DL for Segmentation of Intracranial vessel wall pathologies

Anitha Bhat Talagini Ashoka,
Pavan Kumar Kandapagari.

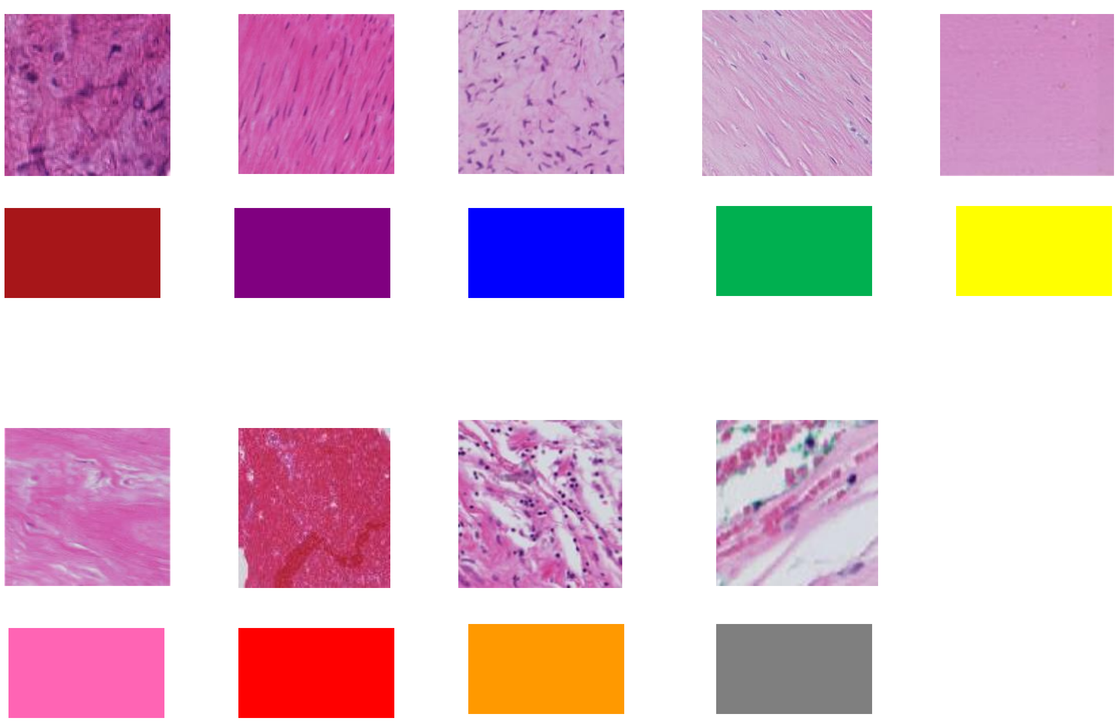
Recap from previous work

- Problem statement

- Literature survey
- Technologies

Recap from previous work

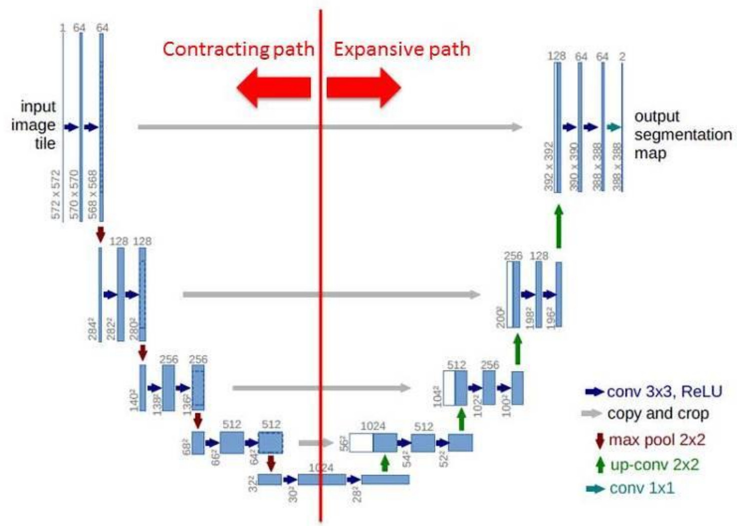
- Classes



Architecture selection

- U-net[1]
- Modification according to our problem statement

Network Architecture



[1] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmen-tation. InMedical Image Computing and Computer-Assisted Intervention (MICCAI), volume 9351 ofLNCS, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]).

Pipeline

- Data pre-processing
  - Patchwise approach
  - Patch size 256*256

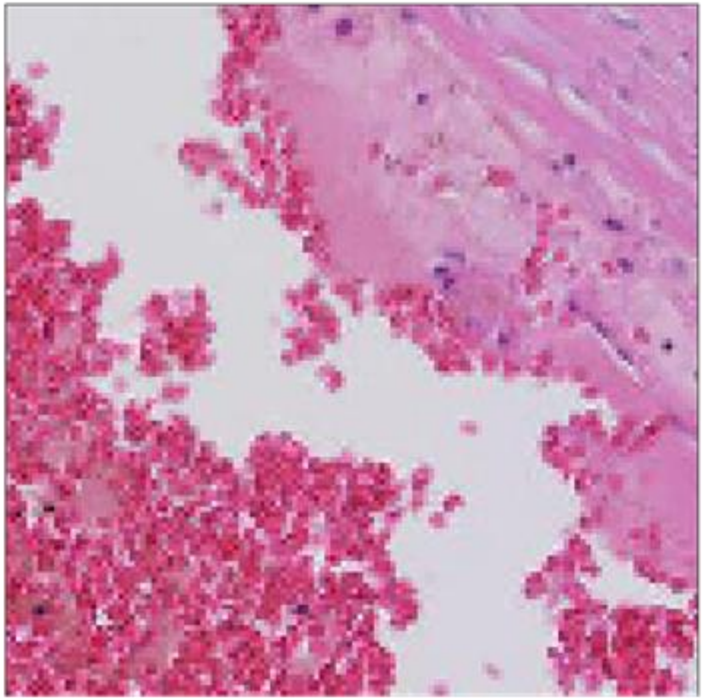
- Problems faced during data pre-processing
  - Single generator could not handle to give both image and mask, model.fit will accept only one generator
  - .tif mask size was huge, could not fit in RAM
  - Convert 3 channel mask into 9 channel mask
  - RGBA format masks
  - Mismatch between image and mask patches
  - Class mismatch because of high number of white patches

Pipeline

- Solutions
  - Make different generators for both image and mask, connect them in a function using yield
  - Conversion of masks from .tif to .png
  - One-hot encoding
  - zip image and mask generators

Pipeline

Image



Mask

Pipeline

- Model Development
  - Model built in python using TensorFlow 2.0 in Jupyter notebook.
  - Modification from binary classification to 9 different classes
  - Last layer of the original model is 2 channels i.e., 256X256X2 which is changed to 9 channels i.e., 256X256X9.
  - To reduce the number of parameters we reduced the number of filters half fold by which parameter reduced from 2,158,841 to 540,298

Pipeline

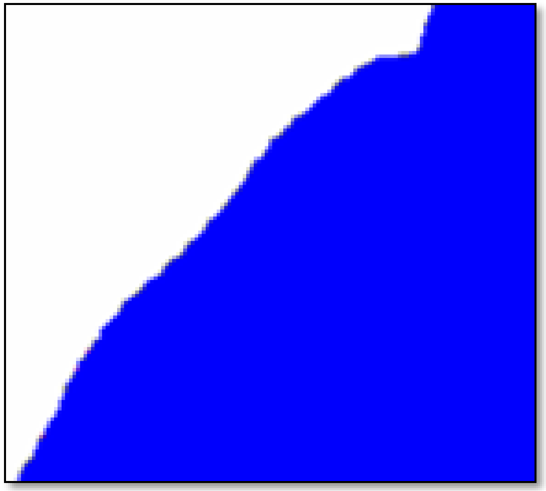
- Model Development
  - Loss function selection
    - Categorical cross entropy
    - Dice loss
  - Metrics
    - Accuracy
    - Dice coefficient
    - Precision
    - Recall

Pipeline

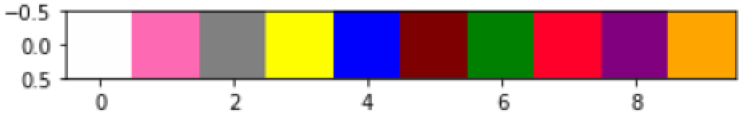
- Problem with converting masks into labels
  - Photoshop artifacts – RGBA,
  - Color interpolation at the edges
  - One hot encoding – shape issue– 256X256X1X9

Pipeline

- Solutions for converting masks into labels
  - Script for converting RGBA images into RGB
  - Using K-Means algorithm to get the labels
  - Using np.squeeze to get 256X256X9, as a result have to use np.expand_dims while converting back to 3 channel image



Pipeline

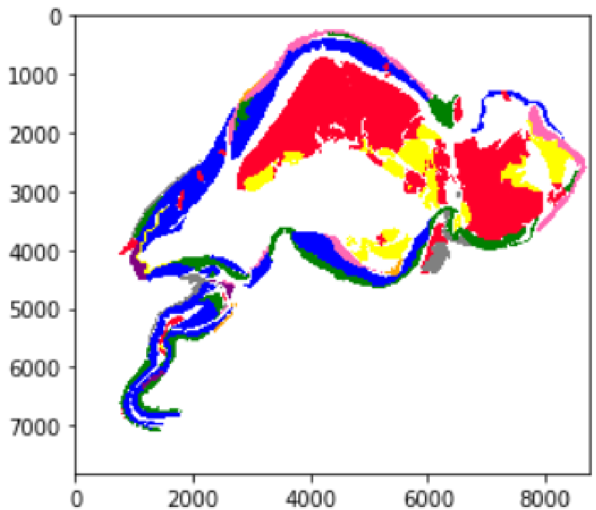


- K-Means clustering
  - K-means semi-supervised
  - Cluster centers as class labels

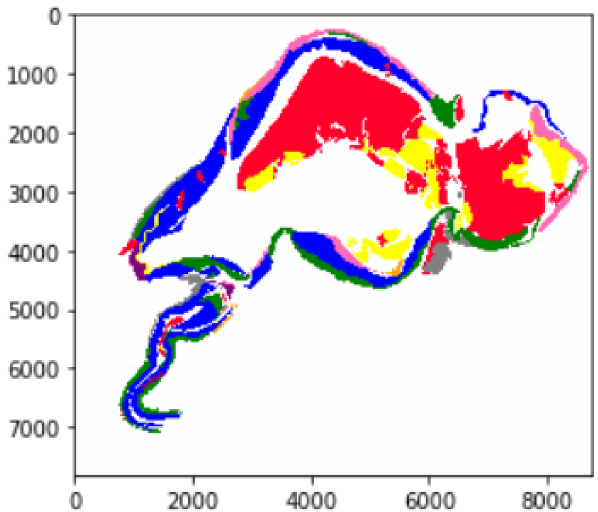
```
t0 = time()
color_palette_2, w, h = preprossesing_image(r'/data/student/github/DLforWallCharacteristics/color_palette_
2.jpg')
kmeans_color_palette_2 = KMeans(n_clusters=n_colors, random_state=42).fit(color_palette_2)
print('done in %0.3fs. ' % (time() - t0))
print(kmeans_color_palette_2.cluster_centers_)
```

Pipeline

Mask read from disk



Mask after K-means clustering



Training

- Hyper perameters
  - Optimizer = Adam
  - Loss function = categorical cross_entropy
  - Activation function(non classifying) = ReLu
  - Learning rate= 0.001
  - Dropout rate = 0.5
  - Kernel shape
  - Regularisation = L2
  - Weight decay= 0.001

Training

- Callbacks

  - Early stoppage. ✗
  - ReduceLRonplateau – for dynamically reduce learning rate
  - Model Checkpoint – for saving model with best weights (.h5 format)
  - CSVlogger – for logging loss and metrics in a csv file.

Problems in training

- Upgrade the code from TensorFlow 1.14 to TensorFlow 2.0
- Difficultly to fit the data on memory.
- Server issues due to incompatibility of cuda version – subsequent change of servers.

Further development

- Change the loss from categorical cross entropy to dice loss which takes care of pixel wise dependencies.
- Combining the very similar classes, and try the network
- If multi class approach fails – have a one vs many approach converting 9 class problem into 9 binary classification problems.
- Complete the project report.

Thank you for your attention!

Any Questions?

Back up

Color palate