COL341

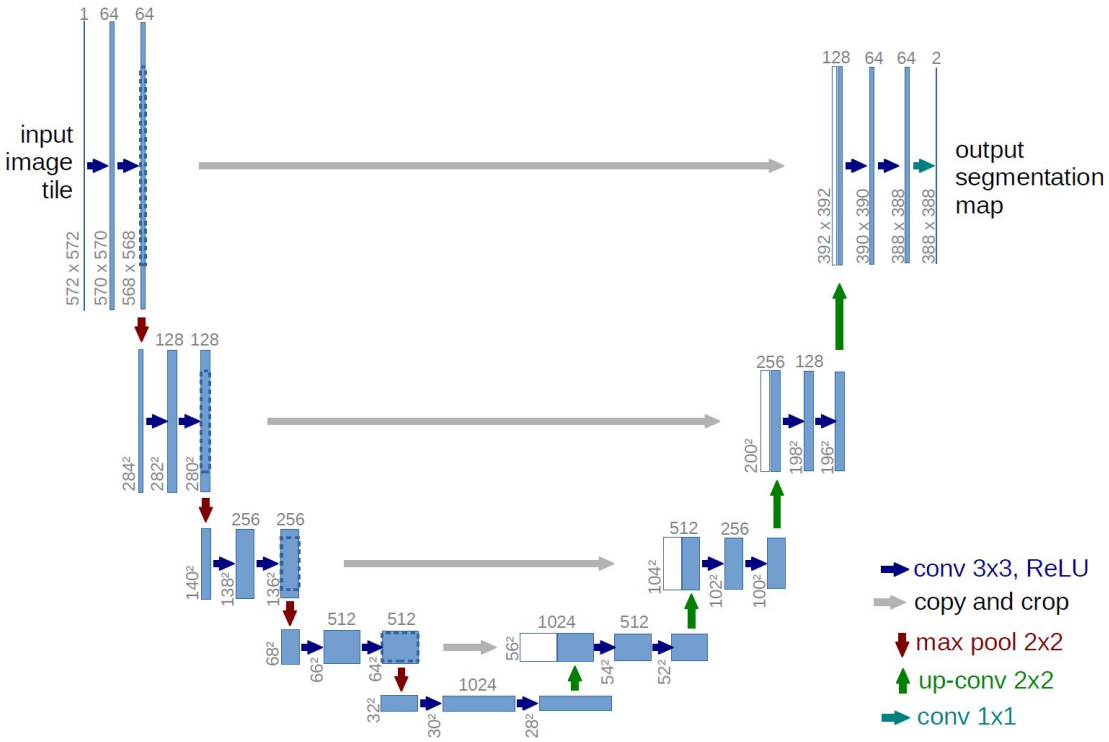
Assignment 3- TGS Salt Classification

Prabhat Kanaujia
2016CS50789
Nishant Verma
2016EE10471

Initial Architecture:

UNet was the starting point for this assignment. It is comprised of 3 parts:

1. Downsampling Path: to extract coarse contextual info
2. Bottleneck: 2 simple convolutional layers
3. Upsampling path: to enable precise localization



We used a special loss function - "**Lovasz Loss"**, which is able to give better accuracy than standard loss functions. It is explained in detail, later in this report.

I used Keras(tensorflow-gpu backend) to build and train this model, which gave an accuracy of 82.7%.

This model is borrowed from a Kaggle kernel- https://www.kaggle.com/abhilashawasthi/unet-with-simple-resnet-blocks

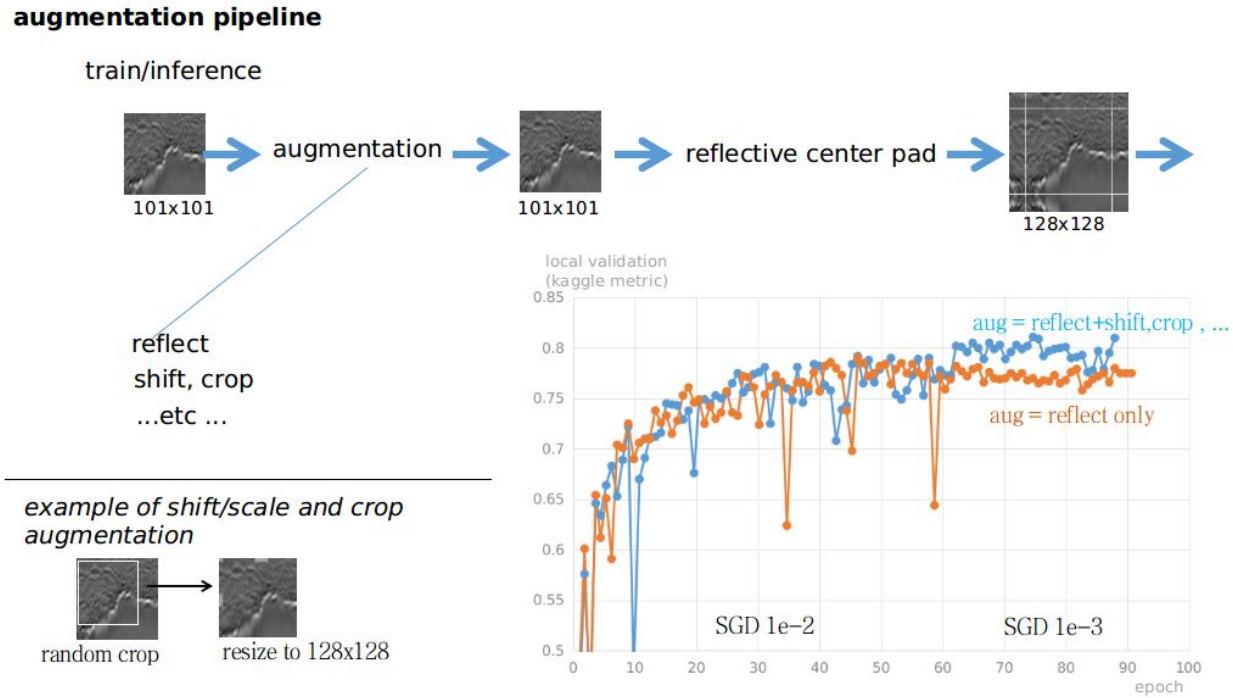
We tried several possible improvements, discussed in various forums(Kaggle and otherwise) and implemented and ran the code for these. Unfortunately, the kernels submitted by various kagglers were already optimized to the peak and most of our efforts, though theoretically correct(and even practically in some examples), didn't show any significant improvements.

Thus, our final model is only a slight improvement over the one mentioned in the kernel above.

Experimented Improvements:

A. **Augmentation:** We tried several augmentation techniques to increase the number of training images and also increase the variance in these images so that the algorithm can learn stronger parameters.
   Some of the techniques we tried were:
   1. **Reflective Centre Pad:** This technique was used to add reflective boundaries to 2D images, so as to convert them from 101*101 to 128*128 in size.
   2. **Random Crop and Resize:** The images obtained from the above step were either used as it is as the training data. Or they

were shifted and/or randomly cropped and then resized again to 128*128 size and then used as training data.

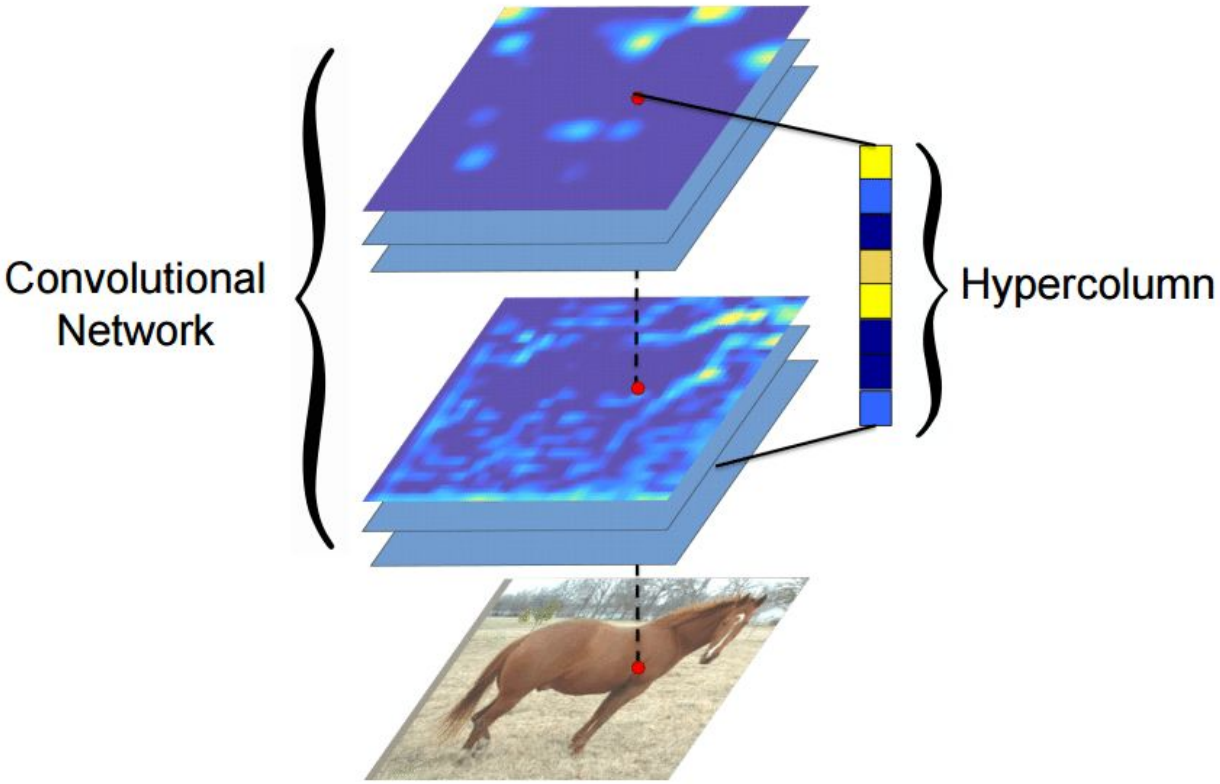


The above image illustrates the augmentation and improvement in accuracy for a certain implementation.

B. Convolutional Hypercolumn: hypercolumn of a pixel is the vector of activations of all CNN units above that pixel. This was a pretty complex feature extraction technique and since we came to know about it towards the end of the deadline, we don't fully understand it. The results are highly promising, according to both independent blogs as well as the original research paper. Our results didn't turn out to be as good as promised, possibly due to faulty implementation of the technique.

Many algorithms using features from CNN usually use the last fully-connected layer features in order to extract information about certain input. However, the information in the last FC layer may be too coarse spatially to allow precise localization (due to sequences of max pooling, etc.), on the other side, the first layers may be spatially

precise but will lack semantic information. Hypercolumn aims to address this issue, and theoretically, it gives significant improvement. The image on the next page gives an illustration of what a hypercolumn is.



C. Lovasz Loss: We opted Jaccard loss for the evaluation of binary image segmentation masks(where each pixel of a given image has to be classified into either a background or a foreground class) which is also known as Intersection-over-union loss because of its better perceptual quality and scale invariance which leads in capturing of very fine details with per pixel losses. Then for the optimization of the per-image intersection-over-union loss, Lovasz hinge method is used which is based on a convex surrogate. This loss performs better with respect to the Jaccard index measure as compared to other losses used for semantic segmentation such as cross-entropy and lovasz hinge is reliably faster and efficient optimization method than other alternatives.

Jaccard index for predicted label A and actual label B is given by the expression given below:

$$J(A, B) = |A \cap B| / |A \cup B|$$

With a convention that J(0,0) = 0

And the loss function which is used for empirical risk minimization is given by:

$$\Delta_J(A, B) = 1 - J(A, B)$$

REFERENCES:

1. https://www.kaggle.com/abhilashawasthi/unet-with-simple-resnet-blocks
2. https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/63974
3. https://www.kaggle.com/c/tgs-salt-identification-challenge/discussion/65933
4. https://github.com/bermanmaxim/LovaszSoftmax
5. http://openaccess.thecvf.com/content_cvpr_2018/papers/Berman_The_LovaSz-Softmax_Loss_CVPR_2018_paper.pdf
6. http://blog.christianperone.com/2016/01/convolutional-hypercolumns-in-python/
7. https://arxiv.org/pdf/1411.5752v2.pdf
8. https://www.groundai.com/project/optimization-of-the-jaccard-index-for-image-segmentation-with-the-lovasz-hinge/