Project 2: Lightning Talk

Team Hasay

(Marcus Hill, Dhaval Bhanderi, Denish Khetan)

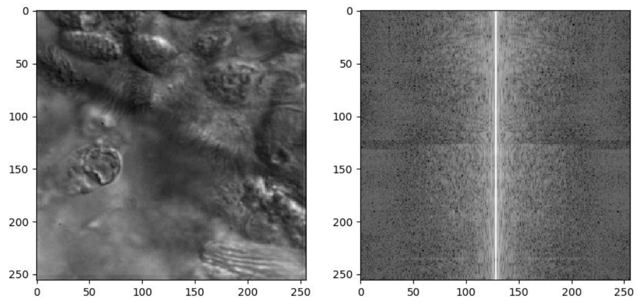
Model Architectures

Convolutional Neural Networks

- Our two primary models constructed, was recreations of the FCN8 and U-Net architectures.
- Adjustments we made to the default architectures:
  - Changed the input image dimensions from 224x224 to 256x256
  - Adjusted kernel sizes to accommodate the dimensionality reduction
  - Combined upsampled layers with their skip connections using Addition, rather than Concatenation (U-Net Model)
  - Used Xavier normal kernel initializers (glorot_normal) instead of He normal.

Data Pre-Processing

Fourier Transform



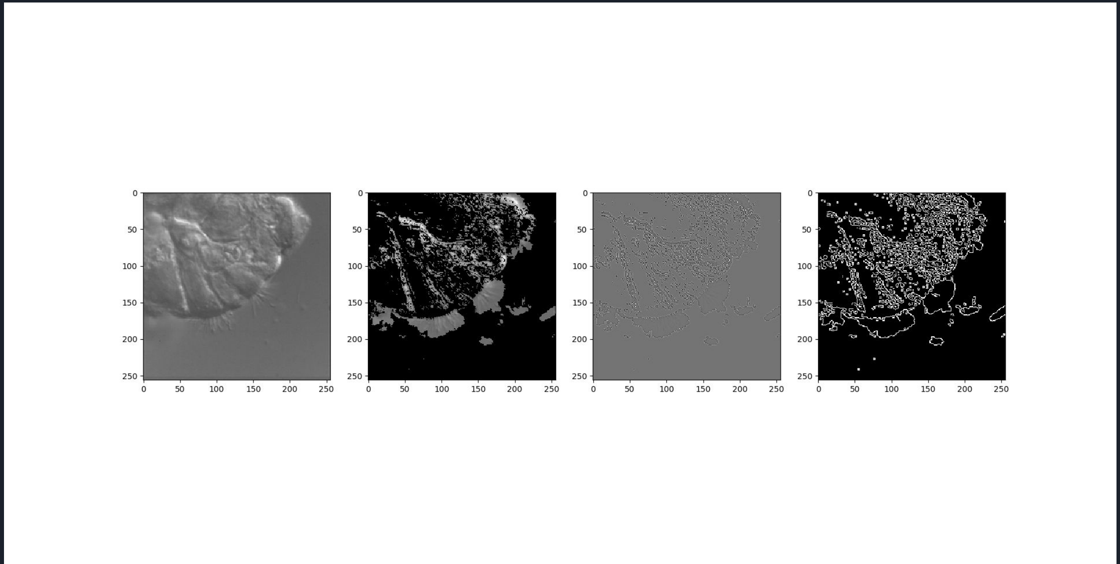
- Computed the Fourier transform for every image and supplied it as the input to the models.
- These images caused an explosion in gradients, caused NaN (Not a Number) exceptions to be reported by the loss functions.

:(

Thresholding using Pixel Variances

1. Calculated the variance of each pixel
2. Utilized the mean plus on standard deviation as the threshold value
    a. Assumptions are that the variances adhere to a normal (Gaussian) distribution
3. All values below the threshold pixel intensity was set to 0. Otherwise, the pixel retained it’s grays scale value.
    a. Chose gray value, so our models could have enough information to learn distinctions between image textures..

Original | Mean Threshold | Laplacian | Canny

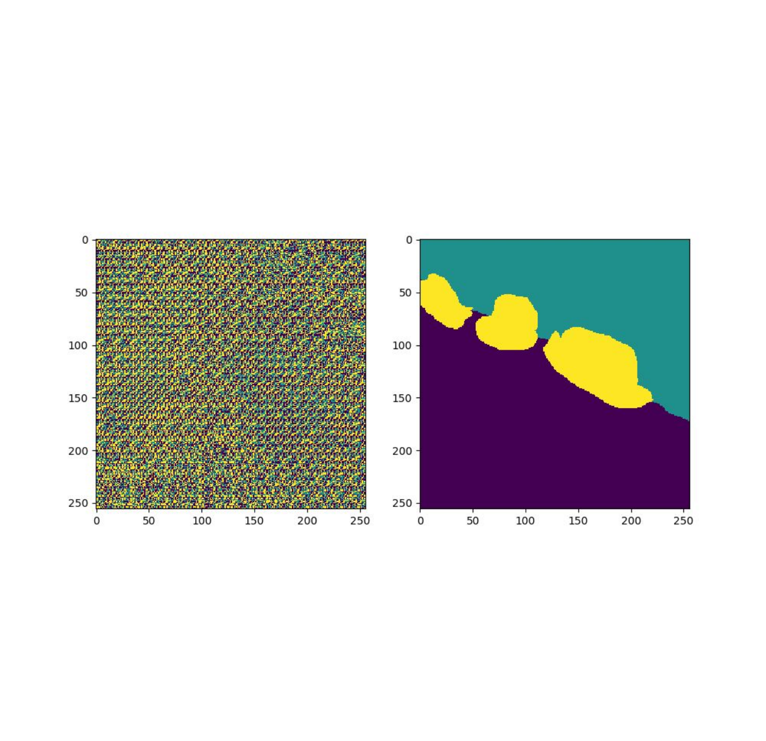


Model Pipeline

Pipeline Overview

1. Read in the thresholded images, which are representative of the videos.
2. Resize all images to 256x256, but keep the original images information.
3. Normalize all the input.
    a. (pixel intensity - mean) / std
4. Pass the input into a CNN
5. Resize the output masks to their original dimensions.

Hiccups along the way



- We learned the hard way, that the prediction masks needed to be one-hot encoded for the pixel labels. For the model to learn.
    - We were confusing the 3 rgb dimensions of an image with the 3 layers the model were expecting based on classes.
- The result is completely random masks, with training and validation accuracy around 33%.

Future Goals

If we had more time to work on the project, would love to explore incorporating:

- Markov Random Fields(Could not complete)
  - A joint probabilistic model over the pixel value and the hidden variable.
- Conditional Random Fields

References

1. FCN Paper
2. U-Net Paper
3. https://fairyonice.github.io/Learn-about-Fully-Convolutional-Networks-for-semantic-segmentation.htm
4. OpenCV Fourier Transform: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_transforms/py_fourier_transform/py_fourier_transform.html
5. https://medium.com/coinmonks/learn-how-to-train-u-net-on-your-dataset-8e3f89fbd623
6. https://github.com/AliMorty/Markov-Random-Field-Project
7. https://github.com/zhixuhao/unet