CSE 627-Advanced Machine Learning

Kimberly Hamilton

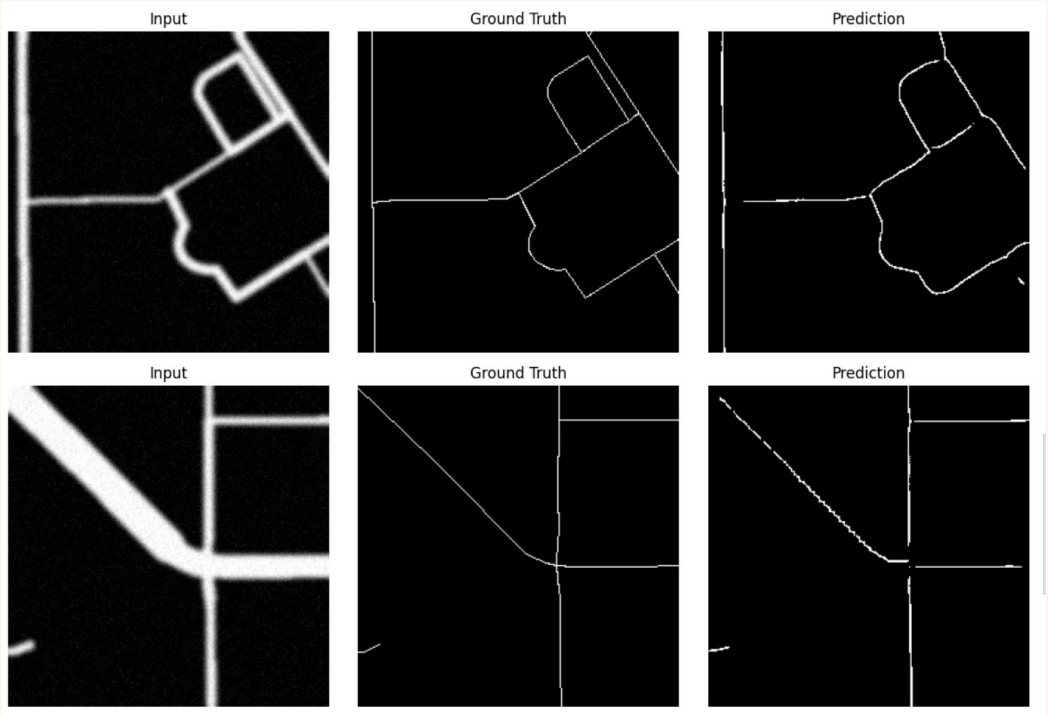
Road Network Skeletonization

Kimberly Hamilton

Qualitative Results (Base Model Test)

****Same parameters as advanced except does not consider advanced multitask implementations.**

Base model manages to have a fairly good prediction skeleton with some issues around intersections missing pixel data. Also intersections have concave behavior vs. straight runs.



Qualitative Results (Advanced Model Test 1)

Model Type:

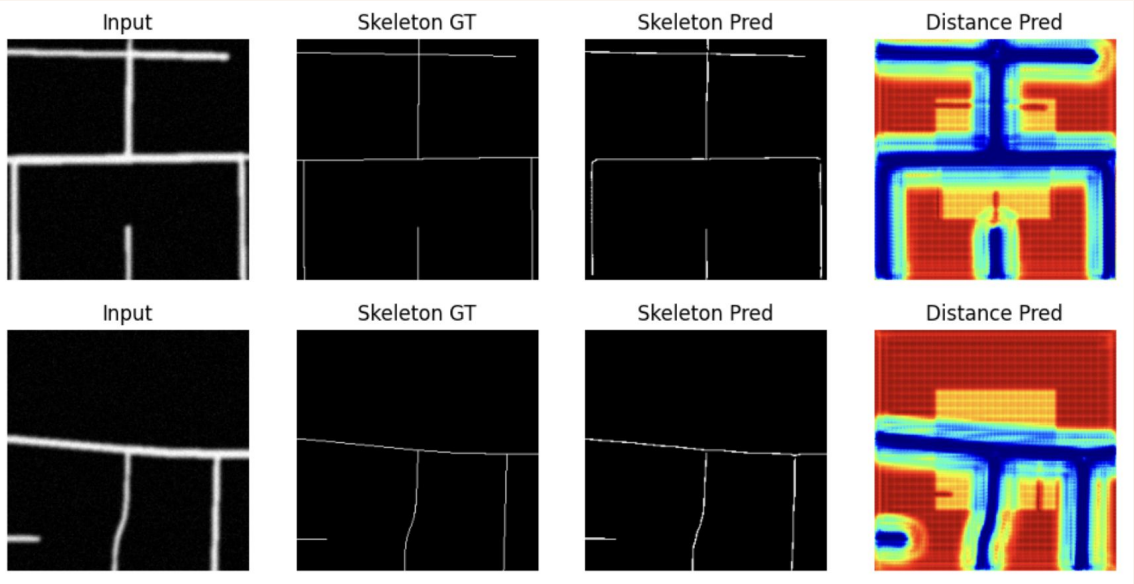
→ Advanced with multitask learning using distance for support

Architecture:

→ Standard U-Net with stride convolutions instead of maxpooling and skip connections

Learning Rate: 1e-3

Loss Function: Combined (BCE + Dice)

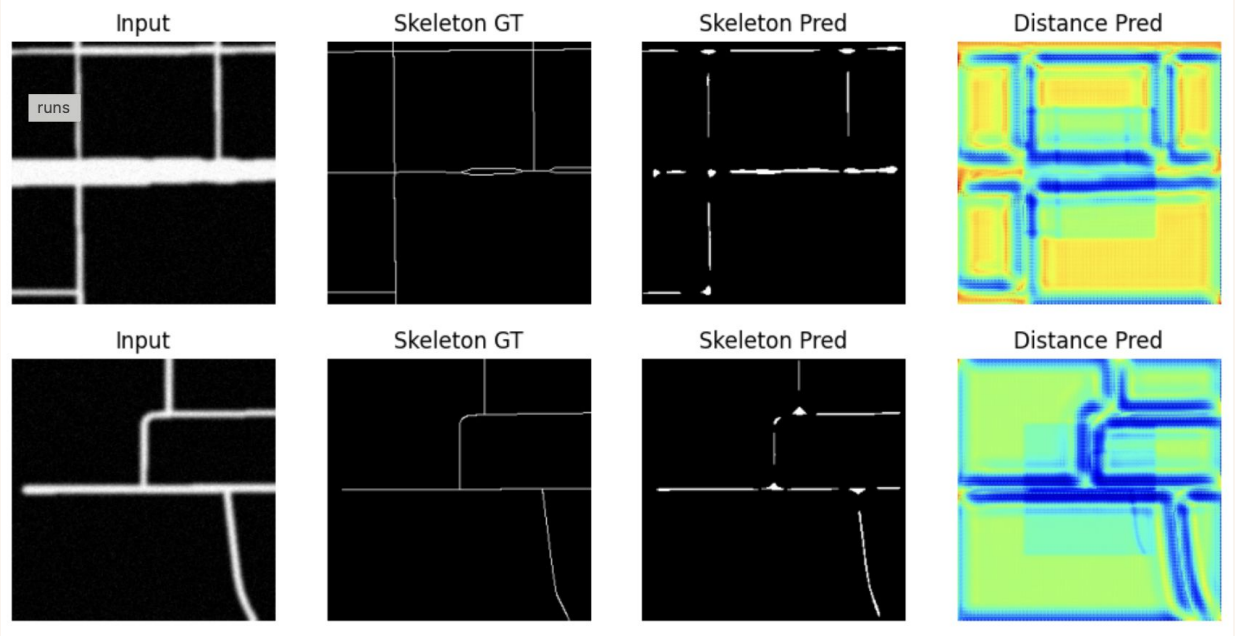


Qualitative Results (Advanced Model Test 2)

New Test Change from Best Model:

Change in learning rate from 1e-3 to 1e-5

Obvious, color distortion in distance prediction and major gaps in skeleton prediction while having large congested pixel areas around intersections.

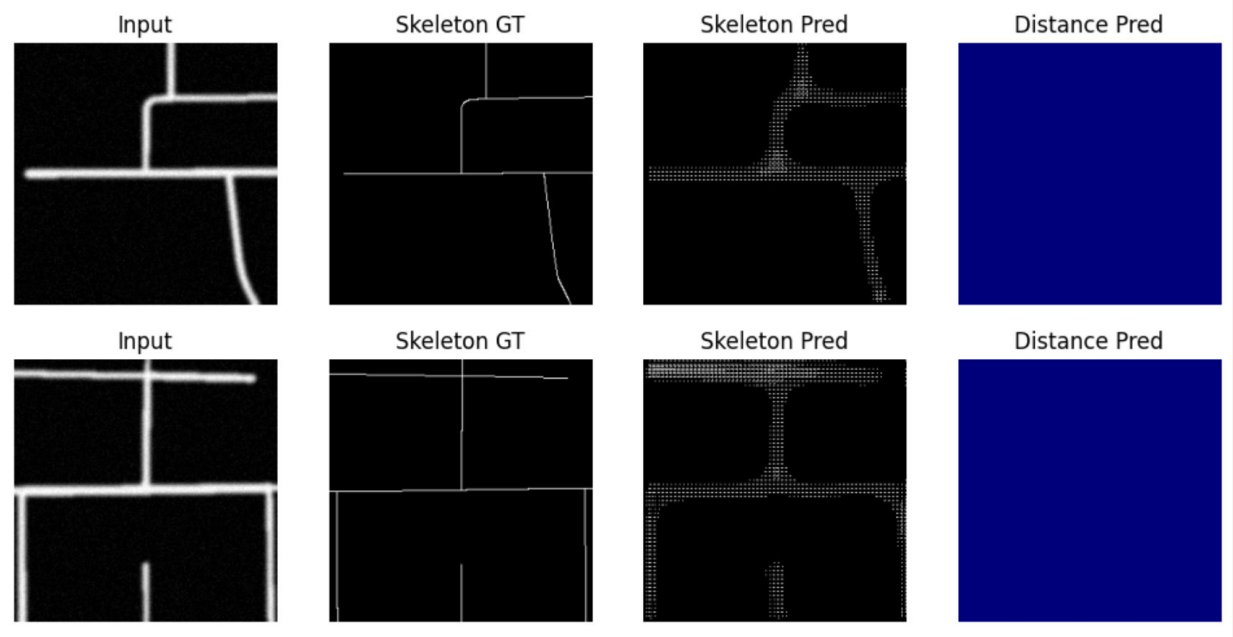


Qualitative Results (Advanced Model Test 3)

New Test Change from Best Model:

Change in learning rate from 1e-3 to 1e-1

No colors at all in distance prediction. Skeleton prediction similar but not single pixel but almost multi pixel dots but same roadways overall.

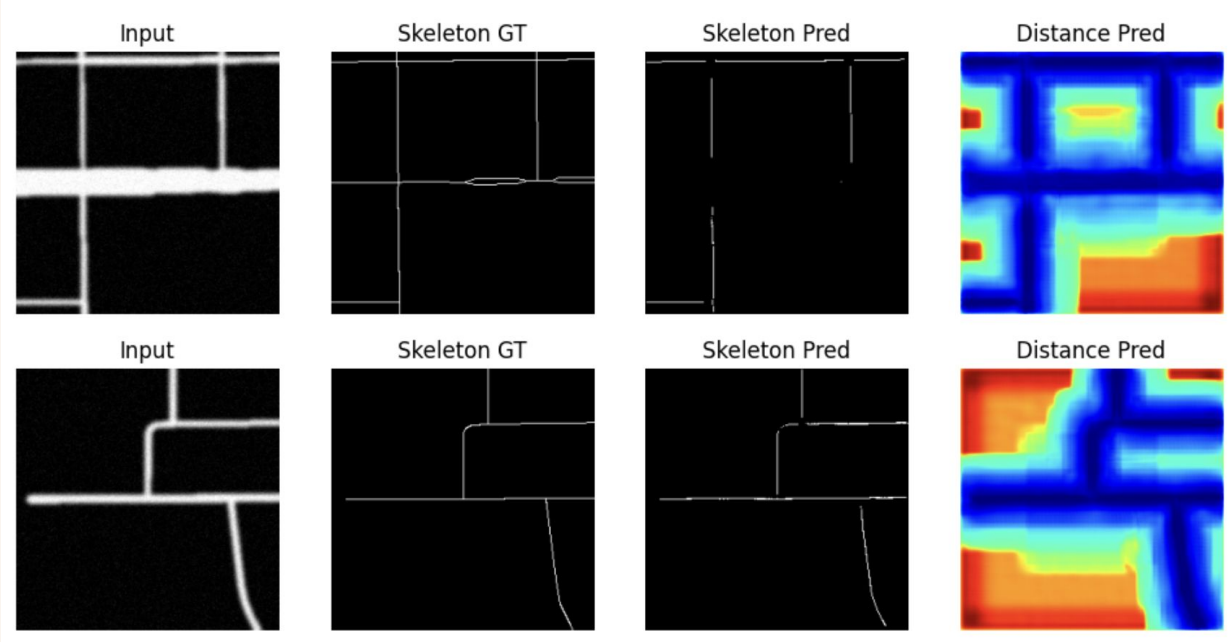


Qualitative Results (Advanced Model Test 4)

New Test Change from Best Model:

Change in Loss function from combined BCE and Dice Loss to only BCE loss

Distance prediction coloring more blobby and not as precise in distance almost like a smudge. Skeleton Prediction off in first test image around thick line masses but fairly accurate in second image test.

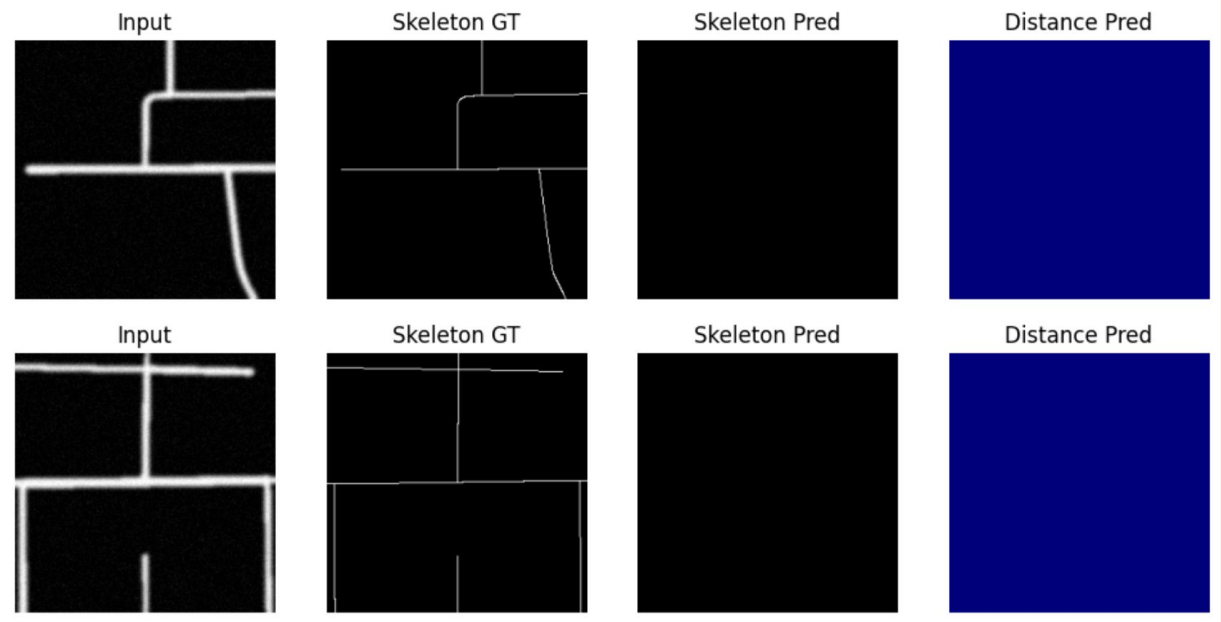


Qualitative Results (Advanced Model Test 5)

New Test Change from Best Model:

Change in Loss function from combined BCE and Dice Loss to only Dice loss

No colors at all in distance prediction from best model and no skeleton prediction image. Poor model.

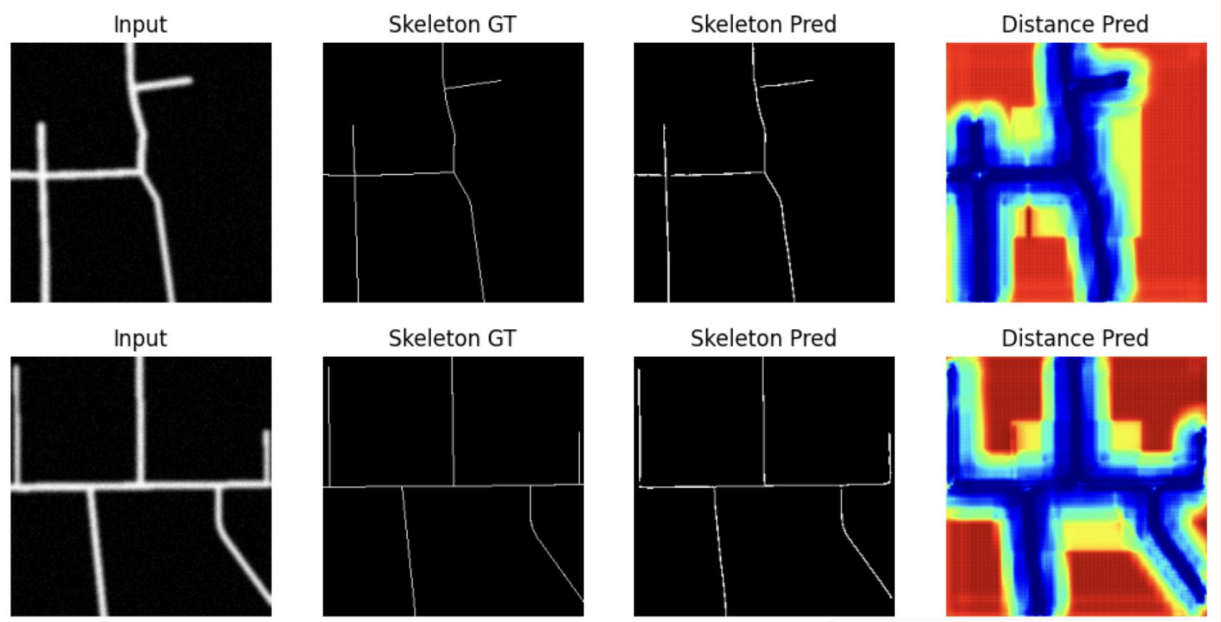


Qualitative Results (Advanced Model Test 6)

New Test Change from Best Model:

Change in model architecture from stride convolution to maxpooling

Distance prediction coloring is not bad here and is close to best model does have some blobbing in areas making it maybe not as precise. Skeleton prediction is fairly accurate.



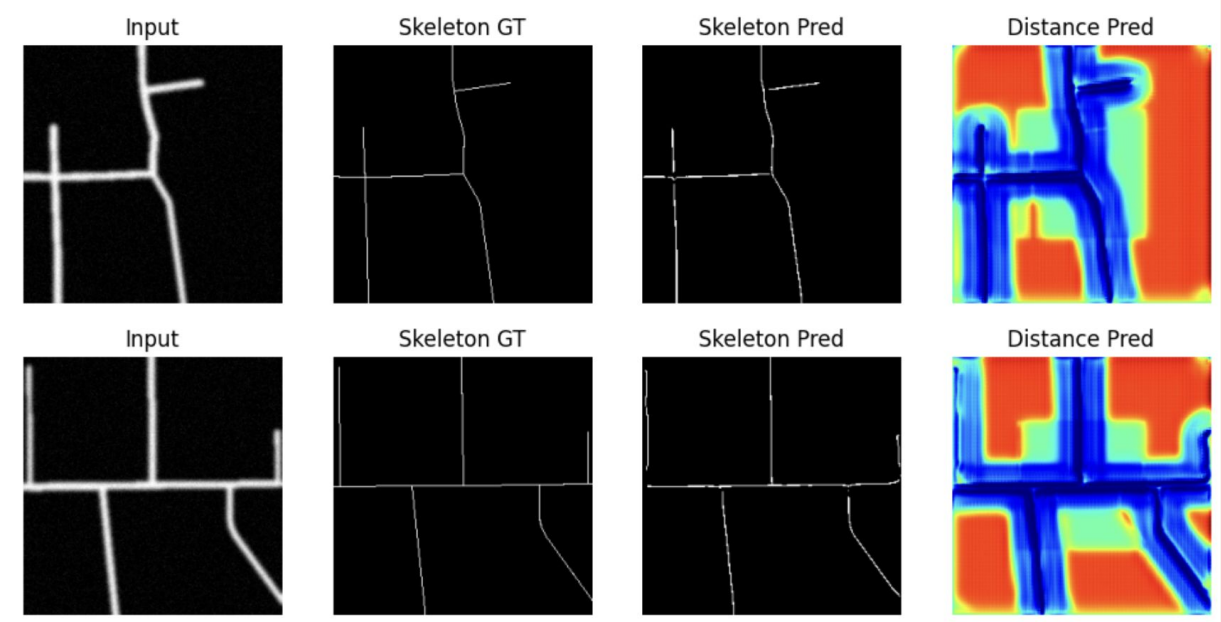
Qualitative Results (Advanced Model Test 7)

Best Model

New Test Change from Best Model:

Change in model architecture from skip connections to no skip connections

Stronger Distance prediction connection. Dark blue color is much smaller and closer to a a 1 pixel line image. Skeleton Prediction fairly accurate with some missed pixels.



Quantitative Results (Base Model vs Advanced Model Tests)

For this set of quantitative data Test 7 has two of the best features including IoU and Dice Coefficient. These values represent the following:

IoU: The overlap between the predicted segmentation and the ground truth, defined as the area of overlap divided by the area of union

Dice Coefficient: Measures similarity between the predicted and ground-truth masks, giving twice the weight to the intersection

** Bolded values in each column are best value for column feature, second best is italicized

Test	Loss / Arch / Model	Training Time (s)	Test Loss	MSE (DT)	IoU	Dice Coef
1	1e-3, BCE+Dice (stride)	314.21	0.4215	2772.7609	0.4651	0.6221
2	1e-5, BCE+Dice (stride)	312.65	0.5880	2772.7433	0.3377	0.4960
3	1e-1, BCE+Dice (stride)	298.69	6.97E+18	2772.5817	0.0415	0.0794
4	1e-3, BCE only (stride)	313.47	**0.0156**	2772.7630	0.4168	0.5659
5	1e-3, Dice only (stride)	*298.63*	0.9808	**2693.4230**	0.0097	0.0192
6	1e-3, BCE+Dice (maxpool)	327.08	0.3806	*2711.9125*	*0.4985*	*0.6558*
7	1e-3, BCE+Dice (no skip)	328.90	*0.3726*	2711.9133	**0.5111**	**0.6672**
8	Base model	**223.42**	0.4601	2423.3056	0.4217	0.5880

Quantitative Results (Base Model vs Advanced Model Tests)

For this set of quantitative data it seems like Test 6 did the best overall with having 3 best values in 3-val Rec, 4-val Prec, and 4-val Rec.

Node Precision: For each node valence, precision is the fraction of predicted nodes of that valence that match a ground-truth node within a certain distance (e.g., 3 pixels)

Node Recall: For each node valence, recall is the fraction of ground-truth nodes of that valence that are matched by a predicted node within a certain distance.

Test	1-val Prec	1-val Rec	2-val Prec	2-val Rec	3-val Prec	3-val Rec	4-val Prec	4-val Rec
1	0.449	**0.503**	0.942	0.413	**0.138**	*0.416*	*0.038*	0.554
2	0.114	0.152	**0.962**	0.139	0.092	0.075	0.024	0.398
3	0.001	0.144	0.491	0.059	0.000	0.000	0.000	0.000
4	0.180	*0.496*	0.932	**0.583**	0.073	0.131	0.028	0.065
5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	**0.516**	0.416	*0.951*	*0.442*	*0.129*	**0.428**	**0.042**	**0.636**
7	0.451	0.483	0.937	*0.442*	0.121	0.386	**0.042**	0.501
8	*0.511*	0.439	0.949	0.381	0.096	0.321	**0.042**	*0.583*

Next Section:
Overview of design and thought process

```
/content/data/thinning/Oxford_split/
    train/
        images/
        skeletons/
    val/
        images/
        skeletons/
    test/
        images/
        skeletons/
```

Goal:
Extract one-pixel-wide road skeletons from thick, noisy road network images generated from OpenStreetMap (OSM) data.

Challenge:

- OSM-based road images often have variable thickness, noise, and distortions.

Approach:
Develop a deep learning-based model that refines thick road masks into accurate, topologically correct skeletons.

Data Generation

→ Base python file (make_data.py) provided by professor Femiani for OSM data extraction

→ A realistic dataset was generated of thick, noisy road network images and their corresponding thin skeletons using OpenStreetMap (OSM) data.

→ Ideally, images were created in hopes the model could learn to handle real-world imperfections and not be bias to a perfect data set.

Key Steps:

OSM Road Data Extraction

Target city - *Oxford, Ohio, USA*

- Extracted road network using OSMnx an open-source Python library used for downloading and utilizing OpenStreetMap data.

Several road types were available to use in order to create and download image datasets. I ran the base model and reviewed which road types were showing present in Oxford, OH to determine an appropriate subset.

Filtered subset for relevant road types:

- *subset_types* = [‘residential’, ‘tertiary’, ‘trunk’, ‘primary’, ‘secondary’, ‘trunk link’]
- Removed ‘unclassified’ data

Rasterization:

- **256x256** pixel image
- Road thickness assignment based on road type

```
DEFAULT_THICKNESS = {
    'trunk': (15, 30),
    'primary': (10, 25),
    'secondary': (7, 15),
    'tertiary': (5, 10),
    'unclassified': (3, 7),
    'residential': (4, 8),
    'trunk_link': (6, 12),
}
```

Data Distortions:

- Applied Gaussian blur parameter to simulate low-resolution or not focused images , **“blur sigma”: 2**
- Added random Gaussian noise parameter to mimic sensor or possible environmental imperfections
- Removed ‘unclassified’ data, **“noise_std”: 10**

Ground Truth Skeleton Creation:

- One-pixel-wide skeleton created for each image by rasterizing the same roads with zero thickness

Saving Outputs:

- Saved the thick, noisy road mask as the input image
- Saved the thin skeleton as the target image.
- Exported corresponding road vector data as GeoJSON for references

Sample Size: “n_samples”: 450

Data Preparation (Split)

→ data paired between noisy images and skeleton image

→ pairs shuffled and randomly selected for dataset splits

Dataset Split:

- Training | **70%**
- Validation | **15%**
- Test | **15%**

→ Images were converted to grayscale(“L”) i.e. 1 channel and resized(256x256) even if not needed

Saved splits:

→ New split datasets saved in designated directories

```
/content/data/thinning/Oxford_split/
    train/
        images/
        skeletons/
    val/
        images/
        skeletons/
    test/
        images/
        skeletons/
```

Base Model Approach

→ **Data Loaders** created for train, validation and test sets

- batch size=32
- train| shuffle=”true” to add random shuffle
- validation & test| shuffle=”false” to test against

→ **U-net Model**

Encoder Structure (Three blocks)

- two convolutional layers per block
- Strided convolution (stride=2) for downsampling
- LeakyReLU activation (slope=0.1)

Feature Reduction:

- Spatial size halves at each block (stride=2)
- Channel dept doubles (64 → 128 → 256)

Bottleneck Structure:

Processes the most compressed features with:

- Two 3x3 convolutions
- Channel depth increase to 512

Decoder Structure (Three blocks):

- For upsampling transposed convolution (upconv) used
- two convolutional layers per block

Feature Reduction:

- Spatial size doubles at each block
- Channel dept halves (256 → 126 → 64)

Final Layer:

- Maps features to 1 channel output

Design Choices

→ **U-Net model** → keeps the classic symmetry of channels downsampling to upsampling

→ **Strided Convolutions** (stride=2) instead of maxpooling for downsampling learning

→ **LeakyReLU** over regular ReLU activation function can help with stability

→ **Transposed Convolutions** for upsampling

→ **Skip connections and center cropping** to help maintain spatial map size between encoder and decoder regions where stride convolution or transpose could mismatch these sizes potentially

Most convolutions used:

→ **Kernal-size** 3x3 convolutions

→ **Padding** 1

→ **Final output** 1x1 convolution

Model Training

Loss Functions

- **Dice Loss:** used to measure overlap from predicted and ground truth skeletons
- **BCEWithLogitsLoss:** binary segmentation standard, combines sigmoid with binary cross-entropy
- **Combined Loss:** Adds both losses to help with pixel accuracy and correct skeleton

Optimizer

- **Adam** to update the neural network weights during training

Training Loop

→ Epochs: 20

→ *Batch Loop*

- Forward Pass → Loss Calculation → Backward Pass

→ *Validation Loop*

- After epoch, evaluate model with validation set with no gradient updates

→ *Logging*

- TensorBoard

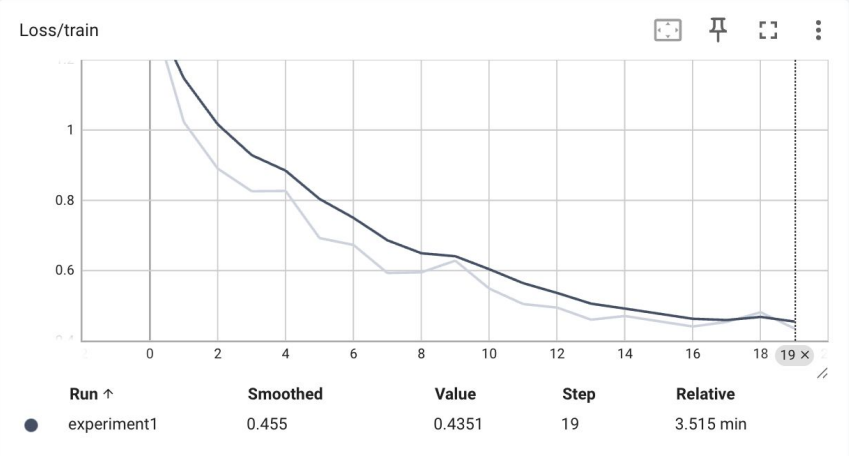
Epoch Loss Evaluation

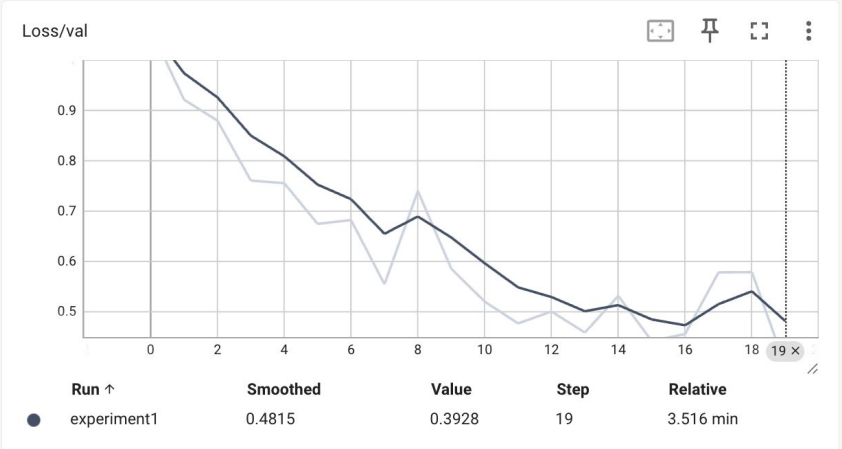
Epoch-TrainingLossavg-ValidationLossavg-EpochTimesec (1)

Epoch	Training Loss (avg)	Validation Loss (avg)	Epoch Time (sec)
1	1.3548	1.0616	11.69
2	1.0228	0.9214	10.62
3	0.8902	0.8798	10.68
4	0.8258	0.7607	10.62
5	0.8267	0.7555	10.80
6	0.6925	0.6747	10.78
7	0.6731	0.6823	10.83
8	0.5932	0.5548	10.97
9	0.5951	0.7400	11.09
10	0.6283	0.5857	11.25
11	0.5489	0.5203	11.26
12	0.5049	0.4770	11.36
13	0.4949	0.5006	11.42
14	0.4603	0.4589	11.33
15	0.4709	0.5316	11.33
16	0.4558	0.4422	11.25
17	0.4410	0.4559	11.31
18	0.4541	0.5783	11.30
19	0.4816	0.5788	11.40
20	0.4351	0.3928	11.37

Total Time : 222.66 seconds

Epoch Loss Evaluation (Tensorboard)

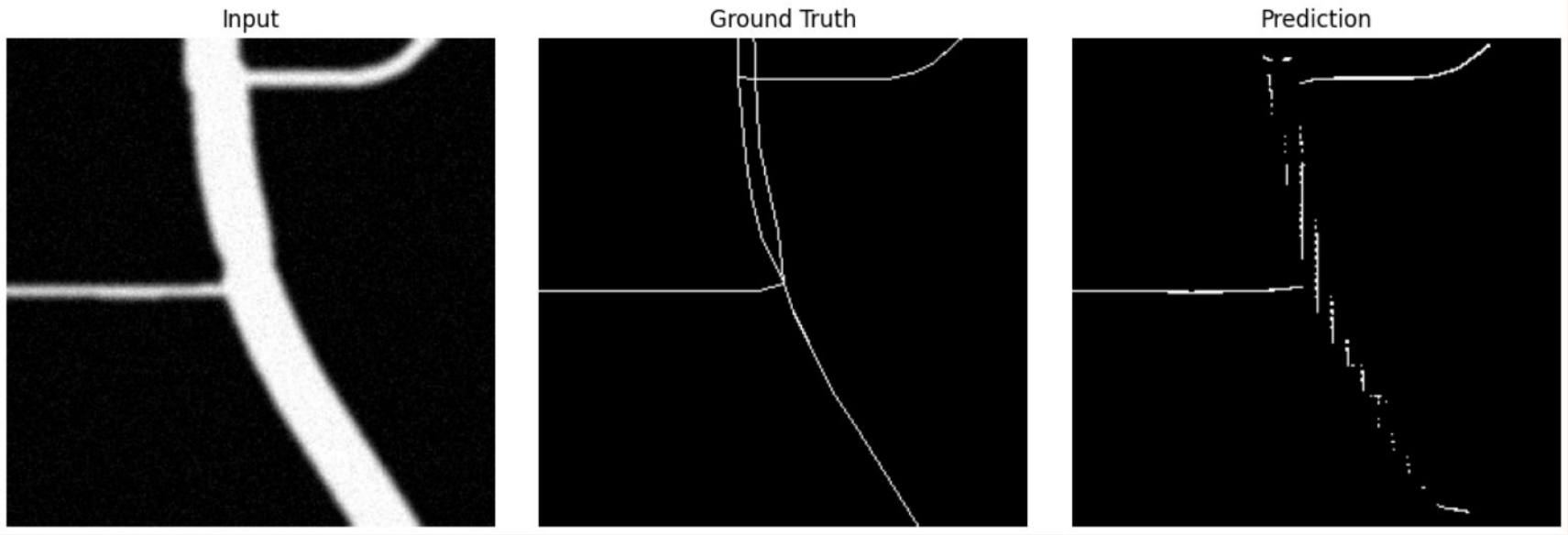




Train Loss → training loss decreases over epoch time

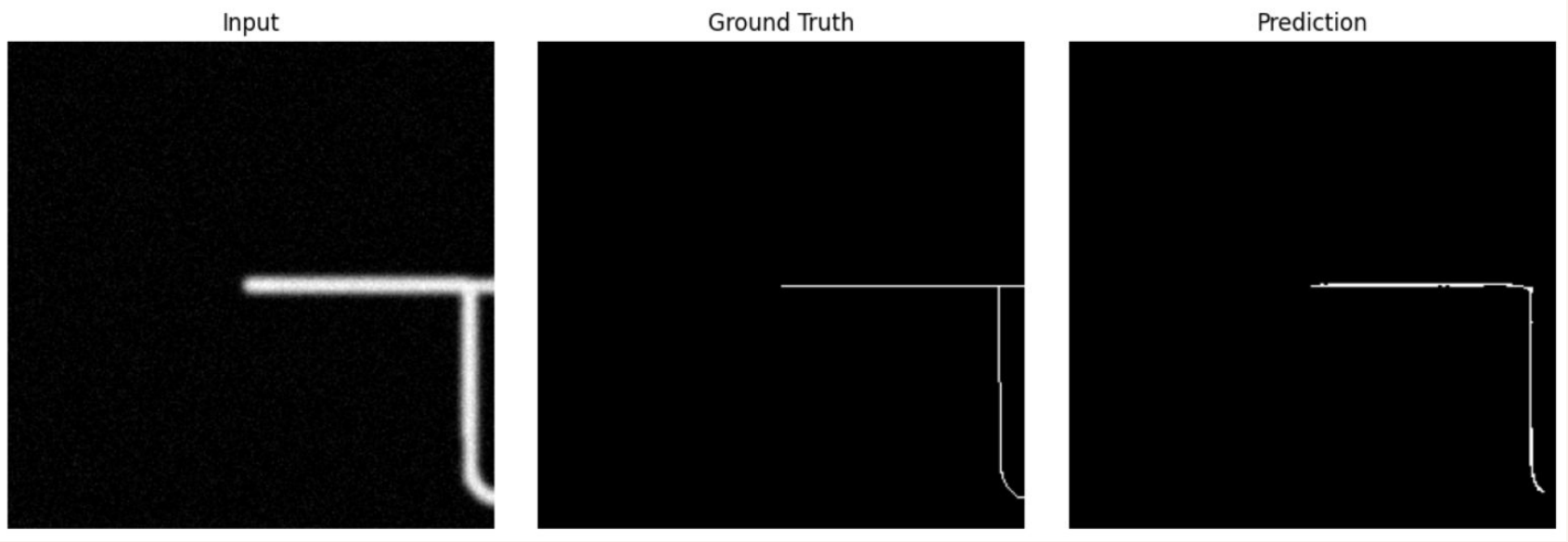
Val Loss → validation loss decreases over epoch time but some variability/noise in graph

→ Ground truth created two pixel lines for think input noisy image causing issues in prediction image.

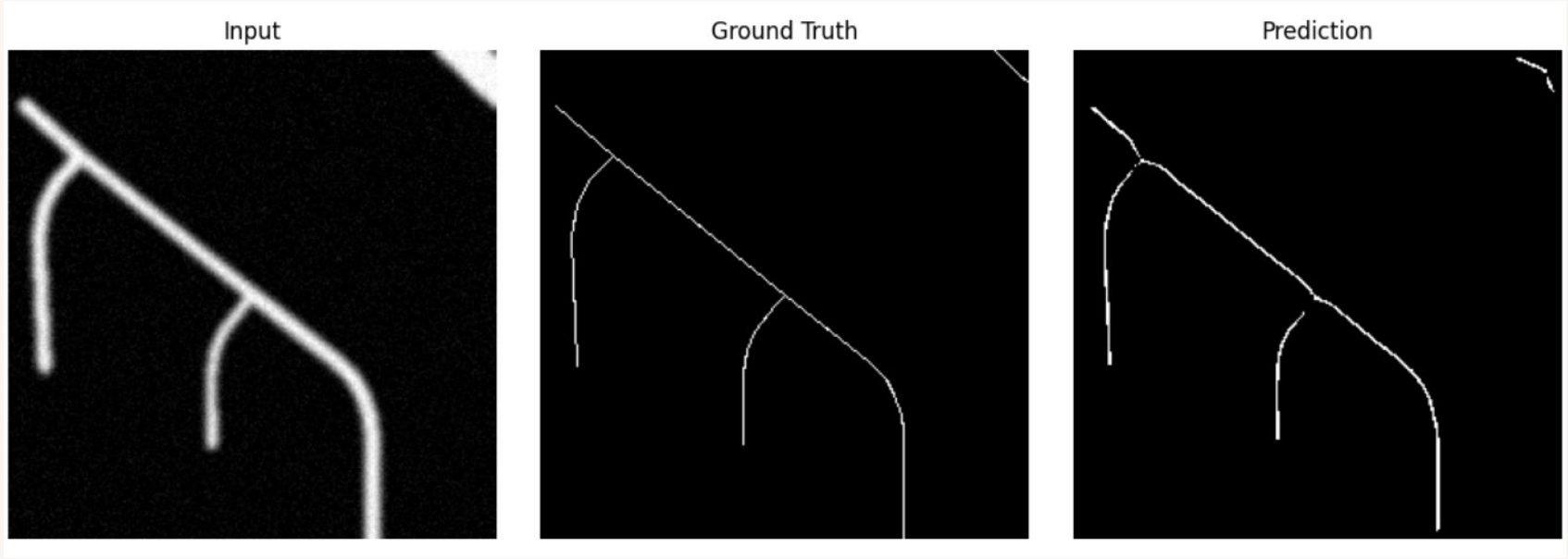


Prediction image Visualization

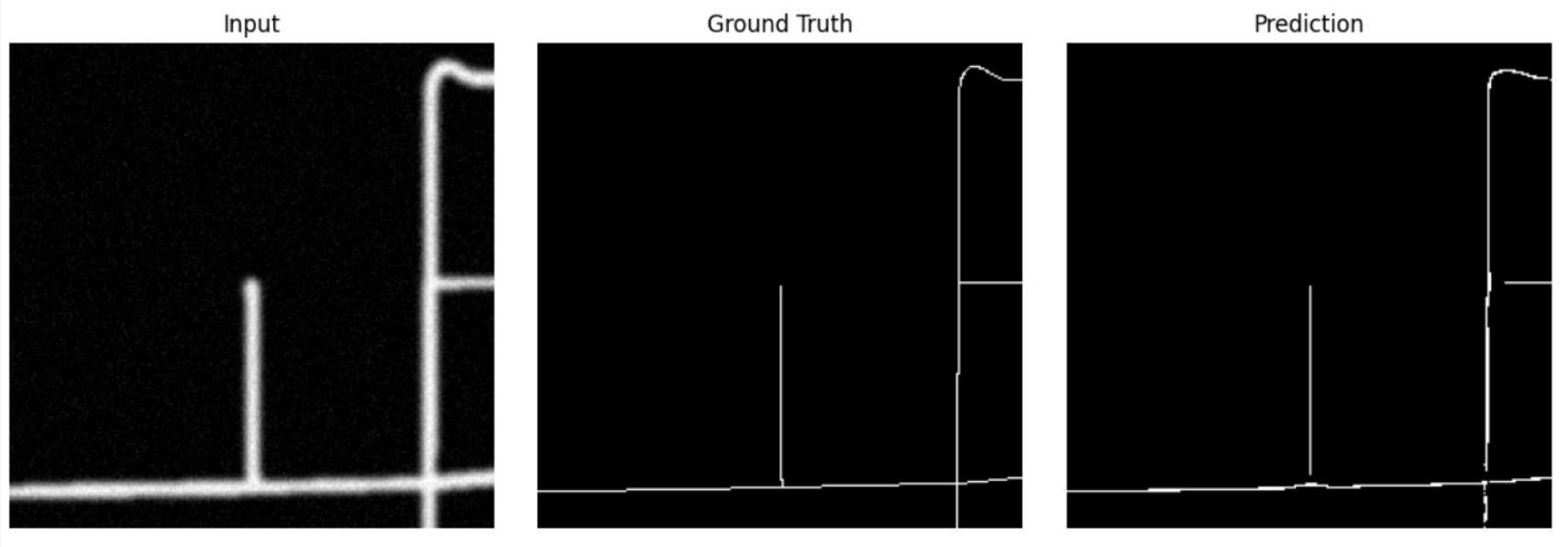
→ Straight run predictions seem visually correct but at some intersections there are some missing pixels



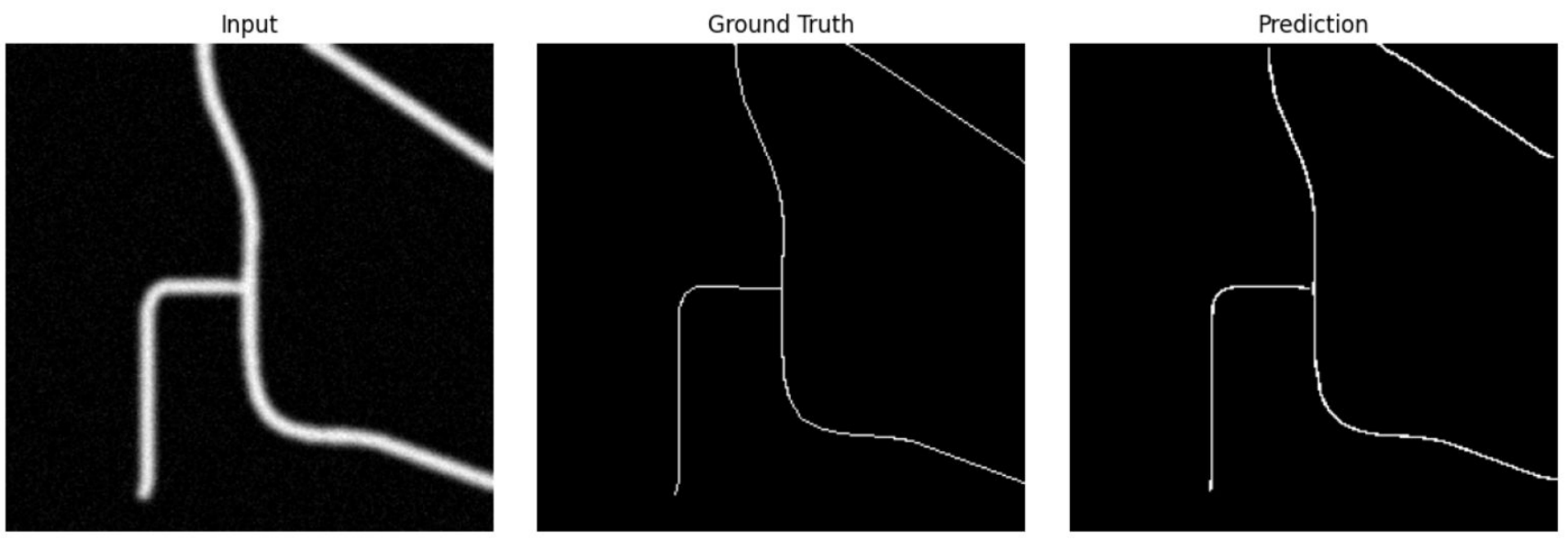
→ Prediction off by some around intersections. Not straight lines more concaved in a “v”.



→ Some concavity in intersections and pixel gapping.



→ Closest prediction so far.



Advanced Model Approach

Same as core architecture as base model approach but added an additional multitask learning setting using distance.

→ Enhancements:

- Multitask Learning by adding a distance transform regression along side the skeleton segmentation
- Model trained to predict both the skeleton mask and the distance transform for each image

→ Model encourages the network to learn more informative features

→ Distance provides the ability for the model to better analyze pixel boundaries and connections

→ Benefits observed.

- Improved segmentation accuracy (higher IoU and Dice scores)
- Better node detection with higher precision and recall for node valences

Advanced Model Approach

Training Loss was still evaluated for Advanced model similar to base model using Tensorboard and intermediate loss outputs.

→ Based on qualitative and quantitative results the best model was Test 7 with Advanced model, stride convolutions, no skip connections, learning rate 1e-3 and combined loss (BCE and Dice).

Data Evaluation

Utilized several suggested data statistics for evaluating each model tested including:

→ Test Loss

→ Mean Squared Error (MSE)

→ IoU

→ Dice Coefficient

→ Node Precision and Recall

Ablation Study

First several slides so the ablation study performed using various parameter changes including the following:

→ Changes in model type. Base model using U-Net structure to Advanced model using multitask distance maps.

Best: Both are good but advanced showcases another way to view predictions.

→ Changes in learning rate for optimizer function. Learning rate changes from 1e-1 , 1e-3 to 1e-5.

Best rate 1e-3.

→ Changes in loss function implementation. Loss function changes from combined BCEwithLogitsLoss and Dice Loss, BCE Loss only, and Dice Loss only.

Best Loss "Combined BCE and Dice".

→ Changes in model architecture from stride convolution in U-net blocks for encoder to maxpooling blocks.

Best: Stride Convolution.

→ Changes in model architecture from skip connected using crop to no skip connections.

Best: No skip connections.

Testing and Verification

Verification:

→ Visual inspections were uses for training loss and validation loss using tensorboard.

→ Intermediate outputs of loss values were printed per epoch for evaluation to ensure decrease and not plateau.

→ I simplified the data and ran only on a few images originally to try to ensure model was reacting appropriately and actually learning. The scaled to more images for training and validation.

Ensure correctness:

→ *Data generation* was visually inspected from base make_data.py output provided and updated make_data_v2.py. Noisy images showed more blur around edges and skeleton models were single pixel. Tqdm was used to see that correct number of images were generated. Data split ensured images were split for clarity in two files ensuring noisy images and ground truth. These images further split into train, validation and test. Ensured image channels were the right shape.

→ **Visual Inspection:** I plotted input images, ground truth masks, and predicted masks at various stages of training to visually check that the model was learning and the outputs were reasonable.

→**Debugging Intermediate Outputs:** I printed shapes and value ranges of tensors at each stage to catch any mismatches or errors early.

The End

CSE627 – Advanced Machine Learning

Kimberly Hamilton