CS 230 – PROJECT FINAL REPORT

Semantic Segmentation of Brachial Plexus Nerve Group on Ultrasound Images

Sibi Shanmugaraj, sibiyes@stanford.edu, SUID – 06407840

Description and Background:

The task at hand is to perform semantic segmentation of a nerve group called the Brachial plexus using ultrasound images. Semantic segmentation has been used in various applications including healthcare. This task drew inspiration from being able to use pain catheters to relieve pain by accessing the origin of the pain. To make that happen, we may need to accurately identify the nerve group we are interested in. This will be the primary motivation for the task at hand. In this particular problem, the images are grey scale ultrasound images which are not high definition. There is minimum variability in the ultrasound images and that might be a major challenge when compared to image segmentation on other areas that involve images with more variability and contrast. Also the images are human annotated and that may bring in possibility of noise.

Dataset and previous approaches:

The dataset for the project is the one obtained from kaggle.com (https://www.kaggle.com/c/ultrasound-nerve-segmentation) and the best model evaluated based on average dice coefficient has a score of 0.73226 on the test set used specifically for evaluation [12].There are a total of 5635 image-mask pairs available and the mask are for regions corresponding to Brachial plexus nerve group provided through human annotation by experts. The provided images are in '.tiff' format. I converted them to '.jpeg' for compatibility with tensorflow and keras libraries. The original images have a width of 580 pixels and a height of 420 pixels. Among the 5635 images in the dataset we have the following.

Number of images with non-positive mask (complete absence of segment of interest): 3312
Percentage of positive mask pixels in images with positive-mask: 2.3 %
Percentage of total positive mask pixels in all images: 0.949%

We can see a sort of class imbalance in the mask area and in several instances, a complete absence of the segment of interest. This is very typical in a medical image dataset. So we need to factor in the possibility of the class imbalance affecting the model performance. For this project a sample from the larger dataset is used for generating training set (N = 500, 750, 1000, 1500) and validation set(N = 250). No further test set was used at this point.

Segmentation problems were attempted using fully convolutional networks where we have a sequence of convolution layers translating the image to the corresponding segmentation mask[2]. However the idea of UNETs for segmentation problems typically involving encoder and decoder blocks wherein the encoder blocks learns the features of interest and the decoder block translates those features to prediction mask have been found very successful[1][3][4]. Keeping the core of the UNET design in place the idea of densely connected convolutional blocks have been used in the UNET design to get good performance[5][6]. The idea of dilated convolutions in place of the traditional convolution layers from semantic image segmentation have been used and found to be successful, especially with the popular DeepLab family of models[7][8][9]. More recently generic image to image translation models (pix2pix) have been built that applies to a family of problems where inputs and outputs are images that are related in someway[10].

Modeling Approach:

The initial experiments were run on a training set of 500 samples randomly sampled from the whole dataset and validation set of 250 examples randomly sampled from the entire dataset is set aside. Later on larger datasets with more training epochs were used for select variants.

MobileNetV2-pix2pix: In this case the down sampling portion is simply made of a pre-learnt MobileNet V2 model. The upsampling portion is composed of 4 pix2pix based upsampling layers. There are 4 skip connections in total, one to each of the upsampling layers from the downsampling portion. The intention is to use some pre-learned architecture in composing a network capable of learning the task at hand. Further training on the problem dataset is carried out. This model variant was trained on images of size 128x128 (for compatibility with the pre-trained architecture)

Unet Models: This variant consists of the U-Net family of models with 4 blocks (128x128 image) or 5 blocks (256x256 image) for downsampling. For upsampling 4 or 5 blocks of 2D conv-transpose layers are used. There are Skip connections between corresponding downsampling and upsampling blocks, similar to a U-Net architecture. There is a block of 2D-convolutions in the middle that connects the downsampling and the upsampling portions.

Multiple variants of the downsampling block are tried for model design for 2 possible image resolutions (128x128 and 256x256). These include the following

- **Unet-Basic:** 2 layers of 2D convolution followed by 2x2 Max Pooling
- **Unet-Skip2:** 2 layers of 2D convolutions with a skip connection followed by 2x2 Max Pooling
- **Unet-Skip3:** 3 layers of 2D convolutions with a skip connection followed by 2x2 Max Pooling
- **Unet-Inc:** Inceptions-Net based block with 2 layers of 3x3 2D convolutions and 2 layers of 5x5 2D convolutions that are concatenated and compressed by 1x1 convolution layer followed by 2x2 Max Pooling.

The network is composed of succession of many of these blocks while increasing the number of channels as we go down the encoder portion. Initially a binary cross-entropy loss function was used to train the models. It was observed that the predicted output masks had values that were always close to 0 due to the class imbalance. To address the issue, a weighted binary cross-entropy loss functions with weights of 10 and 25 were used. This seemed to mitigate the issue observed before. Two different activation functions were used namely relu and elu. A learning rate value of 0.0001 was used. Initial learning rate of 0.001 was tried but later discarded as there were signs of loss divergence. The models take a long time to train (matter of hours) which limited the hyper parameter experiment space by a great deal.

Some of these variants were trained with a batch normalizations before applying activations to study the effect of batch normalizations on the model performance. There was an attempt in using dilated convolutions in the network architecture with the intention of using an larger receptive field. Variants with spatial dropouts where we drop a fraction of the output channels during training epochs is used as a regularization method and the effect of that on the model performance was observed. Lot of variations were tried on the encoder block of the network as described thus far. A few variations in the decoder or the upsampling blocks were also tried to see the effect of those architectural changes on model performance. The initial models described above use 2D convolution transpose layers in the upsampling block. The use of bilinear upsampling layers and pix2pix blocks from pre-trained pix2pix models were used as other option in the upsampling block. 2 extra layers of 2D convolutions follow all the upsampling layers.

All the models were trained for 250 epochs initially. Later on some selected variants were trained on larger training set (750, 1000, 1500 examples) using 500 training epochs. The model evaluation on the aggregate pixel level as well as independent image level detection capabilities have been presented and discussed. Binary crossentropy is a good choice for a loss function for a semantic segmentation problem, but there are other loss functions that apply to this problem space as well. Common among the other choices are dice loss and tverskey loss. These two loss functions were adopted in model building as well. However, all the pixels in the predicted mask turned out to be close to zero. Further investigation is needed to figure out the problem and make this loss function feasible for this problem. One of the recent state of the art model for semantic segmentation is the DeepLab family of models. I attempted to use them on this problem, but couldn't get it running on the compute infrastructure I was able to spin up.

Model Results and Analysis:

The precision, recall, dice coefficient and intersection-over-union (IOU) metrics for the above models on the entire images in the train and the validation set are tabulated below. The precision and recall values computed from them are at an aggregate pixel level on all the images. The dice-coefficient and the IOU values are the averages for image level values. When actual mask is all negative and the predicted mask is all negative, dice-coefficient and IOU values are set to 1 for the example.

Table to the right (Table.1) shows the model evaluation results for the MobileNetV2-pix2pix model built using weighted binary cross-entropy loss function with 2 different weight values.

Table.2 shows the model evaluation results for Unet-Basic model trained of 128x128 images. The models have 4 blocks for encoding and 4 blocks for decoding. Results for model built using weighted binary cross-entropy loss function with 2 different weight values (10 and 25) are provided.

Table.3 shows the model evaluation results for Unet-Basic model trained of 256x256 images. The models have 5 blocks for encoding and 5 blocks for decoding. Results for model built using weighted binary cross-entropy loss function with 2 different weight values (10 and 25) are provided.

Table.4 shows the model evaluation results for Unet-Skip2 model trained of 128x128 images. The models have 4 blocks for encoding and 4 blocks for decoding. Results for model built using weighted binary cross-entropy loss function with 2 different weight values (10 and 25) are provided.

Table.5 shows the model evaluation results for Unet-Skip2 model trained of 256x256 images. The models have 5 blocks for encoding and 5 blocks for decoding. Results for model built using weighted binary cross-entropy loss function with 2 different weight values (10 and 25) are provided.

In the tables provided here, the model evaluation results are presented for the MobileNetV2-pix2pix model, Unet-Basic on 128x128 and 256x256, Unet-Skip2 on 128x128 and 256x256 have been presented. We can see that the MobileNetV2-pix2pix model performs poorly in comparison with the other models, and it is evident from all the metrics namely, precision, recall, dice-coefficient and IOU. The difference in the weights for the loss function doesn't affect the model performance as well. So it is pretty easy to conclude that the model architecture based on MobileNetV2-pix2pix pre-learnt models isn't good enough for the task and there are other model options that turn out to be better.

Now, we can focus our attention on the other model architectures based on UNET design with variations in blocks, activation function, weights used for the weighted binary cross-entropy loss function and the resolution of the input image (128x128 vs 256x256). The variation in the model performance based on the elements described above are minimal across the combinations of these elements. Starting with the weights of the binary cross-entropy loss function, a weight value of 25 results in more false positives (FP) than a weight value of 10 in the Unet-Basic models. This makes intuitive sense as the loss function is tailored to focus more on the positive examples. This manifests itself in decreased precision for weight value of 25 for Unet-Basic models. However on the Unet-Skip2 models, with weight value of 25, there is a decrease in the FP on the validation set resulting in a higher precision. In fact, on the Unet-Skip2 models, we can see a slight decrease in the variance of the model, when the weight value of 25 is used. We need to explore more and take more examples into account before deciding on the trade-off between the bias and the variance.

Model	Dataset	Precision	Recall	Dice	IOU
MobileNet (w = 10)	Train	0.2829	0.1277	0.2398	0.2217
	Validation	0.2834	0.1054	0.2394	0.2254
MobileNet (w = 25)	Train	0.3734	0.0845	0.3627	0.3522
	Validation	0.4275	0.0762	0.3965	0.3888

Table 1: Model Evaluation metrics for MobileNetV2-pix2pix

Model	Dataset	Precision	Recall	Dice	IOU
Unet Basic Relu (w = 10)	Train	0.7391	1.0	0.9375	0.8922
	Validation	0.4542	0.5947	0.5972	0.5504
Unet Basic Relu (w = 25)	Train	0.7174	1	0.9312	0.8827
	Validation	0.4473	0.5619	0.6078	0.5644
Unet Basic elu (w = 10)	Train	0.6597	0.9994	0.9109	0.8566
	Validation	0.4455	0.5682	0.6375	0.5932
Unet Basic elu (w = 25	Train	0.6546	1.0	0.9108	0.8551
	Validation	0.4343	0.6193	0.6459	0.6012

Table 2: Model Evaluation metrics for Unet Basic model on 128x128 image (4 blocks of encoding and 4 blocks of decoding)

Model	Dataset	Precision	Recall	Dice	IOU
Unet Basic Relu (w = 10)	Train	0.71	0.9999	0.9281	0.8783
	Validation	0.4673	0.6208	0.5668	0.4673
Unet Basic Relu (w = 25)	Train	0.6805	1.0	0.9203	0.8673
	Validation	0.4304	0.6046	0.5777	0.5296
Unet Basic elu (w = 10)	Train	0.7175	0.999996	0.9310	0.8824
	Validation	0.4743	0.658960	0.5760	0.5256
Unet Basic elu (w = 25	Train	0.6750	1.0	0.9188	0.8650
	Validation	0.4321	0.6552	0.5669	0.5191

Table 3: Model Evaluation metrics for Unet Basic model on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

Model	Dataset	Precision	Recall	Dice	IOU
Unet Skip2 Relu (w = 10)	Train	0.71323	0.9999	0.9302	0.8813
	Validation	0.46242	0.5875	0.5699	0.5198
Unet Skip2 Relu (w = 25)	Train	0.70190	0.9999	0.9269	0.8766
	Validation	0.48378	0.5796	0.5841	0.5363
Unet Skip2 elu (w = 10)	Train	0.70701	1.0	0.9279	0.8779
	Validation	0.45055	0.5865	0.5075	0.4585
Unet Skip2 elu (w = 25	Train	0.68012	1.0	0.9203	0.8672
	Validation	0.45005	0.58207	0.5708	0.5223

Table 4: Model Evaluation metrics for Unet Skip2 model on 128x128 image (4 blocks of encoding and 4 blocks of decoding)

Model	Dataset	Precision	Recall	Dice	IOU
Unet Skip2 Relu (w = 10)	Train	0.7202	0.9999	0.9279	0.8798
	Validation	0.4693	0.6238	0.5552	0.5067
Unet Skip2 Relu (w = 25)	Train	0.7283	0.9996	0.9345	0.8876
	Validation	0.5089	0.5518	0.5998	0.5525
Unet Skip2 elu (w = 10)	Train	0.7124	0.9999	0.9118	0.8626
	Validation	0.4505	0.6586	0.5410	0.4907
Unet Skip2 elu (w = 25	Train	0.6959	1.0	0.9251	0.8739
	Validation	0.4720	0.6231	0.5684	0.5191

Table 5: Model Evaluation metrics for Unet Skip2 model on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

When we look at the choice of activation functions, there isn't great difference between relu and elu activations. Though the metrics are closer to each other, relu activation gives a better precision and elu activation gives a better recall. The decision on the activation function will boil down to the trade-off between precision and recall. Given that we have a good room for improvement in the model precision on the validation set, it makes sense to focus on relu for the time being. As far as the network designs are considered, the presence of skip connections tend to address the variance problem by a small degree but the difference is minimal. There is still room for improvement when it comes to improving the model variance. The model performance difference between different image resolutions is very minimal at this stage. Computationally, a resolution of 128x128 makes for faster model building, it still feels worth it to use 256x256 on iterations focused on further improvement to keep the possibility of a higher resolution leading to better models open.

Moving further, additional model architectures are tried as described before. Next we have the results for Unet-Skip3 model and Unet-Inc models. To keep it concise, the results are presented only for variants with relu activations though models were built using elu activations as well and a similar trend in terms of the differences described before were observed.

Model	Dataset	Precision	Recall	Dice	IOU
Unet Skip3 relu (w = 10)	Train	0.7175	0.9999	0.9311	0.8825
	Validation	0.5032	0.5705	0.6236	0.5770
Unet Skip3 relu (w = 25)	Train	0.6949	1.0	0.9246	0.8733
	Validation	0.4578	0.5194	0.5828	0.5371

Table 6: Model Evaluation metrics for Unet Skip3 model on 128x128 image (4 blocks of encoding and 4 blocks of decoding)

Model	Dataset	Precision	Recall	Dice	IOU
Unet Skip3 relu (w = 10)	Train	0.7167	1.0	0.9308	0.8822
	Validation	0.46763	0.6132	0.6078	0.5592
Unet Skip3 relu (w = 25)	Train	0.69166	0.9999	0.9197	0.8682
	Validation	0.46248	0.6632	0.5637	0.5140

Table 7: Model Evaluation metrics for Unet Skip3 model on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

Table.6 shows the model evaluation results for Unet-Skip3 model trained of 128x128 images and Table.7 shows the model evaluation results for Unet-Skip3 model trained of 256x256 images.

Table.8 shows the model evaluation results for Unet-Inc model trained of 128x128 images and Table.9 shows the model evaluation results for Unet-Inc model trained of 256x256 images.

Model	Dataset	Precision	Recall	Dice	IOU
Unet-Inc relu (w = 10)	Train	0.6943	1.0	0.9187	0.8676
	Validation	0.4517	0.6600	0.5841	0.5336
Unet-Inc relu (w = 25)	Train	0.6594	1.0	0.9119	0.8565
	Validation	0.4592	0.6554	0.5908	0.5427

Table 8: Model Evaluation metrics for Unet-Inc model on 128x128 image (4 blocks of encoding and 4 blocks of decoding)

Model	Dataset	Precision	Recall	Dice	IOU
Unet-Inc relu (w = 10)	Train	0.7047	1.0	0.9275	0.8774
	Validation	0.4365	0.5751	0.6018	0.5579
Unet-Inc relu (w = 25)	Train	0.6991	0.9999	0.9259	0.8750
	Validation	0.4731	0.6222	0.6062	0.5597

Table 9: Model Evaluation metrics for Unet-Inc model on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

The variation in the model performance between the Unet-skip3 and Unet-skip2 is inconclusive and they are very much similar. The dice-coefficient shows improvement in the case where loss function weight is 10 and shows the opposite or close to no effect, when the weight value is 25. A slight improvement in variance that was observed with Unet-skip2 in comparison with Unet-Basic didn't hold in this variant. Addition of extra layer of 2D convolutions in the encoder block hasn't helped as much. When we observe the results for the Unet-Inc model, we see them to be very close to to the other variants in terms of their performance. In fact there is a small decrease in the precision on the validation set in comparison with the Unet-Basic and Unet-Skip models. Also from a computational cost perspective, the Unet-Inc models are very costly and take much longer to train due to the fact that the network is larger than the others and has many more parameters to train.

Adding Batch Normalization:

The models built thus far didn't have a any sort of batch normalization added. We now add batch normalization and study the effect of that in the model performance. Batch normalization was added to Unet-Basic and Unet-Skip2 models and the results of the model obtained are shown below. Batch normalizations were added only to the encoder blocks.

Model	Dataset	Precision	Recall	Dice	IOU
Unet Basic BN relu (w = 10)	Train	0.7094	0.9999	0.9286	0.8788
	Validation	0.4479	0.6186	0.5389	0.4865
Unet Basic BN relu (w = 25)	Train	0.6732	0.9999	0.9181	0.8640
	Validation	0.3966	0.6212	0.5134	0.4620

Table 10: Model Evaluation metrics for Unet Basic model with batch normalization on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

Model	Dataset	Precision	Recall	Dice	IOU
Unet Skip2 BN relu (w = 10)	Train	0.7181	0.9999	0.9311	0.8824
	Validation	0.4598	0.6542	0.5528	0.5005
Unet Skip2 BN relu (w = 25)	Train	0.6947	1.0	0.9245	0.8729
	Validation	0.4491	0.6654	0.5549	0.5022

Table 11: Model Evaluation metrics for Unet Skip2 model with batch normalization on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

Table.10 shows the model evaluation results for Unet-Basic model with batch normalization trained of 256x256 images and Table.11 shows the model evaluation results for Unet-Skip2 model with batch normalization trained of 256x256 images. On observation we can find that batch normalization has the effect of reducing the precision on the validation set and little to no increase in the recall values. The addition of batch normalization hasn't helped the model performance in the variants attempted.

Adding Dropouts:

Given that we see a variance problem with the models generated thus far, it would be worthwhile to explore possible regularization on the models. Here I have attempted adding spatial regularizations after the convolution layers in the encoder/downsampling blocks. The results for a dropout retention factor of 0.9 is shown below for the Unet-Basic model and the Unet-Skip 2 model with relu activations. By comparing the results with equivalents we have without dropouts (Table 2 and Table 12 for Unet-Basic, Table 4 and Table 13 for Unet-Skip2) we can see that adding dropout leads to drop in the precision in the train and the validation set. Thus the effect of adding dropout is just an increase in the bias of the model without helping address the variance issue.

Model	Dataset	Precision	Recall	Dice	IOU
Unet Basic DO relu (w = 10)	Train	0.6496	0.9999	0.9074	0.8514
	Validation	0.3799	0.5197	0.5429	0.5016
Unet Basic DO relu (w = 25)	Train	0.6219	0.9999	0.8984	0.8393
	Validation	0.3922	0.5145	0.5505	0.5085

Table 12: Model Evaluation metrics for Unet Basic model with dropouts (0.9) on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

Model	Dataset	Precision	Recall	Dice	IOU
Unet Skip2 DO relu (w = 10)	Train	0.6766	0.9999	0.9171	0.8638
	Validation	0.4489	0.5292	0.5994	0.5582
Unet Skip2 DO relu (w = 25)	Train	0.5953	1	0.8862	0.8249
	Validation	0.3931	0.5738	0.5698	0.5252

Table 13: Model Evaluation metrics for Unet Skip2 model with with dropouts (0.9) on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

Adding Dilated Convolutions:

Thus far we have used a traditional 2D-convolutions in our model architecture. Dilated convolutions have been used in deeplab V3 models for image segmentation. Here there is an attempt at using dilated convolutions (with a dilation rate of 2). The dilated convolutions are applied in the first 3 blocks of the encoder and the last 3 blocks of the decoder. Dilated convolution variants are attempted for Unet-Basic and Unet-Skip2 models on 256x256 images and the results are tabulated below. Comparing with the model variants without dilated convolutions, the results are close to same or slightly worse of in terms of all the evaluation metrics presented here for the variants with dilated convolutions.

Model	Dataset	Precision	Recall	Dice	IOU
Unet Basic dil relu (w = 10)	Train	0.7179	0.9999	0.9313	0.8829
	Validation	0.4369	0.5966	0.5963	0.5490
Unet Basic dil relu (w = 25)	Train	0.6359	0.9999	0.8992	0.8415
	Validation	0.4252	0.6393	0.6031	0.5526

Table 14: Model Evaluation metrics for Unet Basic model with dilated convolutions on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

Model	Dataset	Precision	Recall	Dice	IOU
Unet Skip2 dil relu (w = 10)	Train	0.7043	0.9999	0.9249	0.8745
	Validation	0.4473	0.6230	0.5943	0.5463
Unet Skip2 dil relu (w = 25)	Train	0.6804	0.9999	0.9123	0.8591
	Validation	0.4365	0.6218	0.5926	0.5447

Table 15: Model Evaluation metrics for Unet Skip2 model with dilated convolutions on 256x256 image (5 blocks of encoding and 5 blocks of decoding)

Variations in Upsampling:

Until now, the variations in the models presented above have been in the downsampling/encoder blocks. The upsampling or the decoder block structure was pretty much the same with the exception of number of blocks based on the image resolutions used in training the model. All the cases have conv-2D transpose layers for increasing the image resolution of the mask. Two other variation in the upsampling portion were tried out on the Unet-Basic architecture to study if they have any effect on model performance. Th first among the variants is to employ 2D-conv bilinear upsampling layers in place of 2D-conv transpose layers to increase the image resolution (results shown in Table.16). The second variant is to employ upsampling blocks from pix2pix models[10] (results available in Table.17). The number of channels is the same between networks with 2D-conv transpose and 2D-conv upsampling layers where the number of channels decrease by a factor of 2 while the size of the channel increases by a factor of 2. But for the pix2pix blocks the number of channels flatten at 96 while the channel size keeps increasing. This is primarily because the pre-learnt pix2pix blocks have a minimum of 96 channels available. Also this would serve as an opportunity to study whether having more number of channels right up until the output layer could help.

Looking at the tables below that tabulates the performance of the model variants described, we can see that the differing architecture doesn't help the model performance in comparison with the original design employed before. Intuitively it could be reasoned that convolution transpose layer impart proper learning in translating the information while an upsampling layer just performs a resize with very minimal learning in comparison with the transpose layer. For the pix2pix blocks which have 2d-conv transpose layers in them along with batch-normalizarion, the difference is the fact that there is additional batch normalization layer. But that architecture also has more number of channels in final few blocks as specified above. The combination of those result in the performance we see below. Further investigation is needed to figure out which among the two has more reason to cause the drop in performance observed.

Model	Dataset	Precision	Recall	Dice	IOU
Unet Basic US relu (w = 10)	Train	0.6295	0.9989	0.9045	0.8462
	Validation	0.4618	0.5262	0.6196	0.5758
Unet Basic US relu (w = 25)	Train	0.5626	0.9997	0.8757	0.8122
	Validation	0.4275	0.5857	0.6234	0.5765

Table 16: Model Evaluation metrics for Unet Basic model with Upsampling layers in encoding block 256x256 image

Model	Dataset	Precision	Recall	Dice	IOU
Unet Basic p2p relu (w = 10)	Train	0.7241	1.0	0.9331	0.8855
	Validation	0.4664	0.6025	0.5626	0.5148
Unet Basic p2p relu (w = 25)	Train	0.6843	1.0	0.9216	0.8689
	Validation	0.4325	0.6198	0.5298	0.4794

Table 17: Model Evaluation metrics for Unet Basic model with pix2pix blocks in encoding blocks on 256x256 image

Evaluation on image level detection and no detection:

The precision and recall metrics shown before are computed on an aggregate pixel level of all the images in the dataset. It is important to look at the performance on the independent case level. We need to evaluate the model on how good it is in detecting positive cases and not detecting negative cases. In the instances where it detects the presence of positive case, we need to look at how good it is in doing that, especially in medical cases. The next table (Table 18) shows model performance on aggregate image level and shows 4 columns TN (No '+' pixels and None detected), FP (No '+' pixels but some detected), TP (Has '+' pixels and correct ones detected) and FN (Has '+' pixels and None detected)

Model	dataset	No '+' pixels and None detected (TN)	No '+' pixels but detected (FP)	Has '+' pixels and correct ones detected (TP)	Has '+' pixels and None detected (FN)
MobileNetV2	train	91	204	120	86
	validation	49	100	49	53
Unet-Basic 128x128	train	277	18	205	0
	validation	99	50	73	28
Unet-skip2 128x128	train	295	0	205	0
	validation	86	63	85	16
Unet-Basic 256x256	train	295	0	205	0
	validation	96	53	84	17
Unet-Skip2 256x256	train	293	2	205	0
	validation	80	68	86	15
Unet-Skip3 256x256	train	295	0	205	0
	validation	94	55	86	15
Unet-Inc 256x256	train	295	0	205	0
	validation	99	50	79	22
Unet-Basic BN 256x256	train	295	0	205	0
	validation	76	73	90	11
Unet-Skip2 BN 256x256	train	295	0	205	0
	validation	76	73	90	11
Unet-Basic DO 256x256	train	293	2	205	0
	validation	90	59	73	28
Unet-Skip2 DO 256x256	train	294	1	205	0
	validation	102	47	75	26

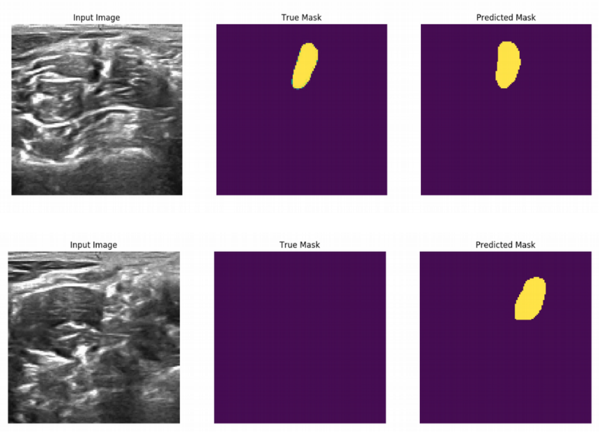
Table 18: Model Evaluation metrics at image level

From the Table. 18 on the left, we can observe that the Unet variant models are better than the MobileNetV2-pip2pix model. Within the Unet variants (relu with w = 10 are shown here), apart from Unet-Basic on 128x128, all others have a perfect response on the train set with FN = 0 (perfect response doesn't mean the predictions are perfect at the pixel level. Just that images with true positive masks have predictions with positive masks in the same neighborhood and images with no positive mask have no mask predicted). On the validation set, we can see that Unet-Skip2 on 128x128 has a better TP value but worse FP value when compared to Unet-Basic on 128x128. Among Unet-Basic, higher resolution images lead to better TP performance. Addition of skip connections in the encoder block does have an effect on the model performance. Between Unet-Skip2 and Unet-Skip3 on 256x256 we can see a better FP performance on Unet-Skip3 with the TP being the same. The Unet-Inc model has a better FP performance in comparison with the others and has the best FP on 256x256 models. The addition of batch normalization has an effect of producing a better TP performance. The models with the batch normalization (Unet-Basic with BN and Unet-Skip2 with BN) have the best TP performance. But that comes at a cost of poor FP performance. Spatial dropouts result in the best TN performance on the validation set, but TP performance is not great in comparison with few other variants.

So if we compare the models at the image level metrics, we can see that the additional convolution layers and skip connections in the encoder block leads to better model performance in comparison to the Unet-Basic. The increase in image resolution does have a positive effect on the model either in terms of better TP or FP based on the model variant. Adding batch normalization gives us a better TP performance and spatial dropouts give a better TN performance.

These variations in the model performance was not evident when we compared the model performance using pixel level metrics. The negative aspects of the model evaluation metrics from the pixel level metrics is mostly due to false positives where the predicted mask has areas outside of the actual mask and false negatives where the mask doesn't full cover the region of interest. In other words, the predictions aren't tight enough. It does help to have a model evaluation at each image level, because in medical situations, it is much more important to detect the presence or absence of specific regions or masses as it is important to accurately identify the region of interest. The aggregate pixel level evaluation tends to lose that information. But an evaluation on independent images helps us learn more about the model performance. With class imbalance observed in this dataset, it makes sense to have 2 ways of analyzing the model results.

One encouraging aspect of Unet model variants is that it predicts a single blob (even on false positive cases) on all but few instances, which is consistent with the nature of the nerve group. The images on the left illustrate that. In image level true positive cases, we can see that the predicted mask tends to have a larger or smaller group of pixels than the actual pixel group or in a slightly distorted shape, but the predicted blob is more or less localized to the same region as the '+' pixels in the true mask.



We can now see that model needs improvement in 2 main areas. First is to address the variance issue and the next is to improve the image level accuracy.

Effect of Dataset size on model performance:

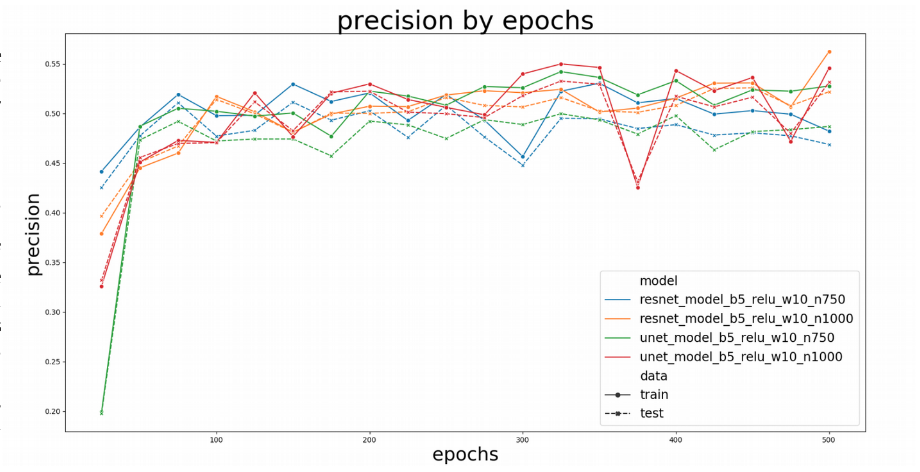
The models built so far have been on a training set of 500 image-mask pairs. We observed the variance issue associated with the models obtained that is presented by a significant difference in the evaluation metrics between the train dataset and the validation dataset. One way to address the variance issue is to bring in more data into the training set. Here we have attempted the same by building models on increasing sizes of training set. Unet-Basic and Unet-Skip2 models are trained on a training set with 750, 1000 and 1500 examples in addition to the original version that was trained on 500 examples. The same validation set is used in all the cases. The initial model on 500 examples was trained for 250 epochs and the others are trained for 500 epochs. We can see that the variance issue is being addressed with more data as the gap between the model evaluation metrics between the train dataset and the validation dataset is reduced significantly. But the performance on the train set drops from before. So we now have a higher bias problem while the variance issue is mitigated a bit. But the positive aspect is that the model performance on the common test set has improved with larger training dataset. This can be seen from the results tabulated in Table 19. The performance on the validation set shows improvement with N = 750 and N = 1000 in comparison with N = 500. There is slight dip between N = 1000 and N = 1500 on the validation set. This could possibly be due to not enough training epochs for the increased number of training examples to get the best loss performance. But Further inspection is needed to confirm the fact. The recall value on the training set has dropped for N = 750, 1000 and 1500 from a value close to 1.0 for N = 500. For the validation set the recall values fluctuate a bit but look to be around 0.60.

Model	dataset	Precision	Recall	Dice	IOU
Unet-Basic (w=10)	train	0.7391	1.0	0.9375	0.8922
N = 500	validation	0.4542	0.5947	0.5972	0.5504
Unet-Basic (w=10)	train	0.5276	0.6325	0.6356	0.5891
N = 750	validation	0.4868	0.6379	0.5828	0.5381
Unet-Basic (w=10)	train	0.5457	0.6402	0.6701	0.6277
N = 1000	validation	0.5313	0.6068	0.6754	0.6316
Unet-Basic (w=10)	train	0.5923	0.6849	0.7510	0.7073
N = 1500	validation	0.5197	0.6227	0.6830	0.6416
Unet-Skip2 (w=10)	train	0.7132	0.9999	0.9302	0.8813
N = 500	validation	0.4624	0.5875	0.5699	0.5198
Unet-Skip2 (w=10)	train	0.4821	0.6141	0.5916	0.5455
N = 750	validation	0.4687	0.6530	0.6002	0.5519
Unet-Skip2 (w=10)	train	0.5623	0.6186	0.6773	0.6378
N = 1000	validation	0.5217	0.5655	0.6820	0.6402
Unet-Skip2 (w=10)	train	0.5960	0.6730	0.7435	0.7014
N = 1500	validation	0.5133	0.5954	0.6814	0.6391

Table 19: Model performance by training set size

Training epochs and model convergence:

On the plot to the right, we have the precision of the models on the training set and the validations set by training epochs. The results for Unet-Basic and Unet-Skip2 model are shown in here. The bold lines are for training set and the dashed lines are for the validation dataset. Different models are color coded. The metrics are computed for various epochs at an interval of 25. We can see that by epoch 50, the curve starts to flatten. This tells us that much of the learning is achieved at about 50 to 75 epochs. The results we have are for models trained using 750 and 1000 training examples. A similar flattening was observed for other model evaluation metrics. This implies that the models learn the basic concepts in the problem space relatively quickly, and it is challenging to learn the finer aspects to be able to come up with high precision outputs.



Summary & Recap:

We explored the effect of block variations in a UNET design and how they affected the model performance. The effect of the image resolutions, loss function weights, activation functions, batch-normalizations and dropout were explored. We showed how evaluating models at the pixel level and the image level help us better understand the model effectiveness. We then explored the effect of training set size on model performance, more importantly as it pertains to model bias and variance. One of the main challenges with this problem, as with several other semantic segmentation tasks is the computational complexity. The models take a long time to train and hence these model variants for all the experiment parameters were run once for one sample of training and validation dataset. Multiple runs on different data samples will help us get more significant results. Also we can increase the training dataset size to a higher value and use that to repeat all the experiments. The positives of having a larger training set were demonstrated. Though other loss functions such as dice loss and tverskey loss were attempted, the outcomes weren't successful. We can explore the issues further and try out those loss functions as well to study their effectiveness. DeepLabV3 model was not successfully implemented for reasons pertaining to compatible computational resources. We can try them again with a more advanced compute environment.

Future steps:

Given the work done thus far and the experiments conducted, I find that we can do the following as the next steps in this work

- Get the model working on dice loss and tverskey loss functions and compare them with the performance obtained by using binary crossentropy loss functions.
- Explore the state of the art pre learnt models for segmentation such as DeepLabV3.
- Explore the effects of data augmentation on getting an improved model fit.
- Dense convolution blocks in the UNET architecture could be explored.
- Build models that are deeper than the ones attempted thus far.
- Use larger resolution images and larger training dataset (though these will be computationally more expensive)

Code Repo: https://github.com/sibiyes/nerve_segmentation

References:

[1] https://www.jeremyjordan.me/semantic-segmentation/
[2] Long et.al. *Fully Convolutional Networks for Semantic Segmentation (https://arxiv.org/pdf/1411.4038.pdf)*
[3] Ronneberger et.al. *U-Net: Convolutional Networks for Biomedical Image Segmentation (https://arxiv.org/pdf/1505.04597.pdf)*
[4] Drozdzal et.al. *The Importance of Skip Connections in Biomedical Image Segmentation (https://arxiv.org/pdf/1608.04117.pdf)*.
[5] Jegou et al. *The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation (*https://arxiv.org/pdf/1611.09326.pdf*)*
[6] Huang et al. *Densely Connected Convolutional Networks (https://arxiv.org/pdf/1608.06993.pdf)*
[7] Chen et al. *DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs (*https://arxiv.org/pdf/1606.00915.pdf*)*
[8] Chen et al. *Rethinking Atrous Convolution for Semantic Image Segmentation (*https://arxiv.org/pdf/1706.05587.pdf*)*
[9] Yu et al. *Multi-Scale Context Aggregation by Dilated Convolutions (*https://arxiv.org/pdf/1511.07122.pdf*)*
[10] Isola et al. Image-to-Image Translation with Conditional Adversarial Networks (https://arxiv.org/pdf/1611.07004.pdf)
[11] https://lars76.github.io/neural-networks/object-detection/losses-for-segmentation/
[12] https://www.kaggle.com/c/ultrasound-nerve-segmentation/leaderboard
[13] https://github.com/tensorflow/examples/tree/master/tensorflow_examples/models/pix2pix
[14] https://github.com/MLearing/Keras-Deeplab-v3-plus