The Search for the Best U-Net Architecture:

A Neural Architecture Search Implementation for Semantic Segmentation

Capstone Project: Haley Massa

Abstract:

Since the introduction of the U-Net in 2015, the deep learning architecture has become the standard for medical imaging segmentation tasks. Due to this popularity, researchers have devoted a lot of resources to developing and improving the auto-encoder-decoder framework of the original U-Net architecture. However, with many research applications focusing on improving small percentages of accuracy, it can be challenging for non-deep learning experts to decide what improvements to add or adjust within their models. This work proposes a solution to that problem through a Neural Architecture Search framework, implemented with the Keras-Tuner, to evaluate the best U-Net parameter selection for a specific dataset. This project also shows promising results for the tune-able U-Net structure on three different medical imaging semantic segmentation datasets and compares the best parameters chosen. The work is, however, limited to the GPU memory resources available, and therefore provides insight for future research with larger search space for models.

Introduction:

In recent years, deep learning has become more common in medical imaging research, with a large portion of publications focused on applying convolutional neural networks to classify, segment, and synthesize medical images. The most popular deep learning method within medical imaging is the convolutional U-Net architecture [1]. Since the introduction of the U-Net by Ronneberger in 2015, it has become the standard for medical imaging applications and overall image segmentation.

The U-Net architecture received its name because of its "U" shape. It is an auto-encoder-decoder network, so the first half of the network works to extract the essential features of the input image into a single array. The second half of the model takes this information and uses a decoder network to rebuild the target information from these features. The simplicity of this format, as well as shown success of the model, has inspired many modifications to this network for medical imaging tasks [2], [3].

These modifications can change everything from the depth of the U-Net to the actual backbone encoder model used to expose essential features [4]. Another way to view these modifications is as changing hyperparameters associated with the model. Hyperparameters are modifications that are external to the model and not chosen by data, so they are explicitly selected by the researchers.

As researchers develop more adaptations, deep learning experts will be needed to decide which hyperparameters will be beneficial to incorporate within the U-Net architecture. Additionally, to achieve a state-of-the-art result, you need not only a deep learning expert but a domain expert in whatever dataset you are using. It can be challenging for non-deep learning or domain experts to decide which hyperparameters to use or adjust when building a new model.

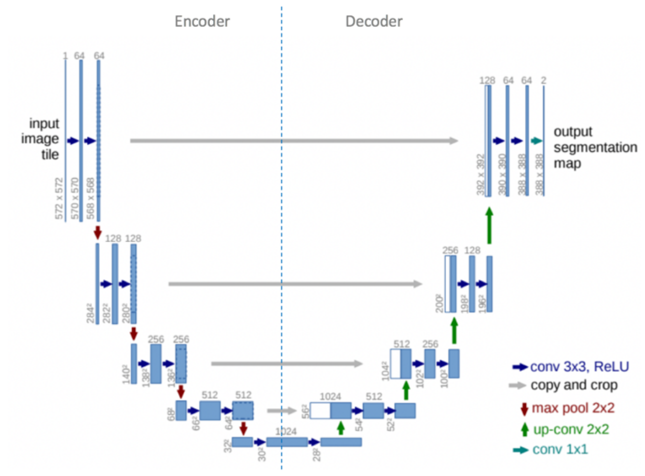

Figure 1: The original U-Net architecture as defined by Ronneberger [1]. This work aims to modify this original architecture, while maintaining the overall architecture shape.

This work aims to solve that problem by creating an easily implementable framework to find the best U-Net architecture that requires no expert analysis or preprocessing. This framework was developed using Neural Architecture Search (NAS) methods [5] and implemented in TensorFlow with the newly released Keras-Tuner. Within this NAS method, eight hyperparameters are searched, and the best configuration is returned based on our specified loss function. Three medical imaging datasets, representing different challenges within medical imaging, were used to support the claim that the framework can improve upon a baseline U-Net model. While these datasets show promising results, this project was limited by the GPU allowance and memory of available virtual machine instances. Therefore, the results found were also limited by these circumstances. The framework for evaluation, however, will be incorporated by the Department of Radiology at UW-Madison, where they will be able to scale up the tuning with in-house (non-remote) larger memory GPUs.

2. Related Work

As research continues to create methods to improve deep learning models, there is an increased need for machine learning experts to analyze datasets and create optimal networks to train them. Neural Architecture Search (NAS) was created as a subset of AutoML and hyperparameter tuning as a process of automating the engineering required to design a machine learning model. Neural architecture search has shown promising results in the realms of convolutional neural networks, computer vision, and semantic segmentation [6].

With the improvements shown through NAS, it is a bit surprising how little research has been dedicated to NAS in the application of medical imaging. From my research, there are only two works that apply NAS techniques to medical imaging segmentation [7], [8]. The NAS-UNet introduced in 2019 was one of the first papers to tune medical image segmentation models [7]. This technique used a micro-search strategy for tuning, which allowed the model to keep the hand-crafted exterior architecture that defines the outward U-shape. This broader exterior architecture is filled in by tuned cells. The authors described three primitive operation sets (downsampling, upsampling, and normal) to work together to help define these cells and how they work together. This NAS-UNet worked not only to define a formal U-Net search process but also to develop an optimal search strategy. The other medical segmentation work, [8], focuses on NAS implementation for 3-D radiological imaging models using similar techniques.

This project works to accomplish a similar goal - the ability to tune medical imaging segmentation models. However, this project works to redefine how the search process works in a user-friendly framework that allows the analysis of not only similar parameters to the NAS-UNet but also more parameters. The framework created in this work through the use of the Keras-Tuner can tune beyond the architectural scope that this work, and the NAS-UNet, focus on and can adjust more algorithmic parameters within the model as well.

3. Search Space and Models

3.1 Hyper Parameters

In deep learning networks, a hyperparameter is a parameter whose value is specified to control the learning process. These differ from parameters, as they are given to the network by the engineer instead of learning through the training process. Within hyperparameters, there are two distinct categories: algorithmic and architectural. Algorithmic hyperparameters are parameters that help evaluate how the algorithm will run, such as the learning rate or the number of epochs. Architectural parameters, on the other hand, help define the overall structure of the model. Since this work is searching to find the best U-Net architecture, our tuning will focus on eight different architectural hyperparameters.

Number of Layers: One of the most interesting aspects of architectural search is the depth of the model. As each layer of a convolutional neural network is used, the network is able to extract more information. So, intuitively, the more layers added, the more elaborate models can be represented. While deep learning has seemingly avoided the traditional machine learning problem of overfitting [9], there is still interest in exploring the trade-off between complexity and accuracy within the U-Net. Due to memory constraints, this work is unable to search through an "infinite" convolutional network and will instead be comparing models with 3 to 5-layer depths.

Filter Values: Filters are often thought of as feature extractors in traditional convolutional neural networks, and the U-Net is no exception. The only difference is that in the U-Net architecture, the filter size is reflected in the encoder/decoder mirror path. The original U-Net architecture started with 64 filters, with that value doubling for each layer in the network. This doubling technique is used in the tunable U-Net, but the starting filter values can be 8, 16, 32, or 64.

Use Batch Normalization: One of the most frequent additions to modified U-Net architectures is the addition of batch normalization layers. Though batch normalization now seems ubiquitous in the world of deep learning, this normalization technique was introduced only two months before the original U-Net paper in 2015 [10]. The original batch normalization paper introduced the method as a solution to reducing the internal covariate shift. Internal covariate shift is the shift that occurs when, during backpropagation of the neural network, the input distribution changes. In our tunable U-Net, this can be set to true or false. When true, every two-dimensional convolutional block uses batch normalization.

Dropout Value: Introduced as a way to prevent overfitting in neural networks, the dropout technique provides the probabilistic removal of inputs during training [11]. While the original U-Net paper did not specify their dropout value used, based on an associated U-Net GitHub, I set the original U-Net dropout value at 0.55. In this work, the dropout value for the tunable U-Net could be set between 0.0 and 0.6.

Dropout Change Per Layer: In a few U-Net GitHub implementations, the researchers did not use dropout in the first few layers of the model. This was not possible to execute in this work, given the way the customizable U-Net model was implemented. The dropout change per layer allows more dropout to be added in subsequent layers of the model. For the tunable U-Net, this value could be set between 0.0 and 0.05.

Use Dropout on Up-Sampling: This parameter allows the usage of dropout during the decoding path of the U-Net architecture. If set to true in the tunable U-Net, the parameter allows the decoding path to follow the exact dropout structure as the encoding path. This includes the dropout values and any dropout changes per layer.

Activation Function: In between the blocks of the traditional U-Net, the relu activation function is used to decide whether or not a neuron will activate. The original U-Net utilizes the Relu architecture, which was created to solve and avoid the vanishing gradient problem through its simple max(0,z) function. In the tunable U-Net architecture, either the Relu or the Elu activation function can be chosen. The Elu function works very similarly to the Relu except for negative values, which have a slight slope instead of a zero value.

Decoder Block Type: The decoder block type refers to the technique that the decoder path within the U-Net uses to up-sample the image. In recent years, especially with the growing popularity of GANs, much research has gone into finding the best decoder architecture to reconstruct images [12]. The original U-Net architecture, [1], used two dimensional strided up-sampling and convolutional blocks to recreate the mask images. In a recent paper performing deep learning for the diagnosis of retinal disease, they implemented bilinear interpolation instead of traditional upsampling [13]. This helped reduce the artifacts within the image and improve upon the overall performance. Finally, a third type of decoder block that this work explores is transposed convolution, also known as deconvolution. This technique has been used in popular publications and projects [14], [12]. It utilizes transposed convolutional blocks instead of upsampling to mirror the original encoder better.

3.2 Search Strategy

After defining the search space, there is a need to define a strategy to search for the best set of hyperparameters. I chose an approach from a recent work that expanded upon the idea of random search using some explore-exploit theory, known as the hyperband search technique [15]. This work utilizes the hyperband approach to search over the entire U-Net architecture, not just micro-blocks. To evaluate the best set of hyperparameters, the hyperband algorithm used a custom loss function that combined binary cross-entropy with dice loss.

Tunable Parameter	**Possible Values**
Number of Layers	3, 4, 5
Filter Values	8, 16, 32, 64
Batch Normalization	True / False
Dropout Value	0.0 – 0.6
Use Dropout on Up Sampling	True / False
Dropout Change Per Layer	0.0 – 0.05
Activation Function	‘relu,’ ‘elu’
Decoder Type	‘simple,’ ‘bilinear,’ ‘transposed.’

Table 1: Description of the tunable parameters in HyperUNet class.

The hyperband algorithm works by randomly selecting a set of hyperparameters and training them for a short period. It then repeats this technique on more groups of random hyperparameters. Then, it takes the models that performed well and trains them for longer. This technique repeats until the hyperparameter space is narrowed down and trains the remaining best models for a set maximum period. This work set the maximum at 200 epochs.

After choosing the hyperband search technique, I wanted to find a simple yet powerful framework to implement it. Through research, I settled on the usage of the Keras-Tuner [16], a newly released TensorFlow package that allows the creation of easily implementable tunable classes. This work modified a customizable U-Net architecture [17] with new hyperparameters and Keras-Tuner to create a tunable class named the HyperUNet. This tunable class tunes the parameters described in Table 1.

4. Evaluation

4.1 Data Sets and Preprocessing

This work searches through the above-defined parameters space of three publicly available medical imaging datasets. Different imaging modalities and areas of the body were chosen for each dataset. Additionally, each dataset was selected to show a unique challenge within the field of medical imaging segmentation. Finally, to ensure this work is accessible to both experts and non-experts, there was minimal preprocessing done on each of the datasets. Pre-processing is a technique researchers use to drive up performance. While there are plenty of simple open-source preprocessing methods, this work wanted to allow the architecture to speak for itself. These differences are emphasized as the selections were made to provide a robust study on the tunable Hyper U-Net class under different circumstances.

Chest Segmentation: The first dataset, taken from Kaggle [18], consists of 267 CT patient lung scans, as well as manually segmented masks. This dataset represents an especially current challenge within radiology. CT scans and their resulting lung size calculations, through mask segmentations evaluated by deep learning, have been useful in the detection of pneumonia and are of interest for COVID-19 research [19], [20]. This is a reasonably straight forward clinical image segmentation task. For that reason, no preprocessing other than preparing the images to be fed into the deep learning model (re-sizing and binarizing the mask) was applied.

LGG Brain MR Segmentation: Another dataset evaluated was the dataset used in [21], [22] collected from the Cancer Imaging Archive. This dataset consists of 110 patients with lower-grade glioma FLAIR MR scans, with over 4000 individual FLAIR slices and matching manually segmented masks. This is a

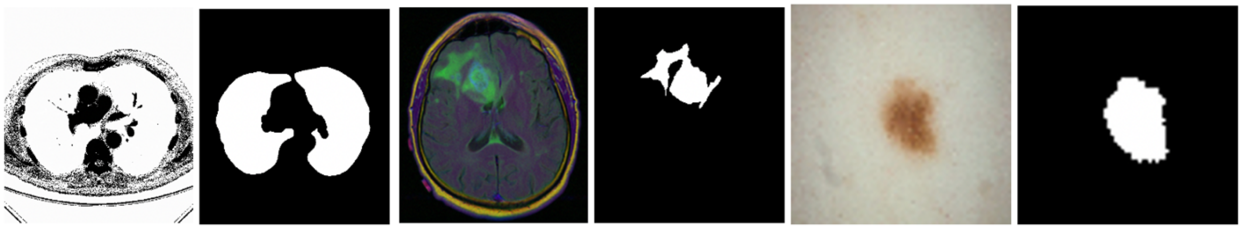

Figure 2: An image mask pair example from each of the datasets. From left to right, Chest Segmentation, LGG Brain MR Segmentation, Skin Lesion Segmentation.

challenging dataset for segmentation as more than half of the sliced images did not contain a tumor, so they had a blank mask. This requires the segmentation task to emphasize cutting down on false negative values, as the model can be accidentally taught to predict blank masks even when a tumor is present. Also, to meet a better baseline for evaluation, this dataset was preprocessed with histogram inspired MR normalization [23], as well as the simple preprocessing techniques that are done in for the chest dataset to feed the images into the network. Similar to the chest segmentation dataset, no other preprocessing was added to the network to evaluate the improvements that just an architecture can show through hyperparameter tuning.

Skin Lesion Segmentation: The third dataset evaluated was a large dataset consisting of photographic images of lesions on the skin from the ISIC 2018 Melanoma Image Detection Challenge [24], [25]. Since these are medical images, but not radiological scans, these images vary widely. The angles, skin tones, discoloration of the lesions, placement of lesion within the image, and numerous other variations complicated the seemingly simple segmentation task. To combat this challenge, I added some simple data augmentation to the preprocessing of this dataset. The random shifts, horizontal and vertical flips, rotations, and magnification of each training batch allowed the data to mimic random variations within the set better. This allowed for better baseline training. Similar to the LGG Brain and Chest Segmentations, some barebones preprocessing was added to feed the images into the network.

4.2 Training and Tuning Model Set-Up

This work is focused on the search for the best U-Net model-based on different medical datasets. However, this project is also focused on evaluating a simple framework for experts and non-experts alike to use this search technique. This was taken into consideration when setting up a training system. Often, researchers would adapt parameters such as learning rate or batch size to drive up the performance of a network. However, this work did not adjust those expert techniques which require domain or deep learning knowledge. Instead, this work standardized training and did not tweak algorithmic parameters between models.

This work used two Euler multi-processing GPU's to train the traditional U-Net models and four to tune the models. Between training and tuning, the algorithmic hyperparameters remained constant. These hyperparameters included the Adam optimizer trained with a learning rate of 1e-4. The models used a kernel size of 3 and a max-pooling/up-sampling stride value of (2,2), consistent with the original U-Net architecture. This allowed the model to run without any extra image padding. A batch size of 16 was used for each of the datasets to optimize the available memory, and each models' steps per epoch were calculated concerning the training size compared to this batch (i.e., training size // 16). Finally, the final activation function for each model was the sigmoid function to ensure proper semantic segmentation and synthesis of the resulting mask image.

Additionally, each of the models was trained and tuned with a loss function that combined binary cross-entropy and dice coefficient loss. In the original U-Net paper, only binary cross-entropy was utilized. This is also known as the log loss function, as it penalizes segmented loss better than common loss functions like mean squared error, which does not place enough emphasis on punishing false negatives. This work emphasizes the penalization of false negatives by adding the dice coefficient loss to binary cross-entropy.

$$loss = Binary\ Cross\ Entropy + (1 - Dice\ Coefficient)$$

During the hyperparameter tuning process, the hyperband algorithm was used to search through the possible networks. A total of 500 random networks were chosen in the beginning, allowing the hyperband model to search this space intuitively. Each model was trained for a maximum of 200 epochs. The hyperband search technique was repeated three times, to standardize and verify the result. These values were limited due to memory resources and could be increased in a later, more comprehensive study for more tenable results.

4.3 Evaluation Metrics

To evaluate the performance of the original U-Net model compared to the model trained with the optimal hyperparameters (referred to as the Tuned Model), I used eight common metrics. The first two metrics are standard metrics for semantic segmentation: IOU and Dice Coefficient. The IOU (Intersection over the Union) score refers to how well the actual and predicted mask overlap with one another. If there is a perfect overlap, the IOU score is 1. This is a metric that was used to help measure the overlap of the predicted mask but is more common in bounding box segmentation where the union isn't always a perfect overlay. The Dice Coefficient may be of most interest as it is a measure of not only the true positives or overlayed values but also penalizes for false positives in the predicted mask.

This is similar to other metrics that were used in the evaluation, such as Thresholded pixel accuracy, precision, and recall. These values were Thresholded at 0.5, with pixels above set to 1 and below set to 0, to account for the fact that predicted values may be close to 0 and 1 but not exact. These metrics evaluate different aspects of accuracy. Pixel accuracy shows the overall accuracy taking into account how many pixels the image correctly guessed within the whole. Precision works similar to the dice coefficient, as it penalizes false positives within its calculation. Recall can also be thought of as sensitivity, as it penalizes false negatives. These metrics help provide a more detailed understanding of how the model learns, and where possible weak points may lie.

Finally, I chose three loss functions as my last metrics. These were mean squared error, mean absolute error, and the mean value of my created loss function, the combination of binary cross-entropy and dice loss. While the downfalls of mean squared error, and in turn, mean absolute error, were discussed above when describing the reason behind the usage of the BCE Dice Loss function, these are still common metrics used in other semantic segmentation work. These values help to complete a robust set of metrics used to evaluate the models.

5. Results

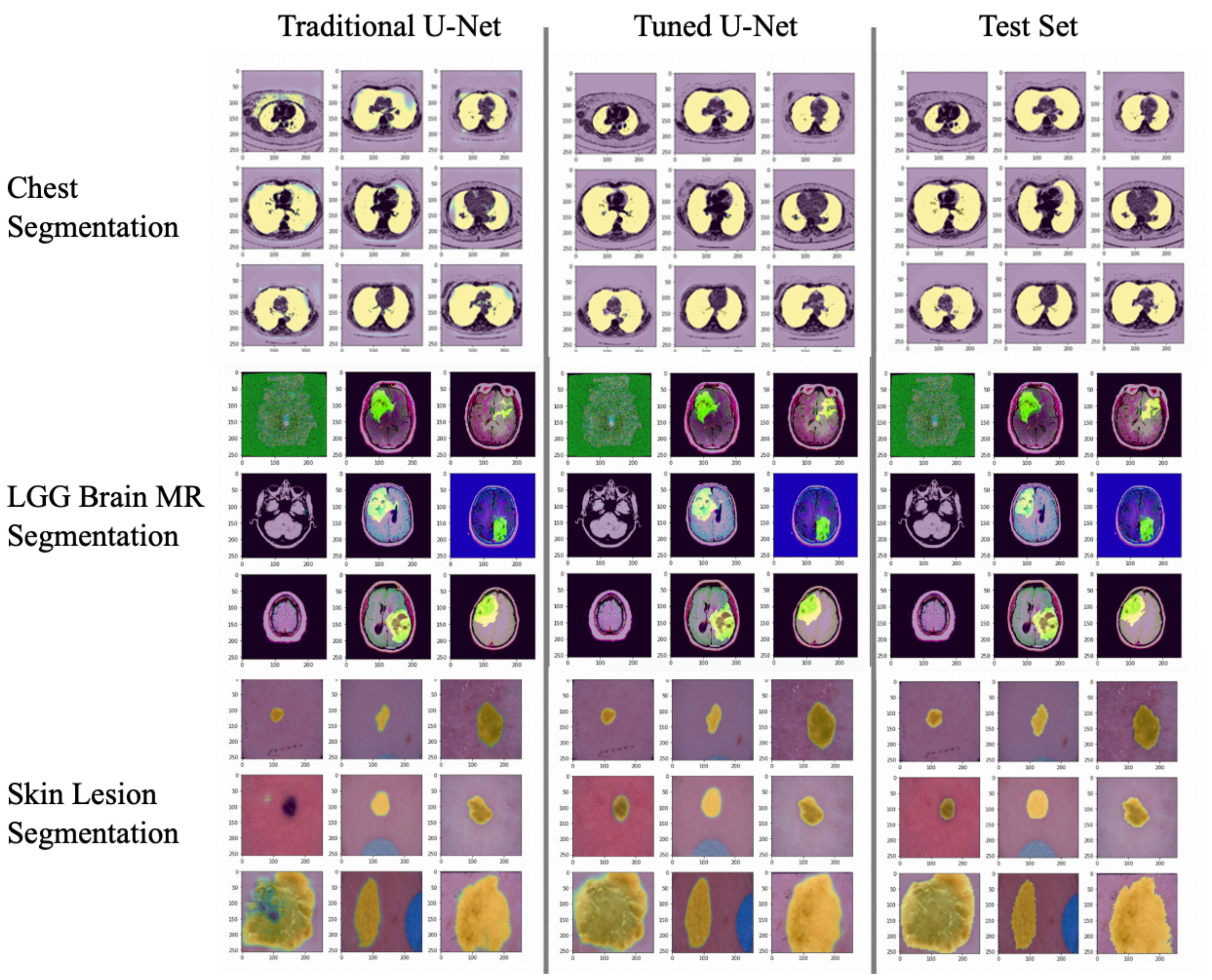

Figure 3: The resulting images from 9 different randomly selected image mask pairs from each dataset. The traditional and tuned dataset images represent the predicted mask from each of those datasets overlayed with the original image. The test set image represents the true mask value overlayed with the original image.

The tunable U-Net model showed promising results for all three datasets. The tunable U-Net structure for each dataset shows noticeable visual improvement, as seen in Figure 3. Beyond visual improvements, Table 2 shows that the tuned U-Net structure for each of the datasets improved for nearly every calculated metric. The two metrics that I was most interested in, IOU and Dice Coefficient, showed significant improvement for every dataset. The only metric that did not improve with the tuned U-Net architecture was the calculated Thresholded precision for the Skin Lesion Segmentation dataset. This could be of interest as precision is used to analyze false positives in the model. While researchers may consider this, I do not believe that this is as problematic as the other metrics, including the Dice Coefficient – which also penalizes false positives, showed consistent improvement.

Dataset	Model Type	IOU	Dice Coefficient	Thresh. Pixel Accuracy	Thresh. Precision	Thresh. Recall	MSE	MAE	Mean BCE + Dice Loss
Lung Segmentation	Basic U-Net	0.804	0.9133	0.955525	0.90925	0.91488	0.0337	0.0523	0.8769
	3 Search	0.9175	0.9612	0.97823	0.969663	0.95229	0.0143	0.0192	0.3114
LGG Brain MRI Segmentation	Basic U-Net	0.4814	0.6499	0.993884	0.740569	0.5815	0.0059	0.0061	0.44435
	3 Search	0.5129	0.67808	0.994079	0.727956	0.63502	0.0058	0.0059	0.4129
Skin Lesion Segmentation	Basic U-Net	0.6583	0.79536	0.915312	0.910508	0.76229	0.0504	0.0746	1.214
	3 Search	0.7404	0.85259	0.9289783	0.86199	0.86246	0.0383	0.0571	0.8954

Table 2: The evaluation metrics for all three datasets for their traditional and tuned U-Net models. The tunable U-Net models are described as 3 Search, as they used three hyperband iterations. This is specified, as future work may want to increase hyperband iterations. They were evaluated with eight common and relevant segmentation metrics.

Beyond the empirical improvements between the traditional U-Net and each dataset's tuned models, it is also of interest to analyze the resulting tuned architectures between datasets. The resulting parameter values for each dataset's tuned architectures can be seen in Table 3. Between these models, some trends can be observed with caution. All tuned architectures chose a relu activation function and batch normalization. Additionally, all models tended to the higher layers, with 4 to 5 layers selected. The filter values varied more than I would expect as well as the dropout values ranging from 0.2 to 0.5. Two of the three models preferred a transposed decoder, and no models chose the new bilinear technique. These observations may be useful to inform construction for future U-Net models, especially for similar datasets.

Model Parameters	Traditional U-Net	Chest Segmentation	LGG Brain MR Segmentation	Skin Lesion Segmentation
Number of Layers	4	4	5	5
Filter Values	64	16	32	64
Batch Normalization	False	True	True	True
Dropout Value	0.55*	0.4	0.2	0.5
Use Dropout on Up Sampling	False	False	False	True
Dropout Change Per Layer	0.0	0.045	0.03	0.035
Activation Function	'relu'	'relu'	'relu'	'relu'
Decoder Type	'simple'	'transpose'	'transpose'	'simple'

Table 3: The selected tuned model parameters for each of the evaluated datasets.

However, it is important to note that those observations were made with caution. Since the work was done to analyze an overall framework accessible to experts and non-experts alike, there were many assumptions made to standardize the training between datasets. Standardizing the dataset allowed the work to be trained easier and easily reproducible between datasets. This was done through techniques such as setting the epochs at arbitrarily high value of 500, creating step-sizes relative to the size of the training dataset, and setting the learning rate at 1e-4. These techniques and others may be changed by a deep learning or domain expert. There are many different tricks (changing the learning rate throughout training, decreasing epochs, and increasing step size, etc.) that researchers will use to improve the overall score of the dataset.

Additionally, there were assumptions made to constrain memory usage during training and tuning. With the GPU resources available, I was only able to tune the model with three hyperband iterations up to 200 epochs. Typically, the model should be trained to an epoch value that is equivalent to 'convergence and a half' and should use as many hyperband iterations as possible. These assumptions made to constrain memory meant that even if the assumptions made to standardize the dataset were true, the resulting tuned model might not be the actual best architecture within the search space.

6. Conclusions and Future Work

In conclusion, I believe that this framework shows promise to be used in practical and theoretical settings within the Department of Radiology at the University of Wisconsin-Madison. The use of the Keras-Tuner has shown improvement and ability to be used in different U-Net applications, which would be interesting for future research done within the group. Beyond this work, I have talked with some members of the group to see how they could implement this into their own expertly created projects. Finally, I have started to work on adding more models into the codebase, including a modified U-Net with a VGG backbone.

References:

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *ArXiv150504597 Cs*, May 2015, Accessed: Jun. 18, 2019. [Online]. Available: http://arxiv.org/abs/1505.04597.

[2] Y. Deng, Y. Sun, Y. Zhu, M. Zhu, W. Han, and K. Yuan, "A Strategy of MR Brain Tissue Images' Suggestive Annotation Based on Modified U-Net," *ArXiv180707510 Cs*, Jul. 2018, Accessed: Jun. 09, 2020. [Online]. Available: http://arxiv.org/abs/1807.07510.

[3] S. L. Oh, E. Y. K. Ng, R. S. Tan, and U. R. Acharya, "Automated beat-wise arrhythmia diagnosis using modified U-net on extended electrocardiographic recordings with heterogeneous arrhythmia types," *Comput. Biol. Med.*, vol. 105, pp. 92–101, Feb. 2019, doi: 10.1016/j.compbiomed.2018.12.012.

[4] S. M. K. Hasan and C. A. Linte, "U-NetPlus: A Modified Encoder-Decoder U-Net Architecture for Semantic and Instance Segmentation of Surgical Instrument," *ArXiv190208994 Cs*, Feb. 2019, Accessed: Jun. 09, 2020. [Online]. Available: http://arxiv.org/abs/1902.08994.

[5] M. A. Rahman, “Neural Architecture Search (NAS)- The Future of Deep Learning,” *Medium*, Nov. 03, 2019. https://towardsdatascience.com/neural-architecture-search-nas-the-future-of-deep-learning-c99356351136 (accessed Jan. 24, 2020).

[6] T. Elsken, J. H. Metzen, and F. Hutter, “Neural Architecture Search: A Survey,” p. 21.

[7] Y. Weng, T. Zhou, Y. Li, and X. Qiu, “NAS-Unet: Neural Architecture Search for Medical Image Segmentation,” *IEEE Access*, vol. 7, pp. 44247–44257, 2019, doi: 10.1109/ACCESS.2019.2908991.

[8] Z. Zhu, C. Liu, D. Yang, A. Yuille, and D. Xu, “V-NAS: Neural Architecture Search for Volumetric Medical Image Segmentation,” *ArXiv190602817 Cs Eess*, Aug. 2019, Accessed: Jan. 30, 2020. [Online]. Available: http://arxiv.org/abs/1906.02817.

[9] T. Poggio *et al.*, “Theory of Deep Learning III: explaining the non-overfitting puzzle,” *ArXiv180100173 Cs*, Jan. 2018, Accessed: Aug. 02, 2020. [Online]. Available: http://arxiv.org/abs/1801.00173.

[10] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *ArXiv150203167 Cs*, Mar. 2015, Accessed: Jul. 27, 2020. [Online]. Available: http://arxiv.org/abs/1502.03167.

[11] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 56, pp. 1929–1958, 2014.

[12] Z. Wojna *et al.*, “The Devil is in the Decoder: Classification, Regression and GANs,” *ArXiv170705847 Cs*, Feb. 2019, Accessed: Aug. 12, 2020. [Online]. Available: http://arxiv.org/abs/1707.05847.

[13] J. D. Fauw *et al.*, “Clinically applicable deep learning for diagnosis and referral in retinal disease,” *Nat. Med.*, vol. 24, no. 9, Art. no. 9, Sep. 2018, doi: 10.1038/s41591-018-0107-6.

[14] A. Radford, L. Metz, and S. Chintala, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,” *ArXiv151106434 Cs*, Jan. 2016, Accessed: Aug. 12, 2020. [Online]. Available: http://arxiv.org/abs/1511.06434.

[15] “[1603.06560] Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization.” https://arxiv.org/abs/1603.06560 (accessed Jun. 09, 2020).

[16] *keras-team/keras-tuner*. Keras, 2020.

[17] “karolzak/keras-unet: Helper package with multiple U-Net implementations in Keras as well as useful utility tools helpful when working with image semantic segmentation tasks. This library and underlying tools come from multiple projects I performed working on semantic segmentation tasks.” https://github.com/karolzak/keras-unet (accessed Aug. 12, 2020).

[18] “Finding and Measuring Lungs in CT Data.” https://kaggle.com/kmader/finding-lungs-in-ct-data (accessed Aug. 02, 2020).

[19] X. Gu, L. Pan, H. Liang, and R. Yang, "Classification of Bacterial and Viral Childhood Pneumonia Using Deep Learning in Chest Radiography," in *Proceedings of the 3rd International Conference on Multimedia and Image Processing - ICMIP 2018*, Guiyang, China, 2018, pp. 88–93, doi: 10.1145/3195588.3195597.

[20] H. Zhang *et al.*, "Automated detection and quantification of COVID-19 pneumonia: CT imaging analysis by a deep learning-based software," *Eur. J. Nucl. Med. Mol. Imaging*, Jul. 2020, doi: 10.1007/s00259-020-04953-1.

[21] M. A. Mazurowski, K. Clark, N. M. Czarnek, P. Shamsesfandabadi, K. B. Peters, and A. Saha, "Radiogenomics of lower-grade glioma: algorithmically-assessed tumor shape is associated with tumor genomic subtypes and patient outcomes in a multi-institutional study with The Cancer Genome Atlas data," *J. Neurooncol.*, vol. 133, no. 1, pp. 27–35, 2017, doi: 10.1007/s11060-017-2420-1.

[22] M. Buda, A. Saha, and M. A. Mazurowski, "Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm," *Comput. Biol. Med.*, vol. 109, pp. 218–225, Jun. 2019, doi: 10.1016/j.compbiomed.2019.05.002.

[23] X. Sun *et al.*, "Histogram-based normalization technique on human brain magnetic resonance images from different acquisitions," *Biomed. Eng. OnLine*, vol. 14, no. 1, p. 73, Jul. 2015, doi: 10.1186/s12938-015-0064-y.

[24] N. Codella *et al.*, "Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)," *ArXiv190203368 Cs*, Mar. 2019, Accessed: Aug. 02, 2020. [Online]. Available: http://arxiv.org/abs/1902.03368.

[25] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Sci. Data*, vol. 5, p. 180161, 14 2018, doi: 10.1038/sdata.2018.161.