Classification and Segmentation of Pneumothorax from Chest X-rays using Deep Convolutional Neural Networks

Arman Haghanifar

December 2020

1 Baseline Architecture: U-Net

U-Net is initially introduced by Ronneberger *et al.* [1] in 2015 for segmentation of biomedical images. U-Net is a u-shaped Convolutional Neural Network (CNN) which is an enhancement of Fully Convolutional Networks (FCN). FCNs are introduced for semantic segmentation earlier in 2015 [2], and U-Net-based architectures hit better results when dealing with smaller datasets. Thus, U-Nets have been successfully applied for medical segmentation tasks. There have been many applications of U-Net in biomedical image segmentation using various imaging modals, such as brain tumor detection and segmentation in Magnetic Resonance Imaging (MRI) [3], liver vessel extraction from Computed Tomography (CT) images [4], pediatric hand bone segmentation in x-ray images [5], etc.

U-Nets have also been utilized for accomplishing different segmentation-based tasks using Chest X-Ray (CXR) images. Original U-Net is used to segment different anatomical structures in a CXR, such as lungs and heart [6]. Attention U-Net is employed for detection of chest lobes from the CXR image, which is trained on multiple datasets of CXRs with annotation masks [7]. The most recent U-Net-based architecture used for lung segmentation in CXR images is Dual Encoder Fusion U-Net (DEFU-Net), hitting the Dice Similarity Coefficient (DSC) score of 0.9667 on the test-set [8]. Most U-Net architectures are developed for lung segmentation of CXRs, due to the availability of a number of standardized datasets with annotations. With the introduction of SIIM-ACR dataset of CXRs with Pneumothorax annotation masks [1], several research studies are conducted to localize pneumothorax using neural networks. Among introduced networks, U-Net-based architectures are popular with the most recent development of 2-Stage Training U-Net (2ST-UNet) which achieved a DSC score of 0.8356 [9].

By the development of several more complicated U-net-based architectures, the initial one proposed by the authors of the U-Net paper is now considered as the vanilla U-Net. Vanilla U-nets are created by adding a couple of convolutional layers as the encoder, followed by the same number of up-convolution layers as the decoder. The difference between U-Net and simple FCN is the concatenations between each convolutional layer in encoder, with its counterpart up-convolution layer in decoder. The baseline architecture for pneumothorax segmentation is shown in Fig. 1.

[1] https://www.kaggle.com/c/siim-acr-pneumothorax-segmentation/

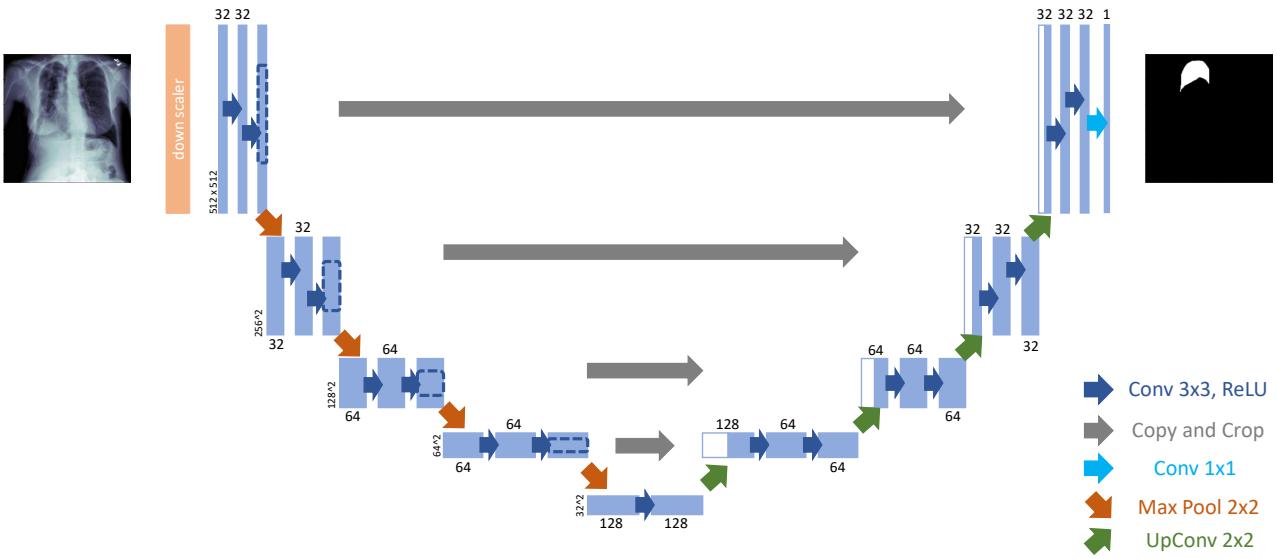

Figure 1: Architecture of a custom vanilla u-net for pneumothorax segmentation

In each block, two convolution layers are followed by a max-pooling layer to reduce the input size. Activation function of convolution layer is set as Rectified Linear Unit (ReLU), which enhances the model performance by adding non-linearity to the network. The final layer is a convolutional layer with 1×1 kernel to merge the information of different channels. In this case, since images are grey-scale, different channels are different preprocessing algorithms applied on the raw CXR image. These preprocessing algorithms are mostly based on histogram equalization, e.g. AHE, CLAHE, BEASF, etc. Initial results of the Vanilla U-Net after 30 epochs is illustrated in Fig. 2.

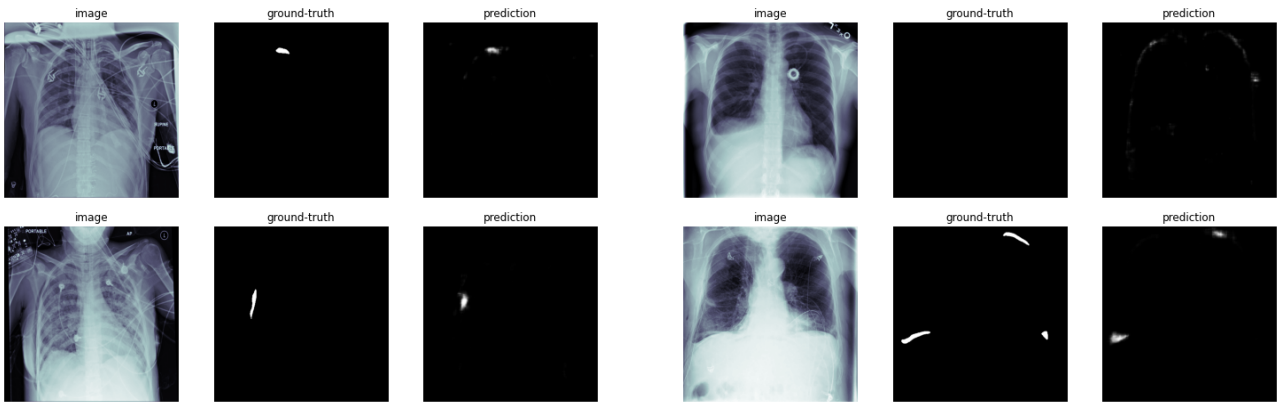

Figure 2: Vanilla U-Net results on sample images

These results are achieved without any regularization themes, i.e. no data augmentation or dropout layers. The custom Vanilla U-Net architecture has approximately $790,000$ all-trainable parameters, and is trained for 30 epochs.

2 Evaluation Details

Validation: Application of deep learning models in the field of biomedical image processing has several challenges among which the lack of sufficient data is one of the main barriers toward developing biomedical decision systems [10]. Deep neural networks require large training datasets to achieve acceptable accuracy scores, while they also need a good number of test datasets to make sure they are performing generalized enough. Thus, test ratio for splitting dataset into training and test sets must be carefully selected. In this study, there are 12,047 images in the dataset which must be split into training and validation sets. Due to the fact that the test set consists of 3205 images and is hosted by Kaggle as private data, cross-validation must be employed to benefit the most from the available data. Since the private test set includes approximately 21% of the total dataset, a validation ratio of 20% is selected. Train-test-split is performed manually by selecting random indices of the images in the dataset and their corresponding indices in the masks. Different cross-validation techniques are used for dealing with medical image datasets. Despite the fact that leave-one-out cross-validation (LOOCV) is typically used when working with small datasets, K-Fold cross-validation is selected as there are sufficient number of images. Number of folds must be limited (e.g. 5-Fold) because training the model takes significant time; several hours for a dozen of epochs.

Augmentation: A decisive factor for whether or not to use image augmentation is the number of images in each class. Since there are only 2669 CXRs associated with Pneumothorax annotation masks, image augmentation must be employed with as many combinations of different algorithms as possible. Data augmentation is available under the Keras library as ImageDataGenerator module [2]. There are a couple of other data augmentation libraries available, such as Albumentation [3] and ImgAug [4], each of which has its own advantages to be used in a project. In this project, Keras data augmentation library is used to get initial results with different models. Later to boost the performance of the model, it could be a good idea to change the augmenter method to use Albumentation library as it has advanced image manipulation algorithms. Currently image augmenter is a combination of rotation, horizontal/vertical flip, zoom, brightness change, and width/height shift algorithms. Rotation range is selected as 170 degrees. Zoom range is between 0.9 and 1.5, because large zooming may exclude the region of interest from the image. Brightness adjustment range is between 0.7 and 1.2, since too much brightening or darkening may result in information loss. Width/height shift range is selected as 0.15 to make sure we have the main frame of the chest inside the augmented image. Other augmentations, such as adding noise or shearing, are to be added in the next steps.

Loss Function: Since our model is expected to create black and white masks for each input, the desired output is a binary decision per each pixel. Thus, the model is behaving similar to a binary classifier to classify each pixel between two classes of 0 and 1. A very straightforward and common loss function for binary classification is the binary cross-entropy or log loss. Binary cross-entropy is a function to calculate the cross-entropy between the predicted classes and the true classes. It results in a prediction value between 0 and 1 for each pixel. Binary cross-entropy is formulated as below.

$$H_p(q) = \frac{-1}{N} \sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

where y is the label (1 for white pixel and 0 for black pixel) and $p(y)$ is the predicted probability

[2] https://keras.io/api/preprocessing/image/
[3] https://albumentations.ai/
[4] https://imgaug.readthedocs.io/

of the pixel being white for all N pixels. In the next steps, binary cross-entropy can be combined with the dice loss to build a better loss function for dealing with imbalance datasets. Dice loss is originated from the initial dice coefficient, and is brought to computer vision community by Milletari *et al.* in 2016 for 3D medical image segmentation [11].

Metrics: While performance of the classifiers are usually expected to be measured via accuracy score, in segmentation tasks, dice coefficient is the routine. To better monitor the functioning of the models, Dice Similarity Coefficient (DSC) and Intersection over Union (IoU) are defined as the metrics.

3 Model Tuning

Hyperparameters: In the baseline model, a set of convolution, deconvolution, and pooling layers are used. The activation function of the middle layers is set to ReLU, which is considered the best option for convolution layers experimentally. Worth mentioning that the activation function of the last layer is set to Sigmoid function. To maintain the same image size after each convolution, padding is set to make the output image have the same size as the input image. Aside from aforementioned preset parameters, there are a number of other hyperparameters/network features that can be set during the training stages:

- Image Input Size: Due to the memory limits, we are unable to use the images with their original size of 1024×1024. Thus, images are initially downsized to 512×512, which can also become smaller while minimizing the information loss.
- Architecture: Vanilla U-Net is a set of 4-5 convolutional blocks accompanied with the same number of deconvolutional blocks. The number of layers and the order of them can be changed to get better results.
- Number of Epochs: After each epoch, model has seen all the training data once. More epoch means better performance, while overfitting may reverse the results after a certain number of epochs. Early-stopping criteria, which will be discussed later, effectively sets maximum number of epochs.
- Batch Size: In each step of an epoch, a random batch of images fed to the model to use the outputs for back-propagation. The batch size is usually set as a power of 2, to accelerate the training process in terms of memory access limits. Higher batch sizes result in better scores, while memory limits prevents using very high sizes. Highest batch size with no memory crashes is considered as the best choice.
- Number of Filters: Usually defined as a power of 2. While we get deeper in the convolutional sequence, it is expected to increase the number of filters in each layer. Number of filters can be set from 32 to 1024 as seen in experimental efforts, with respect to the final number of parameters and overfitting as the bottleneck.
- Kernel Size: Usually expected to be (3×3). While increasing the kernel size, network is forced to search for bigger patterns in the image. Kernel size is a hyperparameter that can have noticeable effect in the model performance.
- Dilation Rate: Normally set to (1×1), which is the same as a normal kernel. Increasing the dilation rate increases the distance between kernel arrays. Hence, model is forced to search for larger patterns, while there could be information loss.

- Kernel Stride: Another hyperparameter that must be set with regards to the kernel size and the padding, to avoid missing information while convolving the image.
- Kernel Initializer: Initializers define the method to set the initial random weights of a network layer. Default initializer for Keras convolution layer is the Glorot Uniform, also called Xavier Uniform initializer.

Regularization: Any modification made to the learning algorithm which is intended to reduce the generalization error while maintaining its training error is called regularization [12]. Regularization is an essential key concept in preventing the model from overfitting on the training data. Regularization techniques used in the current study is:

- Data Augmentation: Mentioned earlier, it helps the model to see more training data by applying several operations on the existing images in the dataset. In medical datasets, where there are usually limited number of data, augmentation significantly helps enhancing the quality of the model.
- Early Stopping Criteria: During the training process, model's loss value decreases on both training and validation sets at first. In some point, the performance on the validation set gets worse, while training loss still decreases. Early-stopping is a set of criteria defined to interrupt the training procedure before overfitting occurs. Early-stopping itself effectively tunes the number of training epochs which is an important hyperparameter. When overfitting occurs after dozens of epochs, model might start learning the noise in the training data. In this case, training error will continue to decrease, while test error (generalization) goes up. Early-stopping is to find the best epoch to stop training with minimum test error.
- Dropout layer: While larger architectures are to be used by expanding U-Net as the baseline, number of parameters drastically will be increased. To better deal with early overfitting of the network, dropout layers are used after each main layer to randomly remove some nodes along with all their incoming/outgoing connections. A basic illustration of dropout effect is shown in Fig. 3.

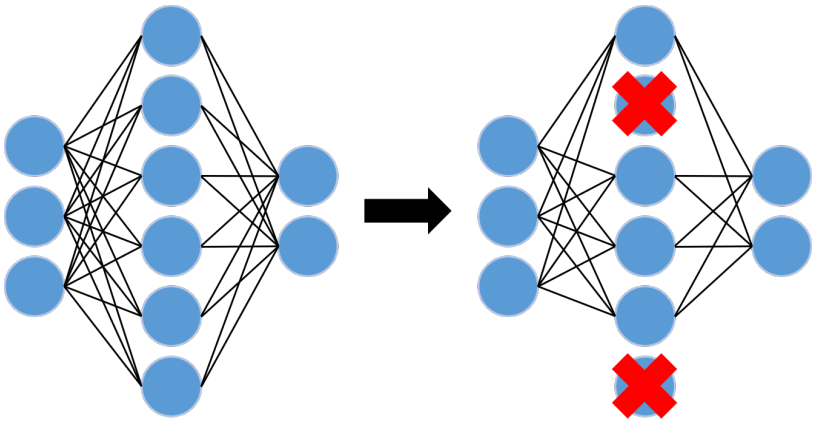

Figure 3: Removing a number of neurons in a hidden layer by applying dropout. In this case, dropout ratio is 0.34 for the layer in the middle and *None* for other layers.

Dropout ratio defines the percentage of randomly selected neurons to be excluded from learning procedure in a specific epoch. Normally, dropout ratio is lower for input layers to prevent underfitting. Dropout ratio is another hyperparameter which needs to be set experimentally.

- Weight Penalty (Kernel Regularizers): Weight penalty is the standard method for regularization, commonly used for training various models. It is based on the implicit assumption that model with small weights is simpler than a model with large weight values. Penalties, also named as weight decay, try to keep the weights small or zero unless there are big counteracting gradients support weight value increase. Kernel regularizers can be either $L1-norm$, $L2-norm$, or set to $None$. While $L2-norm$ tries to keep all weight at small values, $L1-norm$ tends to drive some weights exactly to zero, and keep others at big values.

4 Other Architectures

U-Net-based architectures are dominant players when designing deep neural networks for image segmentation tasks. Backbone, or encoder, part of the U-Net architecture can be manually designed with a set of convolution layers. Reference networks with ImageNet-based weights are usually used as a transfer learning method in roughly all deep learning models. Here, we can use ResNet, DenseNet, or perhaps EfficientNet architectures with pre-trained weights as the encoder for our U-Net architecture. To better utilize transfer learning for Pneumothorax segmentation challenge, we may use a network trained on similar type of images as the backbone; the CheXNet. CheXNet is a DenseNet-121 model trained on more than $100,000$ CXRs for hundreds of epochs, to classify thoracic diseases into 14 different categories, such as Pneumonia and Pneumothorax [13].

Another method is to simply use FCNs, which contain only convolutional layers trained end-to-end for image segmentation. Simple FCNs are not expected to result in better results than the U-Nets. Other possible architectures are Mask R-CNN, Feature Pyramid Network (FPN) and its variants, Different versions of DeepLab, Path Aggregation Network (PANet), Context Encoding Network (EncNet), etc. Among these architectures, only DeepLab V3 and Mask R-CNN are going to be addressed for comparison with U-Net-based developed architecture.

References

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[2] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[3] H. Dong, G. Yang, F. Liu, Y. Mo, and Y. Guo, "Automatic brain tumor detection and segmentation using u-net based fully convolutional networks," in *annual conference on medical image understanding and analysis*. Springer, 2017, pp. 506–517.

[4] Q. Huang, J. Sun, H. Ding, X. Wang, and G. Wang, "Robust liver vessel extraction using 3d u-net with variant dice loss function," *Computers in biology and medicine*, vol. 101, pp. 153–162, 2018.

[5] L. Ding, K. Zhao, X. Zhang, X. Wang, and J. Zhang, "A lightweight u-net architecture multi-scale convolutional network for pediatric hand bone segmentation in x-ray image," *IEEE Access*, vol. 7, pp. 68 436–68 445, 2019.

[6] M. Frid-Adar, A. Ben-Cohen, R. Amer, and H. Greenspan, "Improving the segmentation of anatomical structures in chest radiographs using u-net with an imagenet pre-trained encoder," in *Image Analysis for Moving Organ, Breast, and Thoracic Images*. Springer, 2018, pp. 159–168.

[7] G. Gaál, B. Maga, and A. Lukács, "Attention u-net based adversarial architectures for chest x-ray lung segmentation," *arXiv preprint arXiv:2003.10304*, 2020.

[8] L. Zhang, A. Liu, J. Xiao, and P. Taylor, "Dual encoder fusion u-net (defu-net) for cross-manufacturer chest x-ray segmentation," *arXiv preprint arXiv:2009.10608*, 2020.

[9] A. Abedalla, M. Abdullah, M. Al-Ayyoub, and E. Benkhelifa, "The 2st-unet for pneumothorax segmentation in chest x-rays using resnet34 as a backbone for u-net," *arXiv preprint arXiv:2009.02805*, 2020.

[10] M. I. Razzak, S. Naz, and A. Zaib, "Deep learning for medical image processing: Overview, challenges and the future," in *Classification in BioApps*. Springer, 2018, pp. 323–350.

[11] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 565–571.

[12] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1, no. 2.

[13] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya *et al.*, "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning," *arXiv preprint arXiv:1711.05225*, 2017.