods in 2018 (https://www.nature.com/articles/s41592-018-0216-7)

And source code found in: https://github.com/csbdeep/csbdeep

For a more in-depth description of the features of the network, please refer to this guide provided by the original authors of the work.

We provide a dataset for the training of this notebook as a way to test its functionalities but the training and test data of the


Democratising deep learning for micr
mitochondria CARE_2D_ZeroCost
colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=n4yWFoJNnoin
mitochondria CARE_2D_ZeroCostDL4Mic.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:46 PM
Comment
Share
+ Code + Text
Connect
Editing
Table of contents
Show code

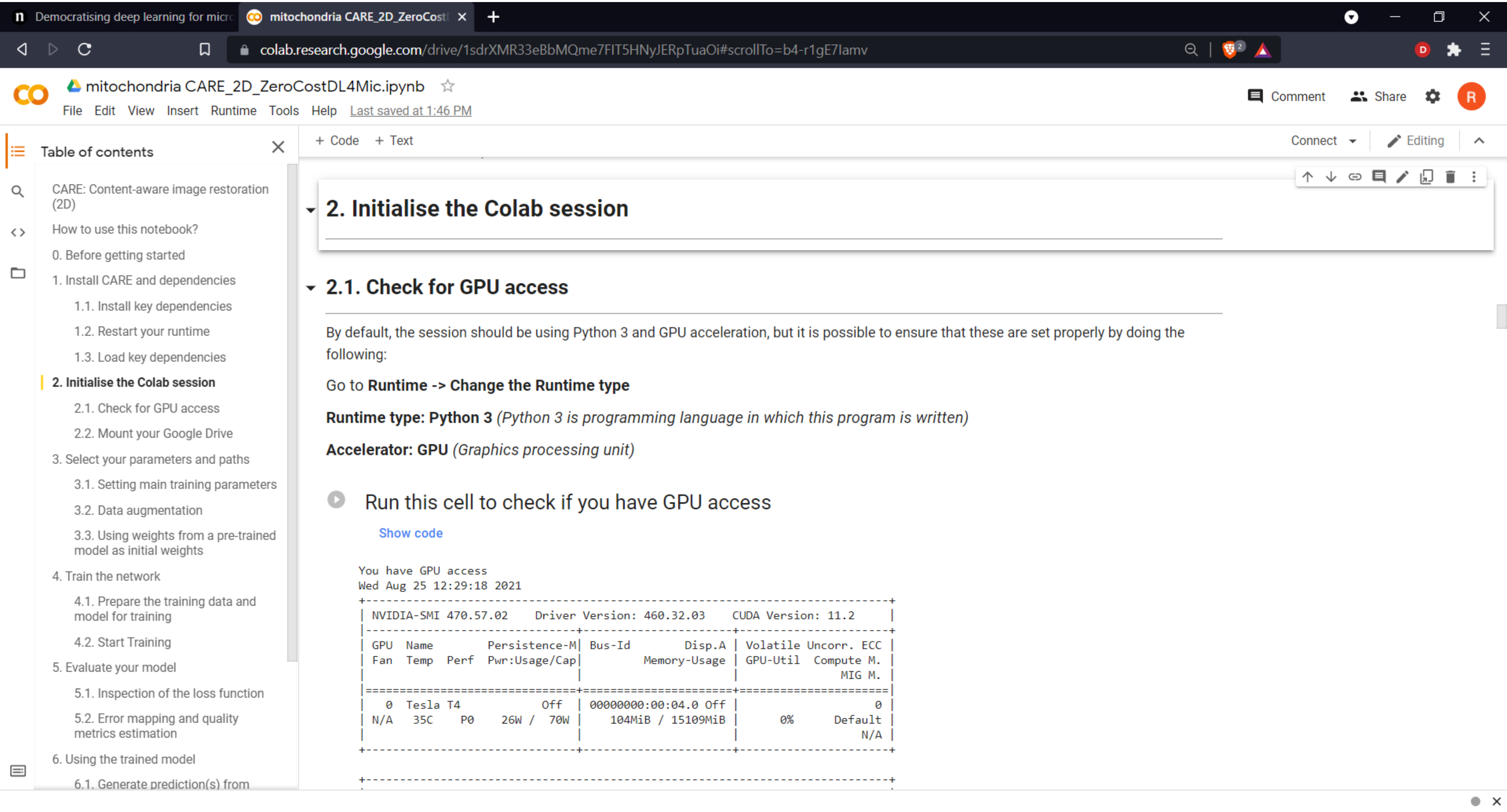
CARE: Content-aware image restoration (2D)
How to use this notebook?
0. Before getting started
1. Install CARE and dependencies
1.1. Install key dependencies
1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from

1. Install CARE and dependencies

1.1. Install key dependencies

Install CARE and dependencies

```
Requirement already satisfied: tifffile in /usr/local/lib/python3.7/dist-packages (2021.8.8)
Requirement already satisfied: numpy>=1.15.1 in /usr/local/lib/python3.7/dist-packages (from tifffile) (1.19.5)
Collecting csbdeep
  Downloading csbdeep-0.6.3-py2.py3-none-any.whl (73 kB)
     |████████████████████████████████| 73 kB 2.0 MB/s
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from csbdeep) (1.4.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from csbdeep) (4.62.0)
Collecting h5py<3
  Downloading h5py-2.10.0-cp37-cp37m-manylinux1_x86_64.whl (2.9 MB)
     |████████████████████████████████| 2.9 MB 15.6 MB/s
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from csbdeep) (3.2.2)
Requirement already satisfied: tifffile in /usr/local/lib/python3.7/dist-packages (from csbdeep) (2021.8.8)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from csbdeep) (1.15.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from csbdeep) (1.19.5)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib->csbdeep) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->csbdeep) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->csbdeep) (2.8.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->csbdeep) (2.4.7)
Installing collected packages: h5py, csbdeep
  Attempting uninstall: h5py
    Found existing installation: h5py 3.1.0
    Uninstalling h5py-3.1.0:
      Successfully uninstalled h5py-3.1.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency con
tensorflow 2.6.0 requires h5py~=3.1.0, but you have h5py 2.10.0 which is incompatible.
```



Democratising deep learning for micro

mitochondria CARE_2D_ZeroCost

colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=b4-r1gE7lamv

mitochondria CARE_2D_ZeroCostDL4Mic.ipynb

File Edit View Insert Runtime Tools Help Last saved at 1:46 PM

Comment Share

+ Code + Text

Connect Editing

Table of contents

CARE: Content-aware image restoration (2D)
How to use this notebook?
0. Before getting started
1. Install CARE and dependencies
1.1. Install key dependencies
1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from

2. Initialise the Colab session

2.1. Check for GPU access

By default, the session should be using Python 3 and GPU acceleration, but it is possible to ensure that these are set properly by doing the following:

Go to **Runtime -> Change the Runtime type**

Runtime type: Python 3 *(Python 3 is programming language in which this program is written)*

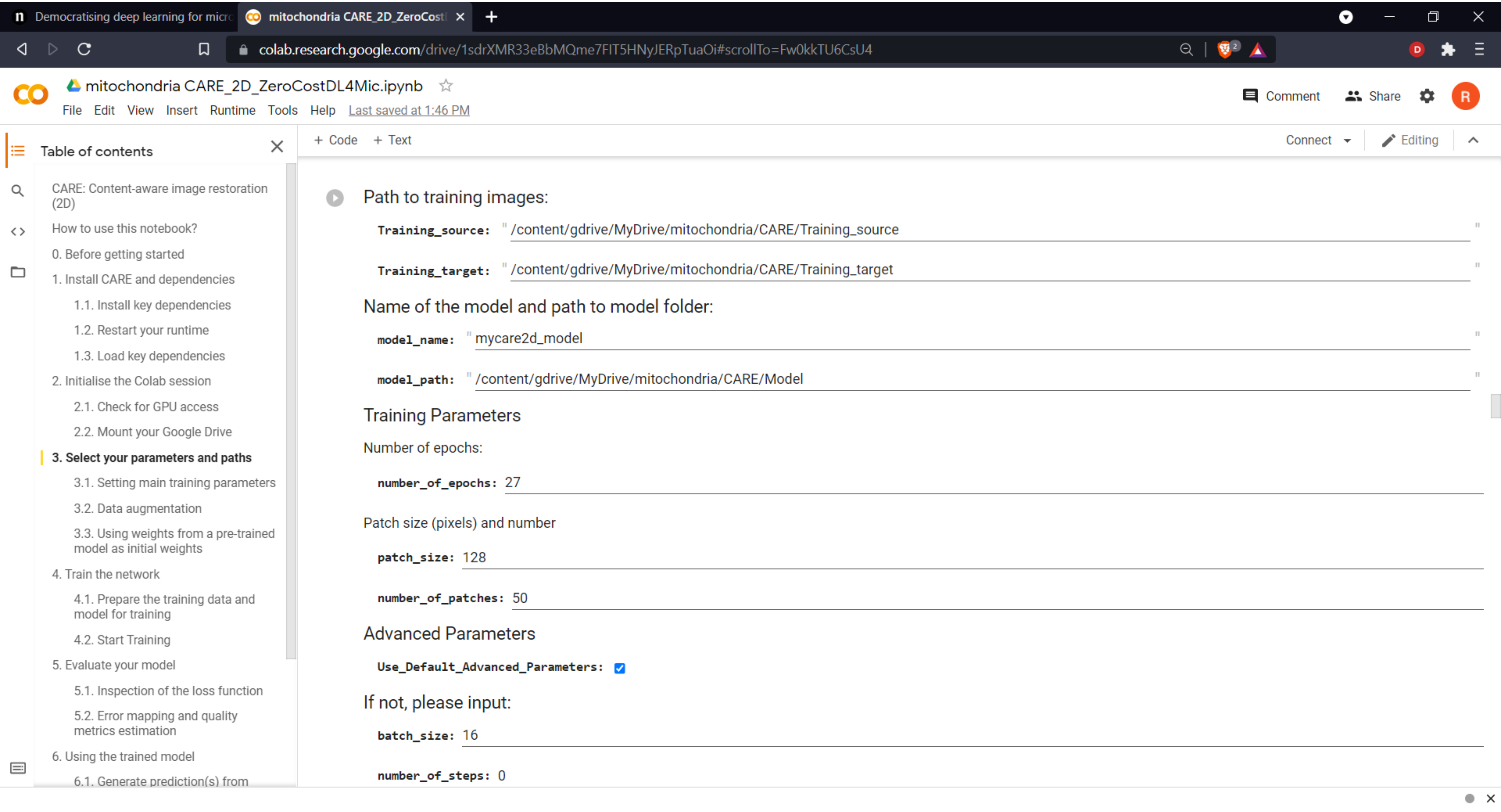
Accelerator: GPU *(Graphics processing unit)*

Run this cell to check if you have GPU access

Show code

```
You have GPU access
Wed Aug 25 12:29:18 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.57.02    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P0    26W /  70W |    104MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
```


Democratising deep learning for micro
mitochondria CARE_2D_ZeroCost
colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=Fw0kkTU6CsU4
mitochondria CARE_2D_ZeroCostDL4Mic.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:46 PM
Comment
Share
+ Code + Text
Connect
Editing
Table of contents

CARE: Content-aware image restoration (2D)
How to use this notebook?
0. Before getting started
1. Install CARE and dependencies
1.1. Install key dependencies
1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from

Path to training images:

`Training_source:` /content/gdrive/MyDrive/mitochondria/CARE/Training_source

`Training_target:` /content/gdrive/MyDrive/mitochondria/CARE/Training_target

Name of the model and path to model folder:

`model_name:` mycare2d_model

`model_path:` /content/gdrive/MyDrive/mitochondria/CARE/Model

Training Parameters

Number of epochs:

`number_of_epochs:` 27

Patch size (pixels) and number

`patch_size:` 128

`number_of_patches:` 50

Advanced Parameters

`Use_Default_Advanced_Parameters:` ✓

If not, please input:

`batch_size:` 16

`number_of_steps:` 0

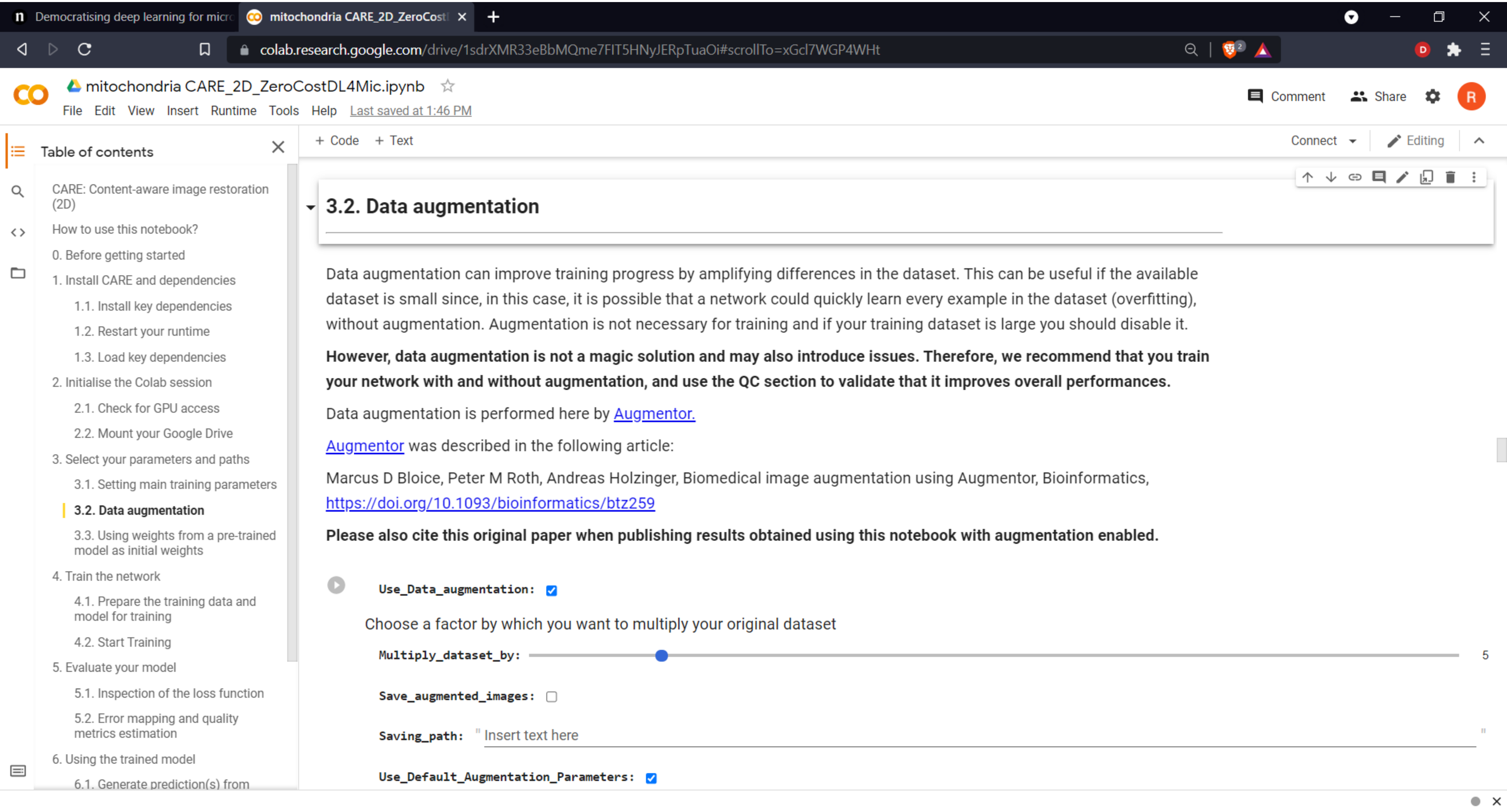

Democratising deep learning for micro
mitochondria CARE_2D_ZeroCost
colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=xGcl7WGP4WHt
mitochondria CARE_2D_ZeroCostDL4Mic.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:46 PM
Comment
Share
+ Code + Text
Connect
Editing

Table of contents

CARE: Content-aware image restoration (2D)
How to use this notebook?
0. Before getting started
1. Install CARE and dependencies
1.1. Install key dependencies
1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from

3.2. Data augmentation

Data augmentation can improve training progress by amplifying differences in the dataset. This can be useful if the available dataset is small since, in this case, it is possible that a network could quickly learn every example in the dataset (overfitting), without augmentation. Augmentation is not necessary for training and if your training dataset is large you should disable it.

However, data augmentation is not a magic solution and may also introduce issues. Therefore, we recommend that you train your network with and without augmentation, and use the QC section to validate that it improves overall performances.

Data augmentation is performed here by Augmentor.

Augmentor was described in the following article:

Marcus D Bloice, Peter M Roth, Andreas Holzinger, Biomedical image augmentation using Augmentor, Bioinformatics, https://doi.org/10.1093/bioinformatics/btz259

Please also cite this original paper when publishing results obtained using this notebook with augmentation enabled.

Use_Data_augmentation: ✓

Choose a factor by which you want to multiply your original dataset

Multiply_dataset_by: 5

Save_augmented_images: ☐

Saving_path: " Insert text here "

Use_Default_Augmentation_Parameters: ✓

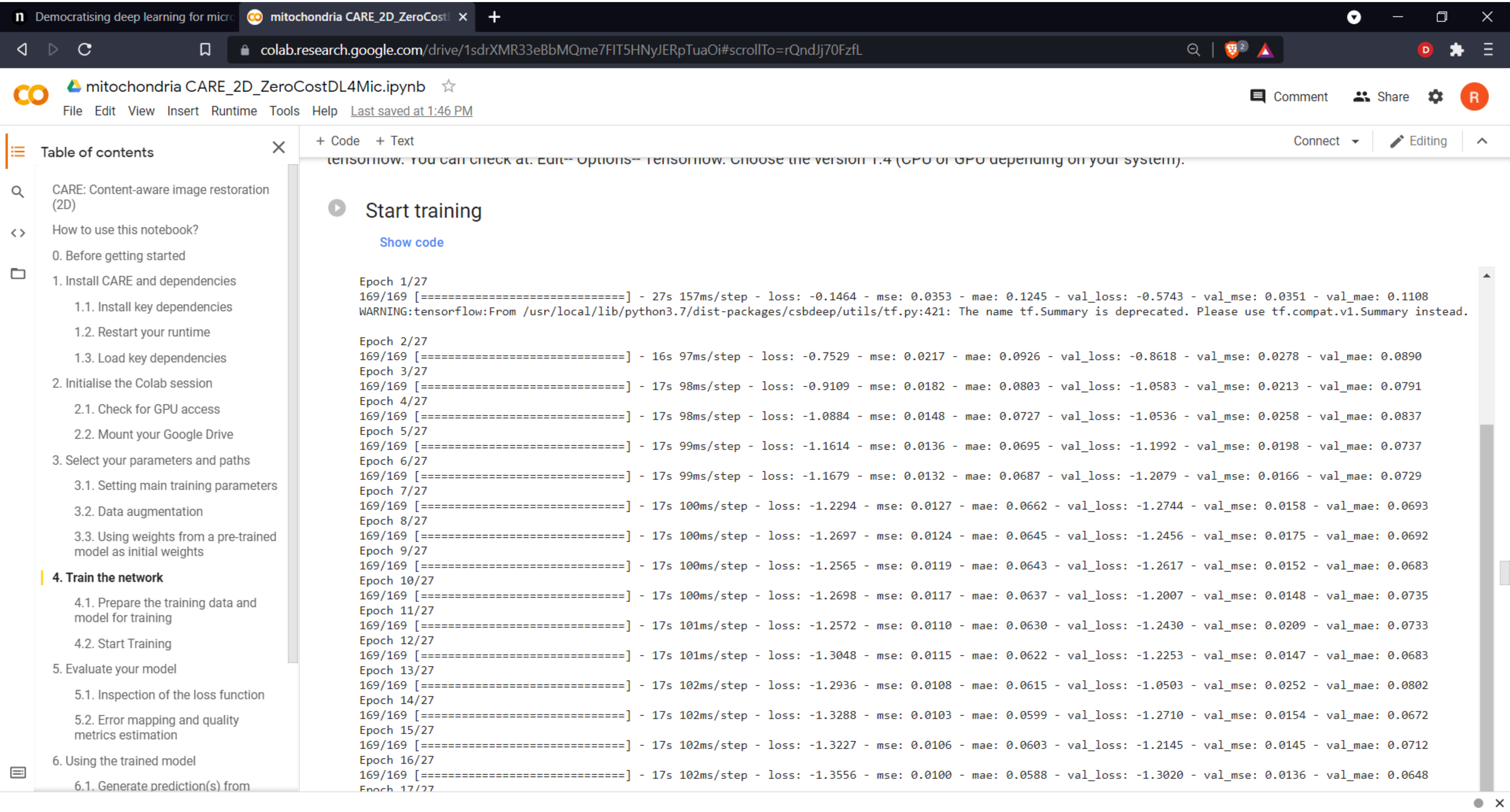

Democratising deep learning for micr
mitochondria CARE_2D_ZeroCost
colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=rQndJj70FzfL
mitochondria CARE_2D_ZeroCostDL4Mic.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:46 PM
Comment
Share
+ Code + Text
Connect
Editing
Show code

Table of contents

CARE: Content-aware image restoration (2D)
How to use this notebook?
0. Before getting started
1. Install CARE and dependencies
1.1. Install key dependencies
1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from

tensorflow. You can check at: Edit-- Options-- Tensorflow. Choose the version 1.4 (CPU or GPU depending on your system).

Start training

```
Epoch 1/27
169/169 [==============================] - 27s 157ms/step - loss: -0.1464 - mse: 0.0353 - mae: 0.1245 - val_loss: -0.5743 - val_mse: 0.0351 - val_mae: 0.1108
WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/csbdeep/utils/tf.py:421: The name tf.Summary is deprecated. Please use tf.compat.v1.Summary instead.

Epoch 2/27
169/169 [==============================] - 16s 97ms/step - loss: -0.7529 - mse: 0.0217 - mae: 0.0926 - val_loss: -0.8618 - val_mse: 0.0278 - val_mae: 0.0890
Epoch 3/27
169/169 [==============================] - 17s 98ms/step - loss: -0.9109 - mse: 0.0182 - mae: 0.0803 - val_loss: -1.0583 - val_mse: 0.0213 - val_mae: 0.0791
Epoch 4/27
169/169 [==============================] - 17s 98ms/step - loss: -1.0884 - mse: 0.0148 - mae: 0.0727 - val_loss: -1.0536 - val_mse: 0.0258 - val_mae: 0.0837
Epoch 5/27
169/169 [==============================] - 17s 99ms/step - loss: -1.1614 - mse: 0.0136 - mae: 0.0695 - val_loss: -1.1992 - val_mse: 0.0198 - val_mae: 0.0737
Epoch 6/27
169/169 [==============================] - 17s 99ms/step - loss: -1.1679 - mse: 0.0132 - mae: 0.0687 - val_loss: -1.2079 - val_mse: 0.0166 - val_mae: 0.0729
Epoch 7/27
169/169 [==============================] - 17s 100ms/step - loss: -1.2294 - mse: 0.0127 - mae: 0.0662 - val_loss: -1.2744 - val_mse: 0.0158 - val_mae: 0.0693
Epoch 8/27
169/169 [==============================] - 17s 100ms/step - loss: -1.2697 - mse: 0.0124 - mae: 0.0645 - val_loss: -1.2456 - val_mse: 0.0175 - val_mae: 0.0692
Epoch 9/27
169/169 [==============================] - 17s 100ms/step - loss: -1.2565 - mse: 0.0119 - mae: 0.0643 - val_loss: -1.2617 - val_mse: 0.0152 - val_mae: 0.0683
Epoch 10/27
169/169 [==============================] - 17s 100ms/step - loss: -1.2698 - mse: 0.0117 - mae: 0.0637 - val_loss: -1.2007 - val_mse: 0.0148 - val_mae: 0.0735
Epoch 11/27
169/169 [==============================] - 17s 101ms/step - loss: -1.2572 - mse: 0.0110 - mae: 0.0630 - val_loss: -1.2430 - val_mse: 0.0209 - val_mae: 0.0733
Epoch 12/27
169/169 [==============================] - 17s 101ms/step - loss: -1.3048 - mse: 0.0115 - mae: 0.0622 - val_loss: -1.2253 - val_mse: 0.0147 - val_mae: 0.0683
Epoch 13/27
169/169 [==============================] - 17s 102ms/step - loss: -1.2936 - mse: 0.0108 - mae: 0.0615 - val_loss: -1.0503 - val_mse: 0.0252 - val_mae: 0.0802
Epoch 14/27
169/169 [==============================] - 17s 102ms/step - loss: -1.3288 - mse: 0.0103 - mae: 0.0599 - val_loss: -1.2710 - val_mse: 0.0154 - val_mae: 0.0672
Epoch 15/27
169/169 [==============================] - 17s 102ms/step - loss: -1.3227 - mse: 0.0106 - mae: 0.0603 - val_loss: -1.2145 - val_mse: 0.0145 - val_mae: 0.0712
Epoch 16/27
169/169 [==============================] - 17s 102ms/step - loss: -1.3556 - mse: 0.0100 - mae: 0.0588 - val_loss: -1.3020 - val_mse: 0.0136 - val_mae: 0.0648
Epoch 17/27
```

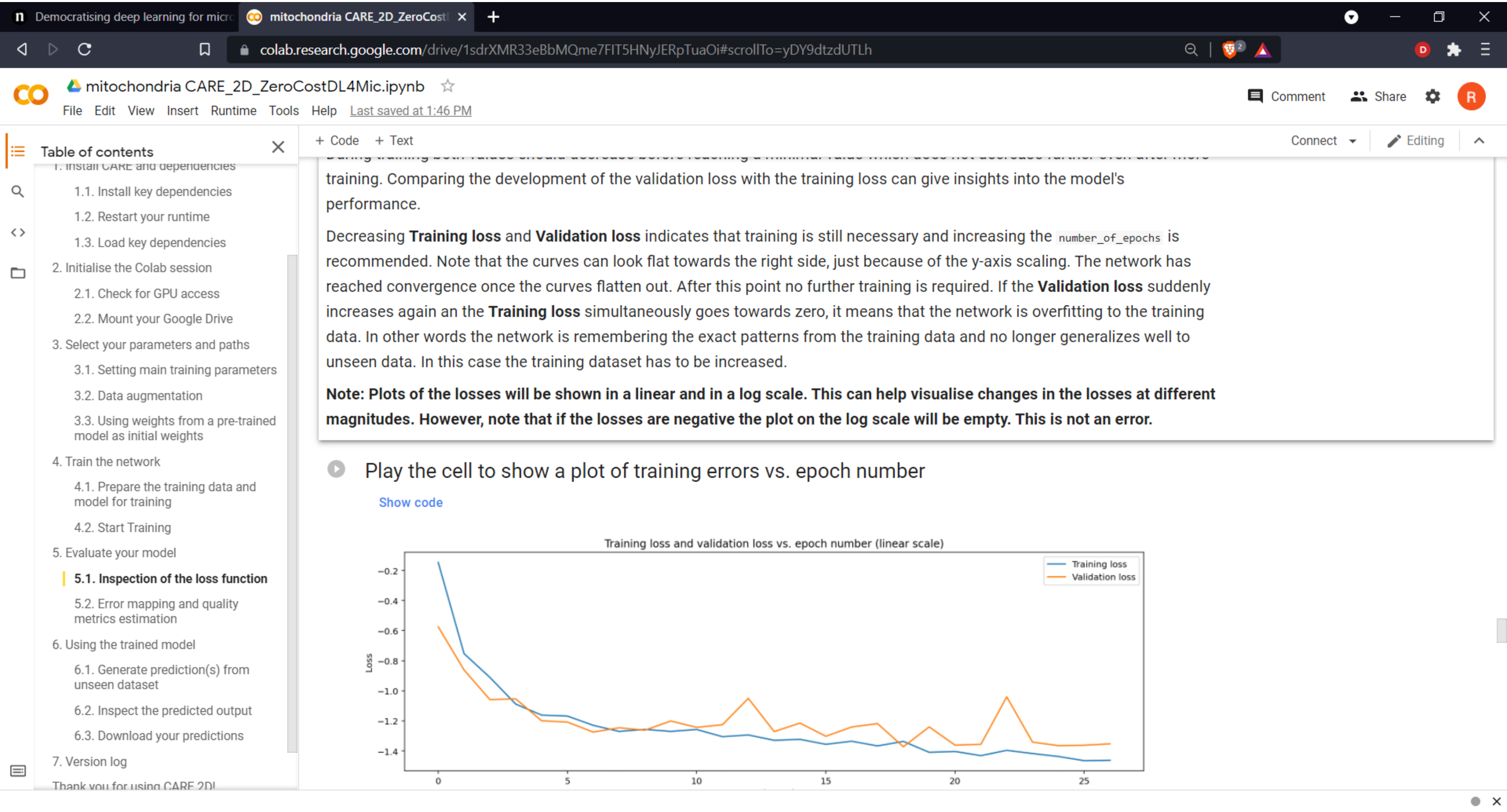

Democratising deep learning for micro
mitochondria CARE_2D_ZeroCost
colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=yDY9dtzdUTLh
mitochondria CARE_2D_ZeroCostDL4Mic.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:46 PM
Comment
Share
+ Code + Text
Connect
Editing

Table of contents

1. Install CARE and dependencies
1.1. Install key dependencies
1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from unseen dataset
6.2. Inspect the predicted output
6.3. Download your predictions
7. Version log
Thank you for using CARE 2D!

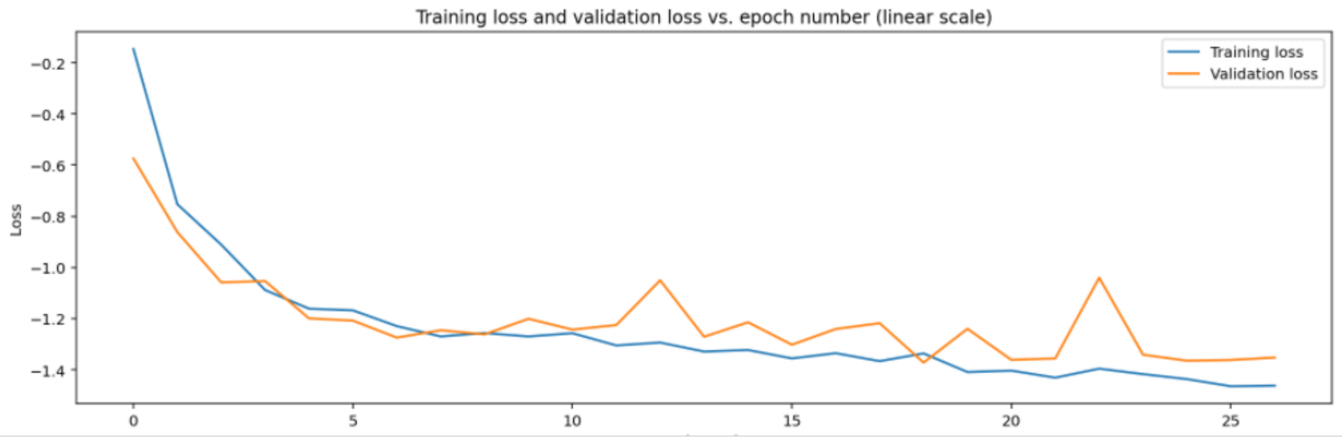
During training both values should decrease before reaching a minimal value which does not decrease further even after more training. Comparing the development of the validation loss with the training loss can give insights into the model's performance.

Decreasing **Training loss** and **Validation loss** indicates that training is still necessary and increasing the `number_of_epochs` is recommended. Note that the curves can look flat towards the right side, just because of the y-axis scaling. The network has reached convergence once the curves flatten out. After this point no further training is required. If the **Validation loss** suddenly increases again an the **Training loss** simultaneously goes towards zero, it means that the network is overfitting to the training data. In other words the network is remembering the exact patterns from the training data and no longer generalizes well to unseen data. In this case the training dataset has to be increased.

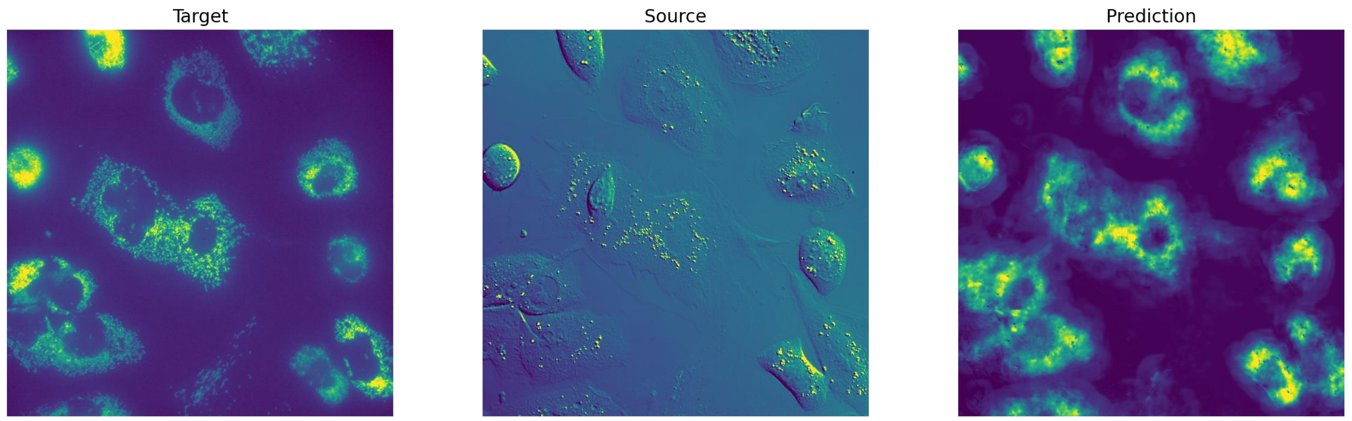
Note: Plots of the losses will be shown in a linear and in a log scale. This can help visualise changes in the losses at different magnitudes. However, note that if the losses are negative the plot on the log scale will be empty. This is not an error.

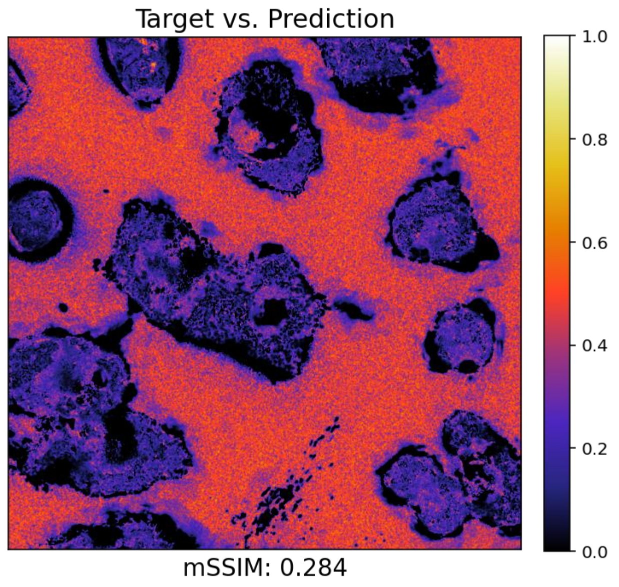
Play the cell to show a plot of training errors vs. epoch number

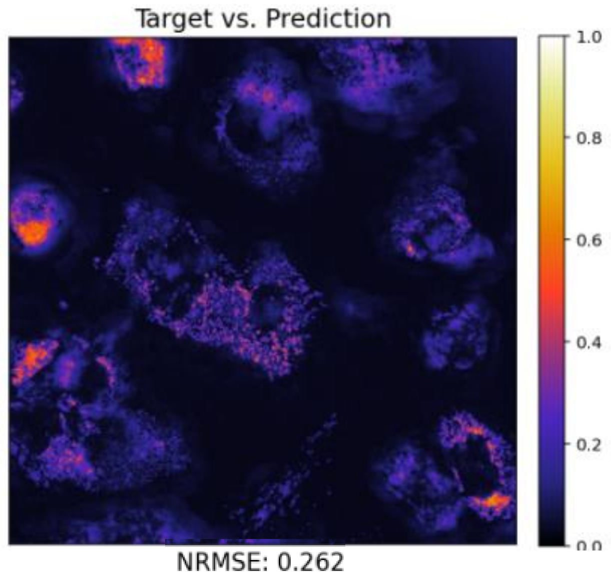
Show code



CARE metrics and prediction







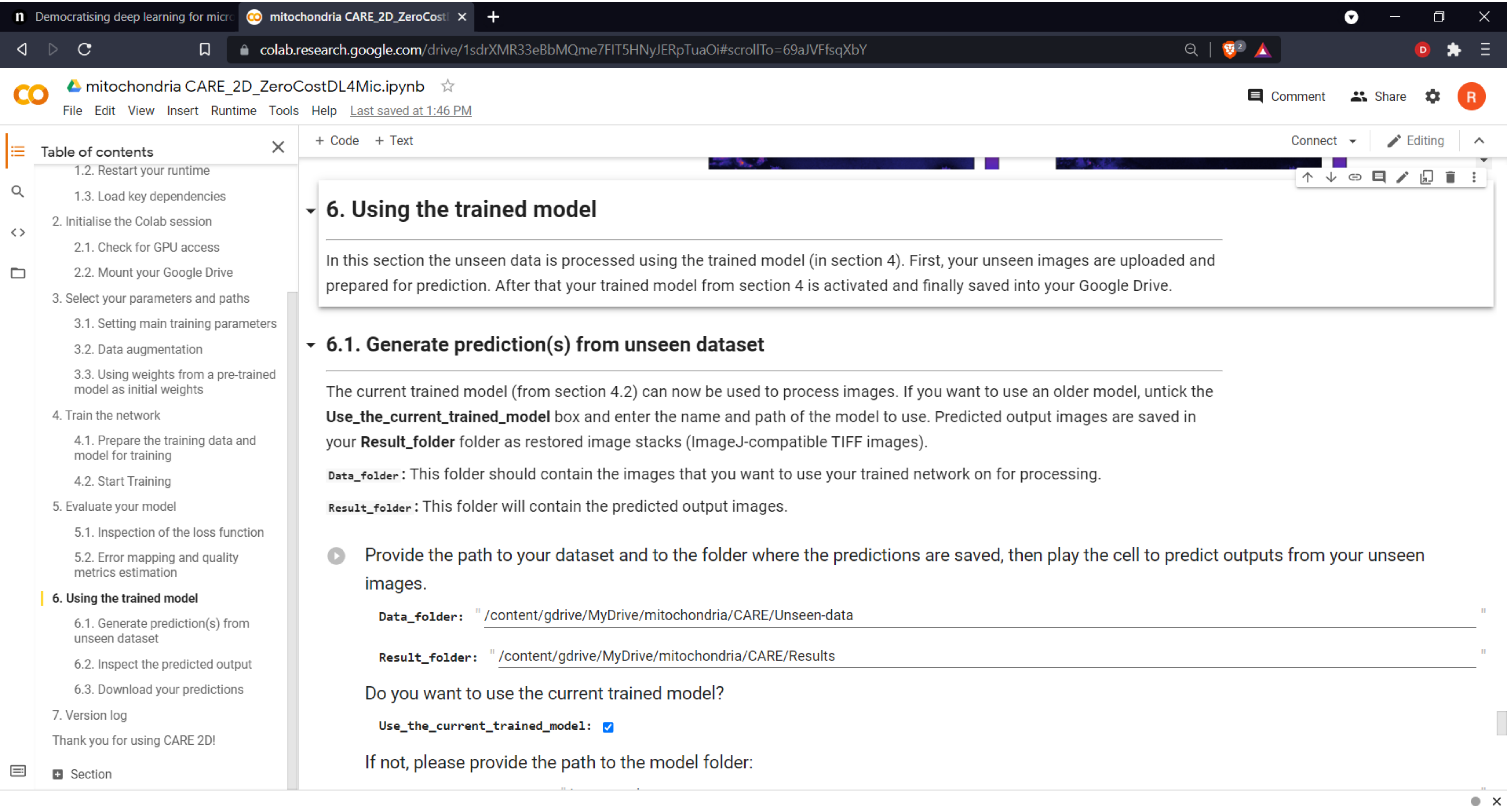

Democratising deep learning for micro
mitochondria CARE_2D_ZeroCostD
colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=69aJVFfsqXbY
mitochondria CARE_2D_ZeroCostDL4Mic.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:46 PM
Comment Share
+ Code + Text
Connect Editing

Table of contents
1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from unseen dataset
6.2. Inspect the predicted output
6.3. Download your predictions
7. Version log
Thank you for using CARE 2D!
Section

6. Using the trained model

In this section the unseen data is processed using the trained model (in section 4). First, your unseen images are uploaded and prepared for prediction. After that your trained model from section 4 is activated and finally saved into your Google Drive.

6.1. Generate prediction(s) from unseen dataset

The current trained model (from section 4.2) can now be used to process images. If you want to use an older model, untick the **Use_the_current_trained_model** box and enter the name and path of the model to use. Predicted output images are saved in your **Result_folder** folder as restored image stacks (ImageJ-compatible TIFF images).

`Data_folder`: This folder should contain the images that you want to use your trained network on for processing.

`Result_folder`: This folder will contain the predicted output images.

Provide the path to your dataset and to the folder where the predictions are saved, then play the cell to predict outputs from your unseen images.

`Data_folder:` " /content/gdrive/MyDrive/mitochondria/CARE/Unseen-data "

`Result_folder:` " /content/gdrive/MyDrive/mitochondria/CARE/Results "

Do you want to use the current trained model?

`Use_the_current_trained_model:` ✓

If not, please provide the path to the model folder:

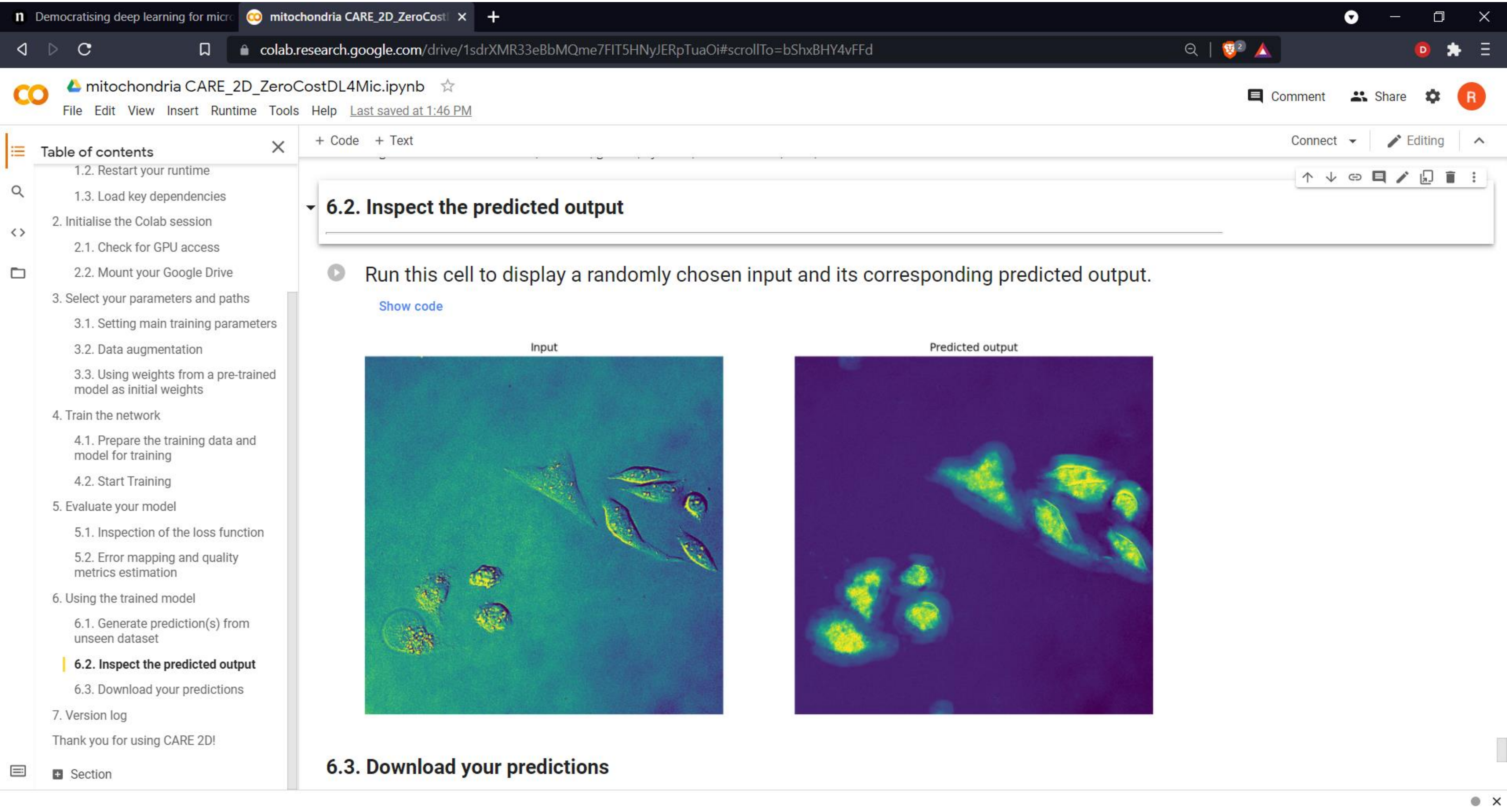

Democratising deep learning for micro
mitochondria CARE_2D_ZeroCost
colab.research.google.com/drive/1sdrXMR33eBbMQme7FIT5HNyJERpTuaOi#scrollTo=bShxBHY4vFFd
mitochondria CARE_2D_ZeroCostDL4Mic.ipynb
File Edit View Insert Runtime Tools Help Last saved at 1:46 PM
Comment Share
+ Code + Text
Connect Editing

Table of contents

1.2. Restart your runtime
1.3. Load key dependencies
2. Initialise the Colab session
2.1. Check for GPU access
2.2. Mount your Google Drive
3. Select your parameters and paths
3.1. Setting main training parameters
3.2. Data augmentation
3.3. Using weights from a pre-trained model as initial weights
4. Train the network
4.1. Prepare the training data and model for training
4.2. Start Training
5. Evaluate your model
5.1. Inspection of the loss function
5.2. Error mapping and quality metrics estimation
6. Using the trained model
6.1. Generate prediction(s) from unseen dataset
6.2. Inspect the predicted output
6.3. Download your predictions
7. Version log
Thank you for using CARE 2D!

Section

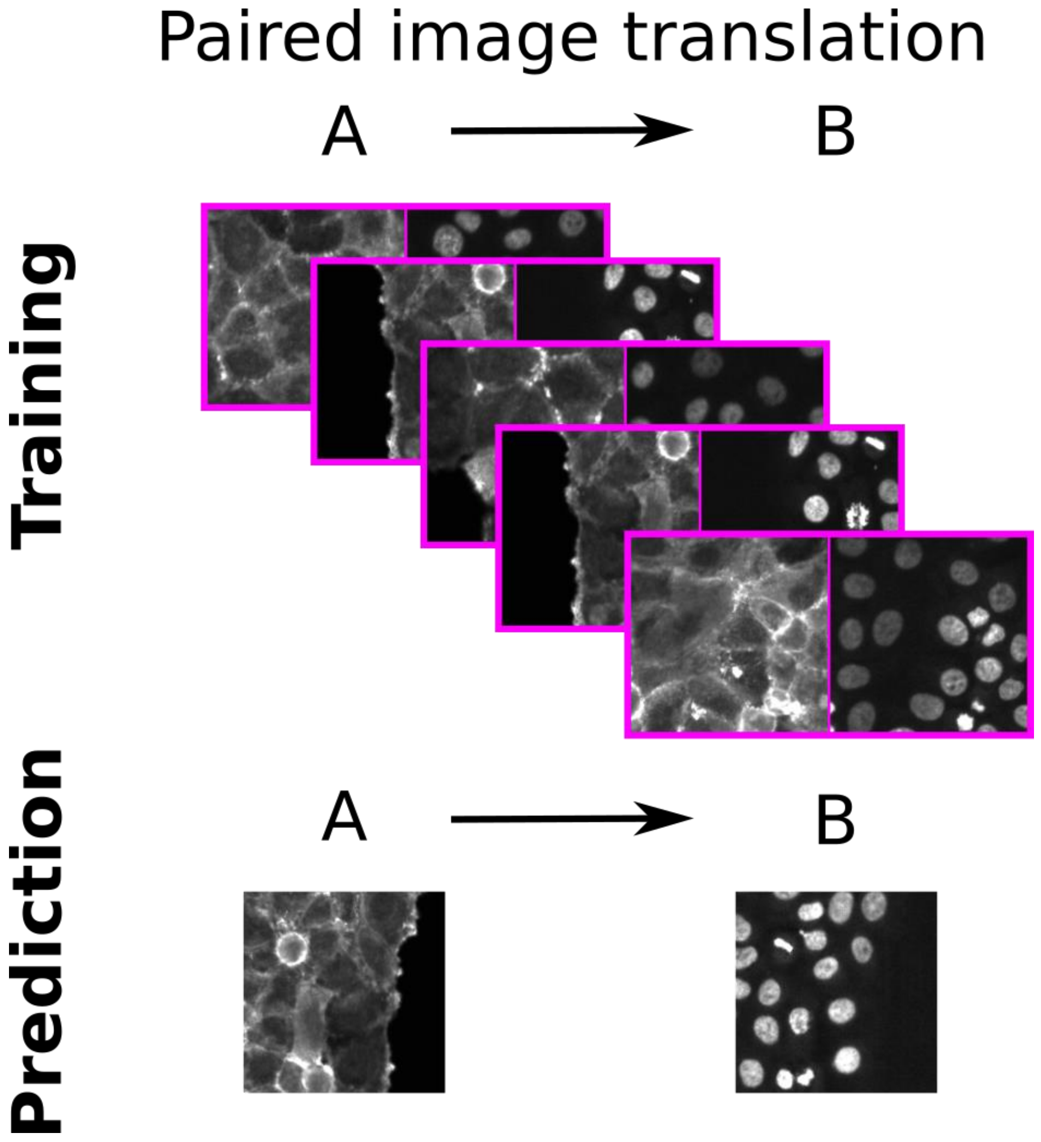
6.2. Inspect the predicted output

Run this cell to display a randomly chosen input and its corresponding predicted output.

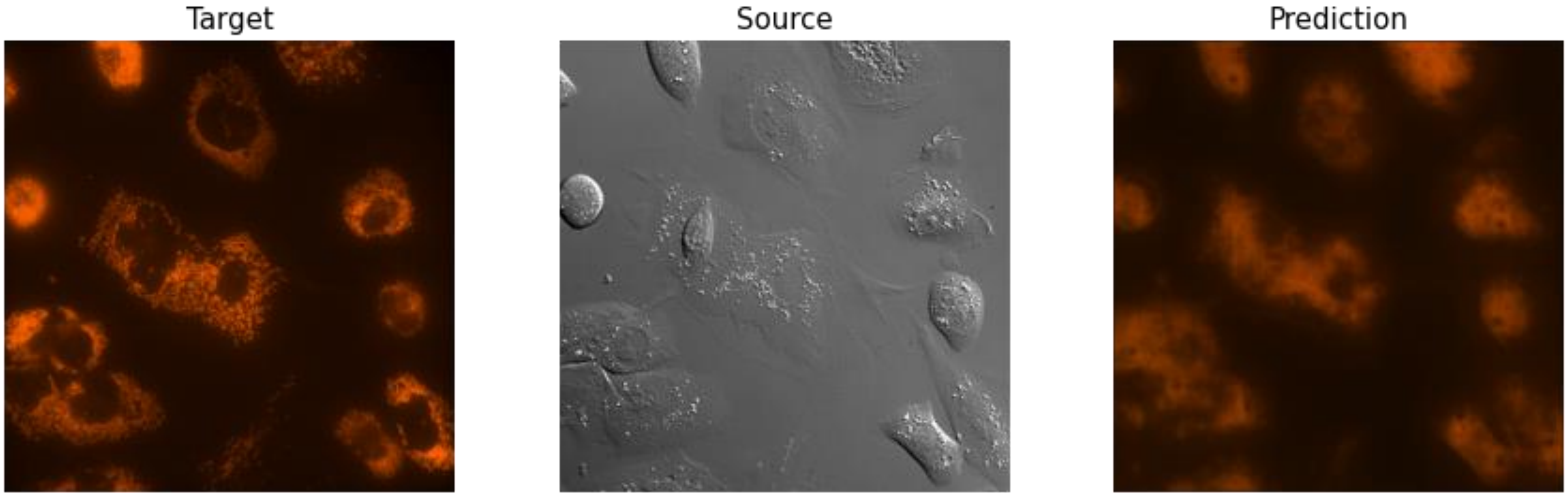
Show code

6.3. Download your predictions

Pix2pix

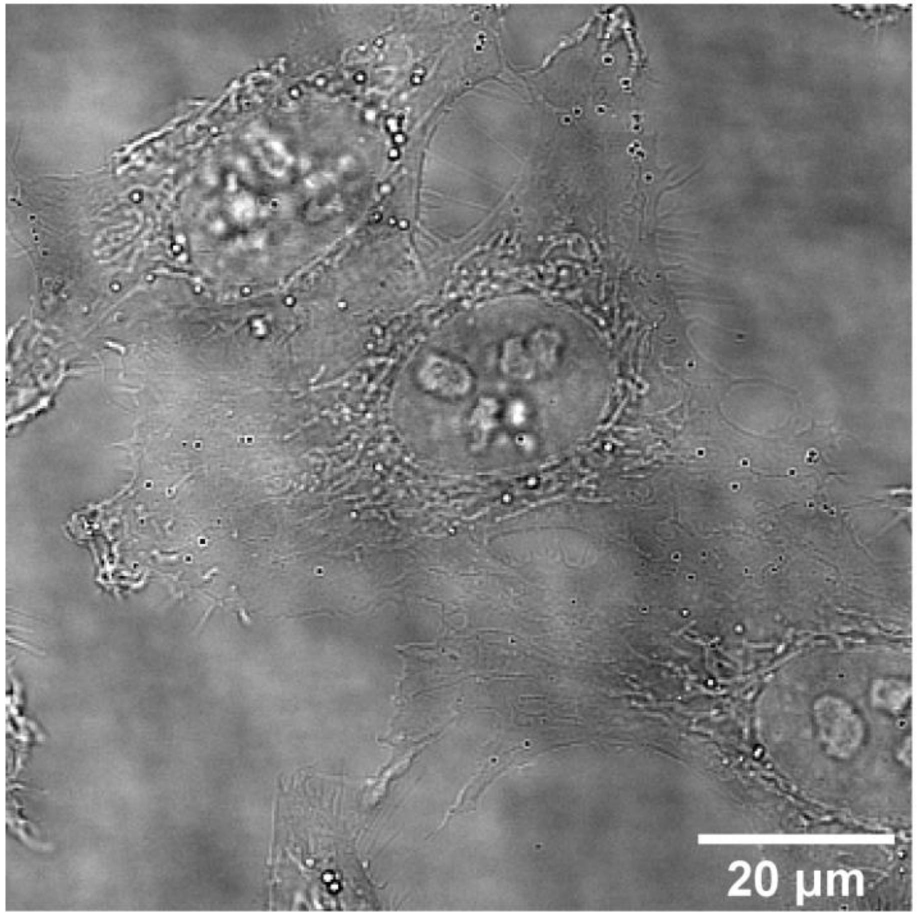


Pix2pix prediction

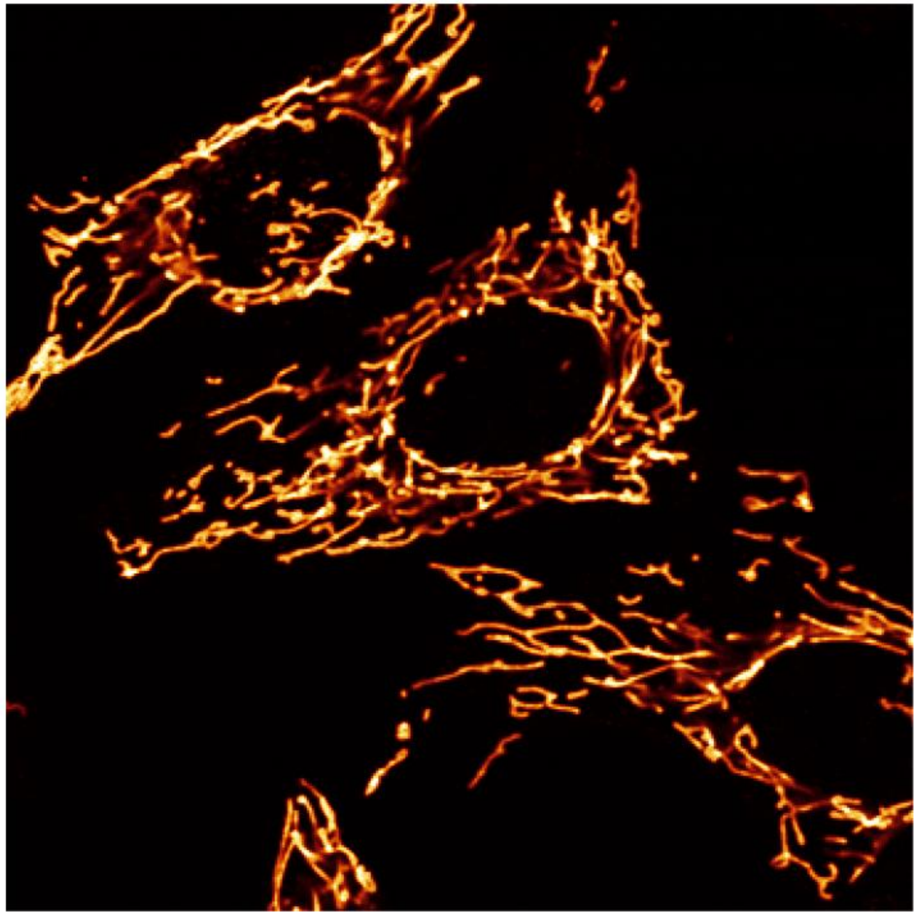


Fnet

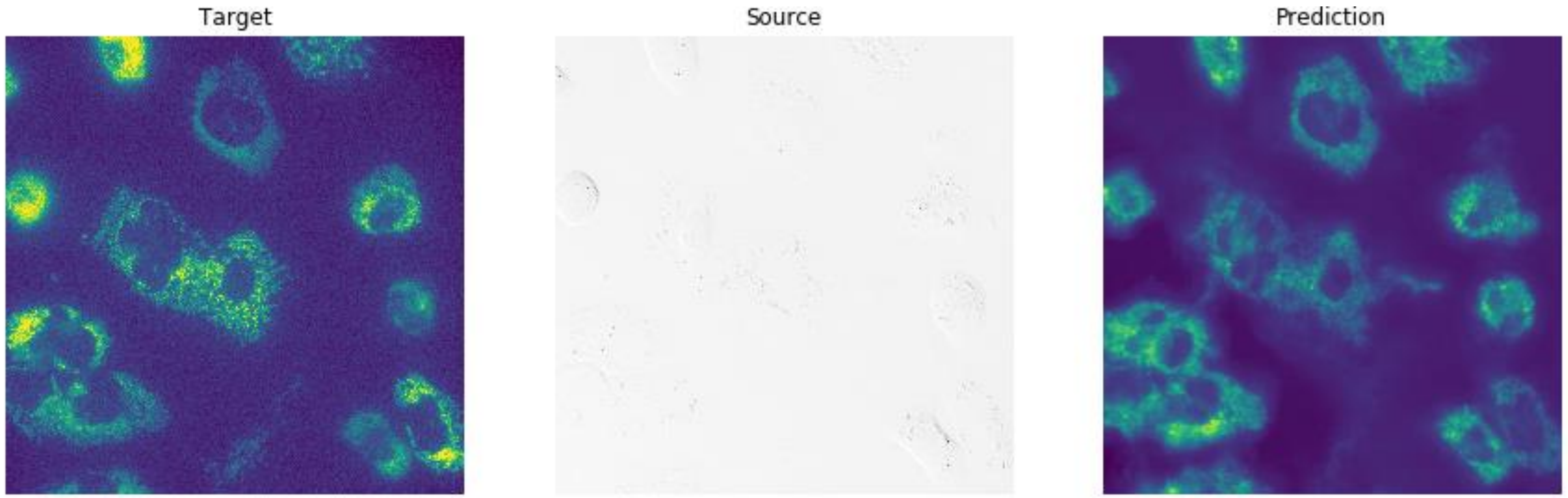
Transmitted light image



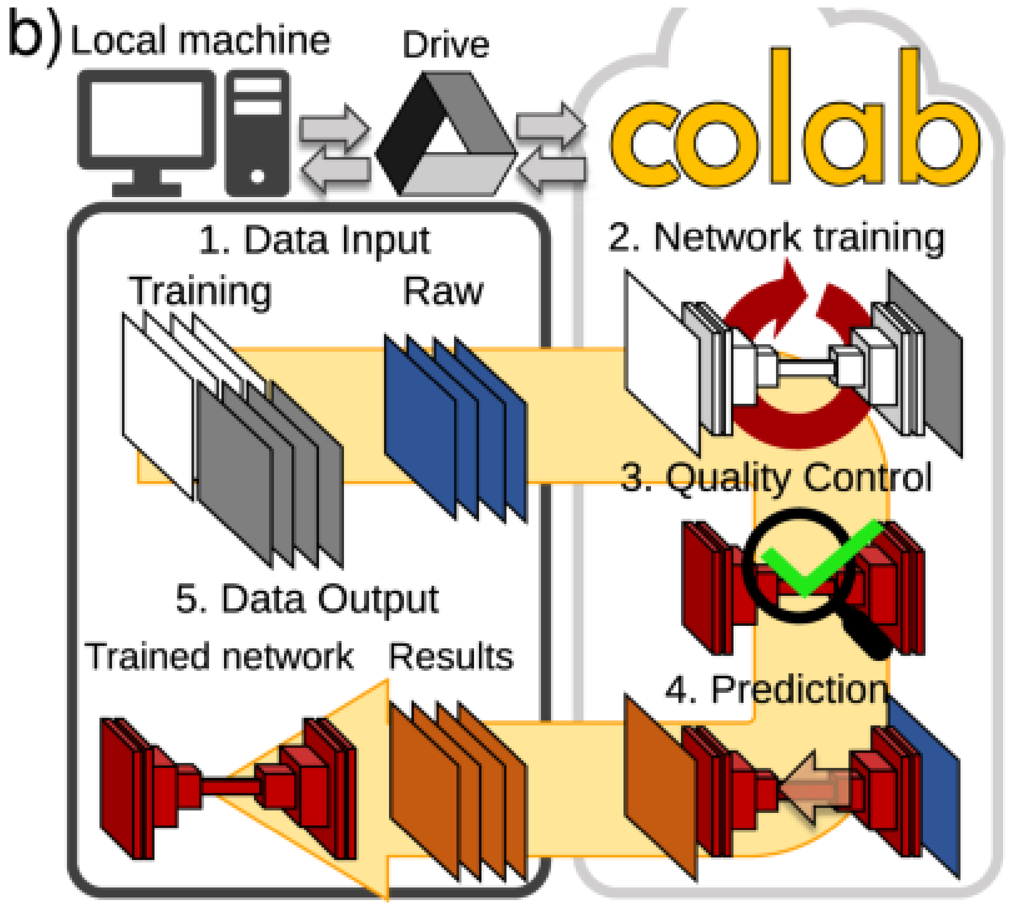
Fluorescence image



Fnet prediction

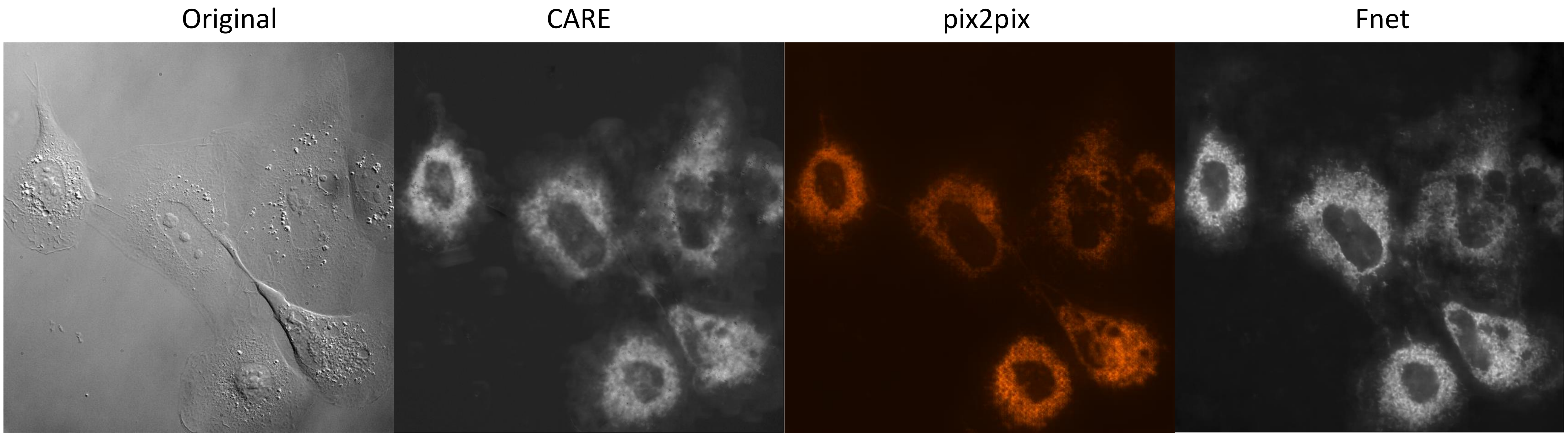


ZeroCostDL4Mic

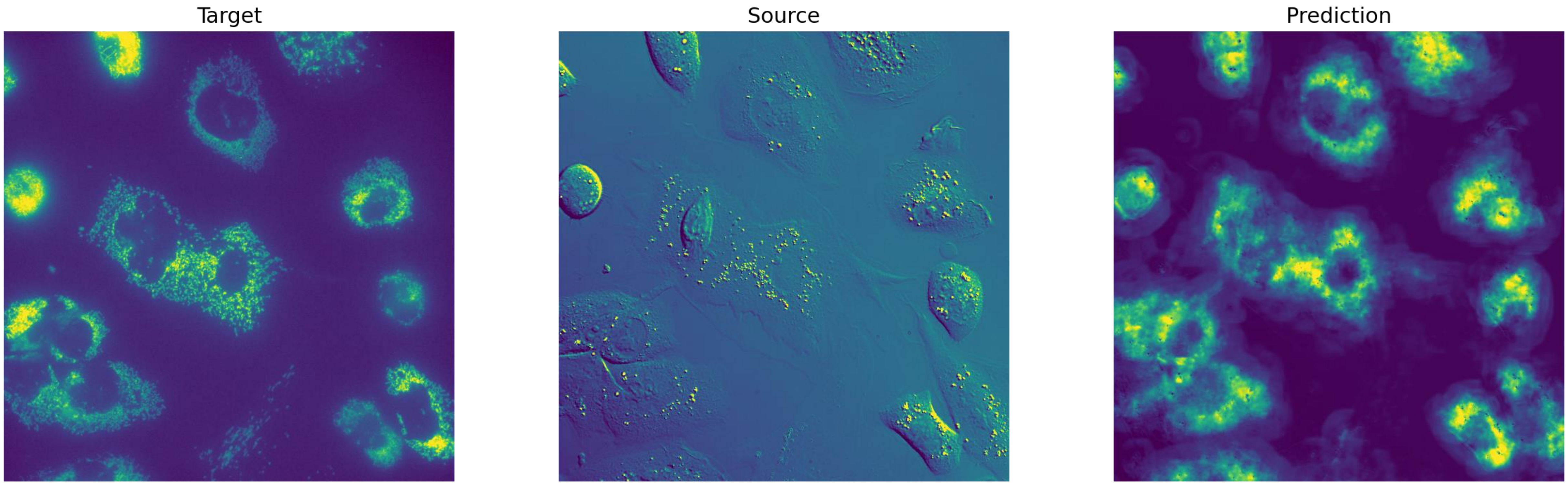


von Chamier & Laine et al., 2020

Comparison



CARE results



Acknowledgements

- Optical cell biology lab
  - Ricardo Henriques
  - Mario Del Rosario
  - Hannah S. Heil
  - Afonso Mendes

GULBENKIAN SCIENCE