**Project 2 – Code Flow and Meaning**

**1) Synopsis**

At this point we need to rein in all programs written so far, determine how/when they are to be used, what values users may adjust, how data needs to be organized, and in what order users should execute the programs. At this point, many things that feel clear to many come across as vague to others (most specifically me, Jonathan Myers), so this document hopes to serve as an initial step towards finding meaning and purpose in all the files used to generate prediction results. Please read through my understanding for the different applications, their purpose, how they are used, and make comments on these perspectives.

**2) Outline**

This document contains different sections, organized chronologically based on GitHub submissions, each discussing changes that occurred on GitHub. The sections talk about which files were modified, the purpose of those files, and a few notes on what changed. The sections also talk about my understanding of how and when the files are used and how other people may use them in the future.

**3) GitHub History**

As alluded earlier, the subsections here encompass GitHub changes chronologically ordered.

**3.1) added initial version of optical flow & variance - jiahaoxu**

This was the initial submission which included a number of programs:

1. **getData.sh** – This shell script copies all data from gs://uga-dsp/project2, moves into the data folder, then extracts all data from the tar files
2. **OpticalFlow.py** – This python code provides a number of functions (in general following the example at <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html>):
   1. **getIOU**: Calculates a prediction score using trainData zip
   2. **calculateThreshold**: Finds a threshold value in the range [5,130] that provides the best getIOU score
   3. **getHashCodeSet**: Reads all the names from train.txt or test.txt into a set called hashcodeSet
   4. **getTrainData**: The function iterates through all subdirectories of a target data directory, and for each it iterates through every pair of frames – ranging from (0,1) to (98,99). On each pair, it executes calcOpticalFlowFarneback between adjacent frame pairs. The gray results are then evaluated to conclude 2 (cilia) if the gray code >= 128, 1 if the gray code >=32, and 0 otherwise. The results for each pixel are scaled to a 256 range and the results are returned in an array of pairs with the predicted value and the supplied mask value
   5. **generateTestResult**: This function performs the same operations at getTrainData, except here there are no mask values provided for the test set and this function does not return results, rather this function generates png mask files using the values generates by the calcOpticalFlowFarneback and cartToPolar functions. This function takes in a threshold value to determine when it should mark a pixel as cilia or not (see earlier functions for derivation of threshold)
   6. **main**: This function looks to the global variables to determine which actions are appropriate. For ISTRAIN, the program generates training data and then derives the threshold values that provides the best IOU score. Otherwise, it takes the hard coded threshold value. In both cases, the program applies the threshold value to the test entries and generates cilia prediction png files
3. **PixelVariance.py** – Virtually the same as OpticalFlow.py in terms of functions: their purpose, the actions they perform, and the files read and/or generated. This program does not apply functions like calcOpticalFlowFarneback and cartToPolar, rather this function simply takes the raw pixel data from the files, scales their range from 0 to 255, and then derives a threshold value for application to test entries to generate cilia prediction png files.
4. **ToyOpticalFlow.ipynb** and **ToyVariance.ipynb** – These are Jupyter files used for visually testing the operations used in the OpticalFlow.py and PixelVariance.py files.

The PixelVariance.py efficiently performs a basic evaluation of frame values to find a simple threshold and produce predictions. Other than locations for input and output data, this program does not appears to carry any constant values that a user would want to modify. In contrast, the OpticalFlow.py reads and writes the same data types as PixelVariance.py, but OpticalFlow.py uses the calcOpticalFlowFarneback function using fixed values:

flow = cv2.calcOpticalFlowFarneback(preFrame, curFrame, None, 0.5, 3, 15, 3, 5, 1.2, 0)

Here are the function’s parameters (taken from <https://docs.opencv.org/3.4/dc/d6b/group__video__track.html#ga5d10ebbd59fe09c5f650289ec0ece5af>)

|  |  |
| --- | --- |
| **prev** | first 8-bit single-channel input image. |
| **next** | second input image of the same size and the same type as prev. |
| **flow** | computed flow image that has the same size as prev and type CV\_32FC2. |
| **pyr\_scale** | parameter, specifying the image scale (<1) to build pyramids for each image; pyr\_scale=0.5 means a classical pyramid, where each next layer is twice smaller than the previous one. |
| **levels** | number of pyramid layers including the initial image; levels=1 means that no extra layers are created and only the original images are used. |
| **winsize** | averaging window size; larger values increase the algorithm robustness to image noise and give more chances for fast motion detection, but yield more blurred motion field. |
| **iterations** | number of iterations the algorithm does at each pyramid level. |
| **poly\_n** | size of the pixel neighborhood used to find polynomial expansion in each pixel; larger values mean that the image will be approximated with smoother surfaces, yielding more robust algorithm and more blurred motion field, typically poly\_n =5 or 7. |
| **poly\_sigma** | standard deviation of the Gaussian that is used to smooth derivatives used as a basis for the polynomial expansion; for poly\_n=5, you can set poly\_sigma=1.1, for poly\_n=7, a good value would be poly\_sigma=1.5. |
| **flags** | operation flags that can be a combination of the following: |
|  | OPTFLOW\_USE\_INITIAL\_FLOW uses the input flow as an initial flow approximation. |
|  | OPTFLOW\_FARNEBACK\_GAUSSIAN uses the Gaussian winsize×winsize filter instead of a box filter of the same size for optical flow estimation; usually, this option gives z more accurate flow than with a box filter, at the cost of lower speed; normally, winsize for a Gaussian window should be set to a larger value to achieve the same level of robustness. |

These programs have not generated our optimal results, but we should supply this program to users with clear reference to the OpenCV function used along with options to override the constants we used when calling calcOpticalFlowFarneback.

**3.2) Found a pytorch tiramisu programs - jiahaoxu**

This is the PyTorch Tiramisu software loaded to GitHub by Brendan Fortuner (bfortuner) and others (<https://github.com/bfortuner/pytorch_tiramisu>). Of their many pieces, we initially got the models layers.py and tiramisu.py (with no apparent changes) along with the utility training\_utils.py (which appears to be a copy of the source file training.py with replacement of the function view\_sample\_predictions with functions get\_test\_results and get\_test\_results\_cpu). The utility file joint\_transforms.py came from another GitHub project (<https://github.com/ZijunDeng/pytorch-semantic-segmentation>) and it appears a number of ZujunDeng functions were dropped.

Here we discuss what the files submitted do and how they interact:

1. getCilia.py – This file provides a general function and a class with functions:
   1. load\_input – This function reads in the first frame (and mask when applicable) from the png files followed by formatting the image values using reshape and astype(np.uint8). One (or two) resulting lists are returned.
   2. CiliaData – This class stores data specified for Training, Validating, or Testing. The data consists of settings (input\_transform, trarget\_transform, joint\_transform, remove\_cell) along with imgs and masks lists for the values
      1. \_\_init\_\_ - Basis initialization function where the user may spefiy the following:
         1. root – Root directory holding both data and masks as subdirectories
         2. input\_transform –
         3. target\_transform –
         4. joint\_transform –
         5. remove\_cell –
         6. imgs – list of images from load\_input function
         7. masks – list of mask images from load\_input (when applicable)
      2. \_\_getitem\_\_ - returns the image (and mask when appropriate) for the index passed
      3. \_\_len\_\_ - returns the number of images stroed in the imgs list
2. layers.py – This is a copy of the code from bfortuner (see earlier). The classes and functions here appears to be an extension of the torch.nn neural network classes :
   1. DenseLayer – This is an extension of nn.Sequential and execute the forward function on nn.Sequential.
      1. \_\_init\_\_ - This takes in\_channels and growth\_rate to as some arguments for initializing certain modules (which appear needed by nn.Sequential):

self.add\_module('norm', nn.BatchNorm2d(in\_channels))

self.add\_module('relu', nn.ReLU(True))

self.add\_module('conv', nn.Conv2d(in\_channels, growth\_rate,

kernel\_size=3, stride=1, padding=1, bias=True))

self.add\_module('drop', nn.Dropout2d(0.2))

* + 1. forward – simply executes the nn.Sequential forward function
  1. DenseBlock – This is an extension of nn.Module
     1. \_\_init\_\_ - define the nn.Module properties upsample and layers using parameter variables:

self.upsample = upsample

self.layers = nn.ModuleList([DenseLayer(

in\_channels + i\*growth\_rate, growth\_rate)

for i in range(n\_layers)])

* + 1. forward – This function iterates through the layers, calculates the values, and returns what appears to be a matrix with values
  1. TransitionDown – The is an extension of nn.Sequential
     1. \_\_inti\_\_ - This initializes a number of modules used by nn.Sequential using many hardcode values:

self.add\_module('norm', nn.BatchNorm2d(num\_features=in\_channels))

self.add\_module('relu', nn.ReLU(inplace=True))

self.add\_module('conv', nn.Conv2d(in\_channels, in\_channels,

kernel\_size=1, stride=1,

padding=0, bias=True))

self.add\_module('drop', nn.Dropout2d(0.2))

self.add\_module('maxpool', nn.MaxPool2d(2))

* + 1. forward – Simply executes the nn.Sequential forward function
  1. TransitionUp - This is an extension of nn.Module
     1. \_\_init\_\_ - initializes the convTrans object with many hardcode values:

self.convTrans = nn.ConvTranspose2d(

in\_channels=in\_channels, out\_channels=out\_channels,

kernel\_size=3, stride=2, padding=0, bias=True)

* + 1. forward – This performs the convTrans function along with what appears to be cropping a number of steps, it appears to skip forward
  1. Bottleneck – This is an extension of nn.Sequential
     1. \_\_init\_\_ - Allows the user to define most of the variables passed to the super class function:

self.add\_module('bottleneck', DenseBlock(

in\_channels, growth\_rate, n\_layers, upsample=True))

* + 1. forward – Simply executes the nn.Sequential forward function
  1. center\_crop – This is the function called in TransitionUp forward function. This function appears to format the values in a layer using argument values max\_height and max\_width

1. tiramisu.py –
2. joint\_transforms.py –
3. training\_utils.py –

\*\* I had to hold off further analysis of this submit because I could not find a clear main function from which I could derive how this was used. I see there are some Jupyder files from other projects, but they don’t appear applicable for the results generated.

**3.3) Adding the files to untar all tar files and resizing all images in the dataset into same size – a-farahani**

The files in this submission are used to resize the images to match predefined dimensions, in this case that appears to be 256 by 256. To use this program, you would need to define the source directory storing the frame png files along with specifying a directory to hold all the “cropped” data in new subdirectories. The untar.py file appears to have been only a one time use file for moving things around.

We need to determine exactly how this was used to generate any of the results submitted to gain IOU scores. We could then provide users with a sequence of actions that will produce appropriate files.

**3.4) added a training/testing demo – jiahaoxu**

The submission contained a Jupyder file that came from a Github used in CSCI 8360 spring 2018. This document contains many good links, more specifically it provides:

1. "The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation" (<https://arxiv.org/pdf/1611.09326.pdf>)
2. Bendan Fortuner(@bfortuner)'s (https://github.com/bfortuner/pytorch\_tiramisu)
3. ZijunDeng(@ZijunDeng)'s (<https://github.com/ZijunDeng/pytorch-semantic-segmentation>)
4. "Automated identification of abnormal respiratory ciliary motion in nasal biopsies." Science translational medicine 7.299 (2015): 299ra124-299ra124. (<http://stm.sciencemag.org/content/7/299/299ra124>)
5. C. Lu, M. Marx, C. Chennubhotla, M. Zahid, C. W. Lo, and S. Quinn. "Stacked Neural Networks for Automated Ciliary Motion Analysis." IEEE ISBI. https://arxiv.org/abs/1803.07534

The Jupyder code appears applicable as a basis to build a main function for the packages referenced here. I believe we should implement these operations in a python main script and test the script to make sure the results generated match those submitted for scoring.

The code shows the following variables as important values that we may want to provide to users:

1. LR – def 1e-4
2. LR\_DECAY – def. 0.995
3. DECAY\_EVERY\_N\_EPOCHS – def. 1
4. N\_EPOCHS – def. 3000
5. tourch.cuda.manual\_seed – def. 0

It’s not clear exactly what these mean in the python torch package, so we should look to them (like the parameter list in section 3.1 above) to provide reasons why users might be interested in the use.

There are many comments and references, one of which stands out are the reference to FCDenseNet57, FCDenseNet67, and FCDenseNet103, along with notes on how FCDenseNet103 performed the best. I see how it is used to define the model, but maybe we should discuss how this works… time is the problem!

**3.5) Merge branch 'develop' of github.com:dsp-uga/team-linden-p2 into develop – jiahaoxu**

The submission contains the files image\_resized.py and untar.py, which show up in the section (3.3) above. Github does not distinguish any differences in the submission as opposed to the last. The comments on the submission suggest it was simply a Github synchronization operation (of some sorts).

**3.6) fixed issue 1 and issue 2 – a-farahani**

It appears many of the variable types in the training\_utils.py file needed adjustments for python to process correctly. There were also Jupyder and .gitignore changes, that might not be as important.

**3.7) added dice loss – jiahaoxu**

There were changes to the Jupyder main function appear to be many constants. It’s not clear why such changes were made, but it would make sense that testing showed certain changes would be advantageous. At this point deriving meaning for all of changes might be wasteful, but this entry will be revisited before implementation of a main function.

**3.8) updated .gitignore – a-farahani**

More changes to .gitignore, getCilia.py and the Jupyder main function. The changes appear small and we’ll see how well they hold up over other github submissions.

**3.9) Merge branch 'develop' of https://github.com/dsp-uga/team-linden-p2 into develop – a-farahani**

Stangly, Github states that no changes were made in this commit (so how was this a commit???)

**3.10) expanded the predicted cilia area– jiahaoxu**

The main function in AccuracyBoosting.py appears to iterate through a target directory containing mask files. The program reads through the row and columns to find cilia marked pixels in a 5 by 5 matrix. It appears the resulting png file generated inflates the mask areas by five pixels in every direction.

**3.11) Merge branch 'develop' of github.com:dsp-uga/team-linden-p2 into develop – jiahaoxu**

Just some basic changes to .gitignore and getCilia.py.

**3.12) updated the expansion window to 3x3 – jiahaoxu**

As discussed in earlier section (3.10), the AccuracyBoosting.py script iterates through a 5 by 5 matrix to determine if a point should be marked as cilia; basically expanding a prediction. The change in this commitment altered the matrix size to 3 by 3.

**3.13) preprocess.py for preprocessing the images – a-farahani**

This committal included a number of changes to Jupyter testing programs, but the introduction of the script src/preprocess.py stands out. This script includes a main function that calls the following three functions sequentially to generate and modify videos from frame images followed by extracting frames from the formatted videos:

1. image\_to\_video: This function pulls all the png imgaes together and generates a mp4 video using the cv2 library into an output directory.
2. video\_stabilizer: This function calls the stabailze function in the VidStab class, which comes from the library vidstab (<https://adamspannbauer.github.io/python_video_stab/html/index.html>)
3. processing\_fram\_from\_video: This function uses cv2 library to
   1. Read in video with VideoCapture
   2. Read out each frame with cvtColor
   3. Correct frame illumination by applying CLAHE
      1. The createCLAHE takes contants clipLimit=2.0 and timeGridSize=(8,8)
   4. Smooth the image using bilateralFilter with arguments (C. Tomasi and R. Manduchi, "Bilateral Filtering for Gray and Color Images", Proceedings of the 1998 IEEE International Conference on Computer Vision, Bombay, India.)
      1. diameter of each pixel neighborhood – 7
      2. sigmaColor color distance for mixing pixels together – 30
      3. sigmaSpace pixel distance for mixing pixels together – 30

It’s my understanding that the Jupyder scripts integrated the application of preprocess.py into the overall processing.

**3.14) Adding preprocessing.py – a-farahani**

This appears to be a second committal of the preprocessing.py script with no changes from the previous Github submission.

**3.15) changed the dropout rate – a-farahani**

Changed the torch neural network Dropout2d rate from 0.2 to 0.1. If this had any impact on performance, I believe adding this to the list of options we allow users to specify on execution will help – certainly shouldn’t hurt.

**3.16) changed number of frames from each video to increase the number of input of the training dataset– a-farahani**

Here we moved the getCilia.py program from reading in just one frame (frame0000.png) as input to the neural network to reading in five specific frames (frame0000.png, frame0020.png, frame0050.png, 0070.png 0090.png). The mask was copied five times as well, so the resulting CNN U-Net would have input and output matrixes with the same dimensions. At this point through, it appears only frame0000.png was still being used for test cases.

**3.17) Merge branch 'develop' of https://github.com/dsp-uga/team-linden-p2 into develop– a-farahani**

As the title suggests, Abolfazi integrated the AccuracyBoosting.py script into Github (still trying to understand how/when git regulates these actions since it appears the AccuracyBoosting.py here is no different from the one submitted earlier – a branch thing?)

**3.18) refactor the main.ipynb – a-farahani**

Not clear what changes were made to the main.ipynb program for testing. Perhaps just basic tests.

**3.19) added reference – jiahaoxu**

Added a citation to <https://docs.opencv.org/3.4/d7/d8b/tutorial_py_lucas_kanade.html> in the python script OpticalFlow.py. This reference is also noted in this document back in section (3.1)

**3.20) I added the directory submyers/imageRendering with the Perl program genHighlightVideo.pl (yes an old-school Perl script). The script takes a directory with all frames, a mask png file, and a prediction mask png file and generates a gif movie that highlights different areas based on how the two masks overlap. RED highlights areas where the masks agree, GREEN signifies areas only selected by the correct mask, and LIGHT BLUE highlights ares only selected by the prediction mask. This script is only applicable on OS X with the program 'convert' accessible through command line. – submyers**

This submission made changes to the Python scripts preprocess.py and OpticalFlow.py – changes that I (submyers) only intended for personal test, but git decided to commit those changes as well. The Perl script src/submyers/imageRendering/genHighlightVideo.pl only generates new gif movies that show how our predictions align with the mask predictions, this code does not generate solutions.

**3.21) Merge branch 'develop' of https://github.com/dsp-uga/team-linden-p2 into develop – submyers**

I (submyers) am not really sure why this git commit made the changes that Jiahao made in an earlier submission, see (3.19) above.

**3.22) refactor the cilia data input – a-farahani**

Here Abolfazi introduced the python script cilia.py as a replacement for the getCilia.py script defined in an earlier submission, see (3.2) above, along with continued reference to Brendan Fortuner. These changes moved from the getCilia.py, which only loaded one frame, to the five-frame approach used earlier, see (3.16) above. These changes appear to have the same functional impact as change (3.16), but the code looks more elegant.

**3.23) updated the tiramisu model – a-farahani**

Abolfazi expanded the python code format used for citing Brendan Fortuner as the original source for code.

**3.24) modified the resize method – a-farahani**

Abolfazi appears to have replaced the joint\_transforms.py script, from (3.2) above, with a new instance, now in the directory datasets, not utils. This implementation uses torch, torchvision, and skimage to resize an image to match a square image followed by padding any areas removed with 0 values.

\*\* NOTE: We should determine which variables settings were used to generate the different results submitted. That would give us a nice table for basic evaluation.

**3.25) refactored and cleaned the code – a-farahani**

Abolfazi crunched some code to make things look better!

**3.26) removed codes that not needed anymore – a-farahani**

This is where Abolfazi removed the getiCilia.py file and the joint\_transforms.py files, which he had replaced with cilia.py (3.22) and a new joint\_transforms.py (3.24) earlier.

**3.27) updated gitignore file – a-farahani**

As the name suggests, this tells git which files it should ignore for synchronization purposes. Abolfazi expanded the files to ignore for the src and src/tiramisu directories.

**3.28) updated path for the preprocess input data – a-farahani**

Applied a few directory path changes along with commenting out the bilateralFilter method.

**3.29) updated test image path – a-farahani**

A few, small changes to code and approval of certain changes made in (3.20) earlier.

**3.30) The software use and meaning document is in progress. I'm working to formalize the function of all code in the github package and determine where we could/should allow users to influence the functions executed. I will continue to update this document over the following days... – submyers**

First submission of this document. It was a start to adding documents to github for sharing and record purposes.

**3.31) Applying adabound optimizer and cleaning the main.py and modifying the cilia.py to feed more samples from each cilia – a-farahani**

Abolfazi both expanded the range of frames considered as CNN input (from [00,20,50,70,90] to [5,15,25,35,45,55,65,75,85,95]) along with the introduction of diceloss.py (from <https://discuss.pytorch.org/t/one-hot-encoding-with-autograd-dice-loss/9781/4> and <https://github.com/kevinzakka/pytorch-goodies/blob/master/losses.py>) and focalloss.py (from <https://github.com/clcarwin/focal_loss_pytorch/blob/master/focalloss.py>). These scripts appear to perform the following actions:

1. diceLoss.py: This is a CNN augmentation approach by which the program derives patterns in occurrence of different types in the training images to derive ratios applied to new, similar images. This approach has shown success altering the initial perspective of CNN. (Vladimir Iglovikov, Sergey Mushinskiy, Vladimir Osin, “Satellite Imagery Feature Detection using Deep Convolutional Neural Network: A Kaggle Competition”, arXiv:1706.06169)
2. focalloss.py: As the paper referenced suggests, broadening the possible conclusions considered expands one’s opportunity for finding an accurate prediction. (Tsung-Yi Lin, Priya Goyal, Ross Girshick Kaiming, He Piotr Dollar, “Focal Loss for Dense Object Detection”, arXiv:1708.02002v2 [cs.CV] 7 Feb 2018)

It appears Abolfazi tested these approaches in the Jupyder script.

**3.32) I'm continuing to expand the Github history analysis document. Once this document is more complete, we will be able to derive a plan for how the code should be used to generate the same results we generated along with demining which options would be appropriate to provide users– a-farahani**

I (submyers) uploaded another copy of this document to keep people updated on progress.

**3.33) fixed the resizing input images helper function – a-farahani**

Abolfazi rewrote and removed secions of the joint\_transforms.py file to make the code look better.

**3.34) fixed the resizing input images helper function – a-farahani**

Similar to (3.33) above, Abolfazi cleaned up the cilia.py code.

**3.35) modified the main.ipynb for the updated resized inputs – a-farahani**

Appears to have change the type passed in the training.py script from img to pic.

**3.36) Merge branch 'develop' of https://github.com/dsp-uga/team-linden-p2 into develop – a-farahani**

This was a git branch management commit change.

**3.37) modified the main.ipynb for the updated resized inputs – a-farahani**

More changes were made to the main.ipynb Jupyder code, but such code isn’t yet easy for quickly analyzing changes and impact.

**3.38) added reference for the joint\_transforms.py code – a-farahani**

The joint\_transforms.py code, generated in (3.24) earlier, now holds reference to Brendan Fortuner, see (3.2) above.

**3.39) Updated Readme – jiahaoxu**

First step in building a readme file

**3.40) Updated Readme – jiahaoxu**

Jiahao cleaned up many files, including .gitignore, cilia.py, joint\_transforms.py, and training.py along with the integration of diceloss.py and focalloss.py. These operations may have been git management operations.

**3.41) Here is an example for how I would write brief introduction to some of the ideas we used. I'm working on long, detailed sections for placement in our team's wiki page to show details on how we used their code, changes we made, and tests we ran (successful or not).**

As the description states, this was a simple change to illustrate what the final readme may consist of, along with notes stating further details will be placed in the team’s wiki page.