Instructions on using unet\_train\_0\_004\_KH.ipynb and unet\_train\_0\_0009\_KH.ipynb files

* Both files were run in Google Colabtoratory to train CNN models using the fastAI library and both models are trained with the real component of the wavefield.
* For both models, there are 503 images for the real component of the wavefield and the corresponding 503 labels. Thus, the original dataset size is 503.
* The folder names and filenames are bolded in the following instructions.

**\*All the steps to train the model are commented in both notebooks as well as written below.**

\* Separate real, imaginary, and magnitude components of the wavefield before starting step 1.

**Procedure:**

1. Directory setup:

* Create 7 folders beforehand:

The folder names used in both notebooks are shown below. The folders were created in Google Drive for both notebooks.

* **images\_real**: store original images (real component)
* **labels\_real**: store original labels (real component)
* **images**: store original, augmented, and noisy images (one noise level)
* **labels**: store original, augmented, and noisy labels
* **test\_images\_real**: store independent test dataset (real component)
* **models**: store trained models
* **noise**: store noisy images with different noise levels for noise case study

(In both notebooks, **exp** folder was created for experimental data and **test** folder was created for additional small width defects testset. These two folders are optional and can be combined with the **test\_images\_real** folder.)

* Save **code.txt** (shown in the src folder in Dropbox) in the same directory as the 7 folders. **code.txt** has all the classes listed in order line by line. (i.e. 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)

2. Color scheme for CNN prediction:

* Follow instructions in **consistent\_colors.txt** and upload the **interpret.py** file to set up the color scheme for CNN predictions.
* Click on the folder icon on the left bar to show the file systems in Google Colabtoratory.
* The directory path to the original **interpret.py** file fastAI installed is ‘../usr/local/lib/python3.6/dist-packages/fastai/vision/interpret.py’.
* Delete the original **interpret.py** file fastAI installed.
* Upload the customized **interpret.py** file.

3. Filenames setup:

The script for changing filenames is combined in both notebooks. For data augmentation, 180 rotation and Gaussian noise are performed on both models.

* Original image filename: **rd8\_9\_60\_real.png**
* Original label filename: **rd8\_9\_60\_mask.png**
* 180 rotated image filename: **rd8\_9\_60\_real\_aug0.png**
* 180 rotated label filename: **rd8\_9\_60\_mask\_aug0.png**
* Gaussian noise image filename: **rd8\_9\_60\_real\_gaussian.png**
* Gaussian noise label filename: **rd8\_9\_60\_mask\_gaussian.png**
* 180 rotated + Gaussian noise image filename: **rd8\_9\_60\_real\_aug0\_gaussian.png**
* 180 rotated + Gaussian noise label filename: **rd8\_9\_60\_mask\_aug0\_gaussian.png**

4. Dataset preparation:

The script and instructions for dataset preparation and data augmentation are combined and commented in both notebooks. For data augmentation, 180 rotation and Gaussian noise are performed on both models.

* **unet\_train\_0\_0009\_KH.ipynb**:
* Import the original 503 images and labels from the **images\_real** and **labels\_real** folders.
* Perform 180 rotation on both original 503 images and labels.
* Save the 180 rotated images and labels with different filenames based on step 3 in the **images** and **labels** folders.
* Apply Gaussian noise with var=0.004 “only” to the original 503 images.
* Save the noisy images and the corresponding original labels with different filenames based on step 3 in the **images** and **labels** folders.
* The total dataset size is 1509.   
  (original 503 + 180 rotated 503 + noisy 503 = total 1509)
* **unet\_train\_0\_004\_KH.ipynb**:
* Import the original 503 images and labels from the **images\_real** and **labels\_real** folders.
* Perform 180 rotation on both original 503 images and labels.
* Save the 180 rotated images and labels with different filenames based on step 3 in the **images** and **labels** folders.
* Apply Gaussian noise with var=0.0009 “only” on the original 503 images and the 180 rotated 503 images.
* Save the noisy images and the corresponding original labels with different filenames based on step 3 in the **images** and **labels** folders.
* The total dataset size is 2012.

(original 503 + 180 rotated 503 + noisy 1006 = total 2012)

5. Training process:

* Parameters to adjust: batch size/learning rate/epochs
* Choose batch size based on available GPU memory:
* **unet\_train\_0\_0009\_KH.ipynb**: batch size = 8
* **unet\_train\_0\_004\_KH.ipynb**: batch size = 16
* Choose the learning rate corresponding to the steepest downward slope based on the plot of learning rate vs loss:
* **unet\_train\_0\_0009\_KH.ipynb**:

1. Stage 1: 3e-03
2. Stage 2: lower bound = 1e-04, upper bound = 6e-04

* **unet\_train\_0\_004\_KH.ipynb**:

1. Stage 1: 5e-04
2. Stage 2: lower bound = 1e-05, upper bound = 1e-04

* Choose the number of epochs based on the accuracy reported by the accuracy metrics:
* **unet\_train\_0\_0009\_KH.ipynb**:

1. Stage 1: epoch = 30
2. Stage 2: epoch = 40

* **unet\_train\_0\_004\_KH.ipynb**:

1. Stage 1: epoch = 30
2. Stage 2: epoch = 40

* Run through the rest of the cells until the model finishes stage 1 training.
* Save stage 1 model as **.pth files** in the **models** folder. (i.e. **stage-1-real-noise0.004.pth**)
* Load stage 1 model and run through the rest of the cells until the model finishes stage 2 training.
* Save stage 2 model as **.pth files** in the **models** folder. (i.e. **stage-2-real-noise0.004.pth**)
* Load stage 2 model and run through the rest of the cells to:
* Plot the confusion matrix on the validation dataset
* Get predictions on the independent test dataset
* Conduct different levels of noise case study
* Save and export the model as **.pkl files** in the **models** folder. (i.e. **real-noise0.004-export.pkl**)

6. Saved models:

The saved models are shown in the trained CNN models folder in Dropbox.

* **unet\_train\_0\_0009\_KH.ipynb:**
* **stage-1-real-noise0.0009.pth**
* **stage-2-real-noise0.0009.pth**
* **real-noise0.0009-export.pkl**
* **unet\_train\_0\_004\_KH.ipynb**:
* **stage-1-real-noise0.004.pth**
* **stage-2-real-noise0.004.pth**
* **real-noise0.004-export.pkl**

\*The whole CNN model setup and training process are completed after step 6. Step 7 is performed when we would like to use the trained model to make more predictions.

6. Reload models for more predictions:

* Run the cells from the beginning and stop after running the cell that creates paths to all the folders as shown in figure 1.  
    
  

Figure 1. Create paths to all folders.

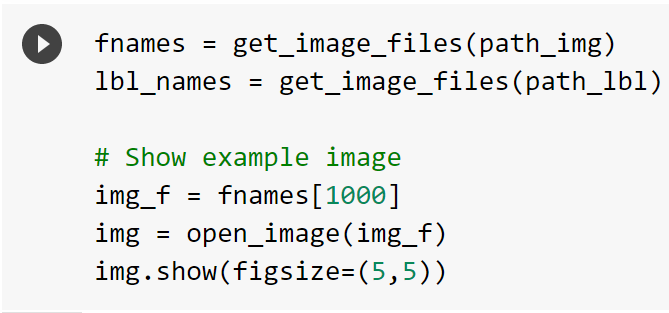
* Run the cell that shows an example image as shown in figure 2.   
    
  

Figure 2. Show an example image.

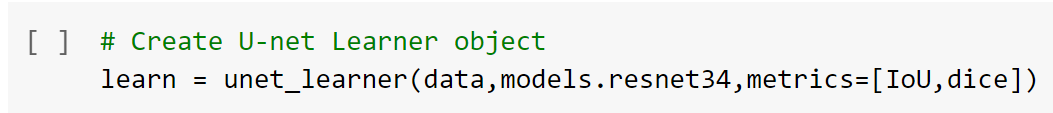
* Run the rest of the cells and stop after running the cell that creates a U-Net learner object as shown in figure 3.   
    
  

Figure 3. Create a U-Net learner object.

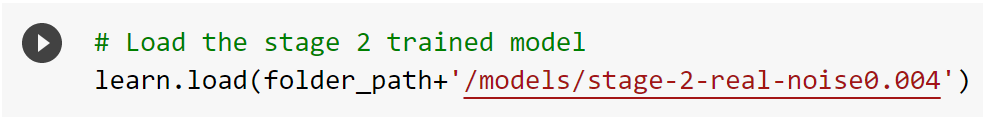
* Run the cell that loads the stage 2 trained model (.pth files) as shown in figure 4.   
    
  

Figure 4. Load the stage 2 trained model.

* Run the rest of the cells to get predictions.