Tuning Full-Connected U-Net Skip Connections with β-DARTs

Christian Granados
College of Computing, Data Science, and Society
University of California, Berkeley
centralsafety23@berkeley.edu

Cole Agard
School of Information
University of California, Berkeley
coleagard@berkeley.edu

Abstract

Semantic segmentation is a crucial task for many computer vision applications. This thesis aims to improve the skip connection structure of the widely used U-Net architecture via Differentiable Architecture Search (DARTs). DARTs relaxes the discrete filter selection problem into a continuous optimization by assigning trainable weights to each candidate operation. To address the performance collapse issue in DARTs, beta-decay regularization is incorporated during the architecture search phase. The search space and candidate operations are modified to allow more flexible cell structures. The HAM10000 dataset of dermatoscopic images is utilized; with additional data augmentation to increase robustness. Training is composed of both a search and discretion phase. The top operations found in the search phase are then selected to form the final network. While the final segmentation performance falls below some other methods (IoU: 0.51, Dice: 0.65; Accuracy: 0.67), the work provides insights into applying DARTs to a non–traditional search space.

Introduction

Semantic Segmentation is the process of separating a target class from a non-target background. Systems that segment images in this way are widely used as subprocesses of larger networks such as YOLOv7 (Wang et al., 2023) for object detection, and autonomous vehicle applications (Liu et al., 2023). However, it is also used in isolation of biomedical image segmentation. For applications such as MRI scans, skin lesion dermoscopic pictures, and much more, it can be used as a quick, accurate, and cost effective way to provide better care.

The U-Net architecture was a major breakthrough in the image segmentation field at the time of its 2015 publishing. The network itself was able to efficiently and accurately segment images with a (comparatively) sparse model and little data. The advancement was mainly credited to an encoder-decoder structure that extracted high-level features, and skip connections that gave the decoder more fine-grained context. Iterations on U-Net (Ronneberger et al., 2015) such as

U-Net++ (Zhou et al., 2018), U-Net3+ (Huang et al., 2020), and ELU-Net (Deng et al., 2022), have attempted to improve the architecture by modifying the skip connection. This thesis aims to take a more high-level approach to designing these skip connections by utilizing DARTs (Liu et al., 2019), a Neural Architecture Search algorithm, to make these decisions via a gradient-based relaxation of the filter search space.

Background

Data Description

For this thesis the HAM10000 dataset was used. It is composed of dermatoscopic images depicting pigmented skin lesions. These lesions come from a variety of populations across the globe, have a large training set of 10015 images, and possess a variety of diagnoses. Originally the dataset was constructed to bridge the gap between neural network models and the data quality/quantity of biomedical images. This runs in contrast to the commonly used ISIC Challenge Datasets; these datasets contain no more than 200 images in most cases.

However, the HAM10000 does not have semantic segmentation masks by default. Thanks to a recent 2020 paper, Human-Computer Collaboration for Skin Recognition, semantic segmentation masks have been added to the entirety of the 10015 dermatoscopic images. To create the masks, the HAM10000 images were first passed through a pre-trained Fully Convolutional Network (FCN). The paper details ResNet34-based and VGG16-based backbones for this network. After FCN inference, each image and mask was then manually tuned by a dermatologist.

Figure 1: an example of a skin lesion and its semantic segmentation mask

U-Net and its Variants

U-Net (Ronneberger et al., 2015) gets its name from the shape of the network itself. It is composed of an encoder and decoder portion. What separates it from a strict autoencoder

structure is the skip connections between it. The skip connections help to carry over higher level features, while the bottom-most connection carries over important latent features. The encoder feature maps themselves are concentrated to the corresponding decoder feature maps that match in image dimensionality.

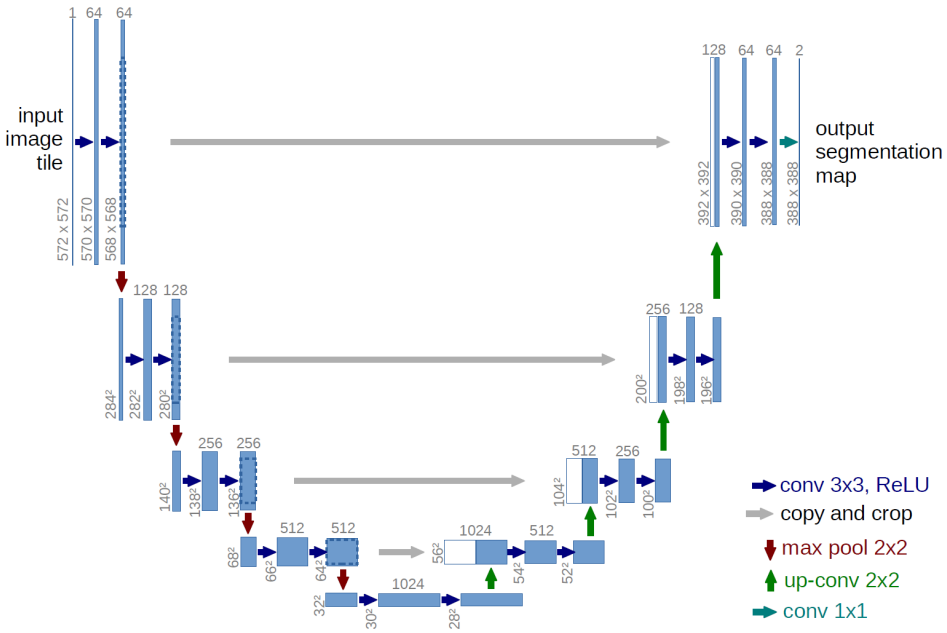

Figure 2: U-Net architecture from the original paper

How the skip connections are constructed can heavily impact the final network performance. A good analogy is the conversation between someone commissioning a piece of art and an artist. The commissioner has an idea of what he or she would like, but has to translate that to language for the artist to understand. Depending what the artist is told, he or she will do their best to interpret and best reconstruct what the commissioner wants. In this case, *how* the description is communicated has a large effect on the final output. Thus, it is no surprise a majority of the research into different architectures has been focused on the connections between the encoding and decoding branches.

Differentiable Architecture Search (DARTs)

Deciding between different filters is usually a binary decision. A filter is included as a connection or not; it is a discrete decision. DARTs relaxes the assumption of binary filters, and instead assigns a learnable parameter to each filter. During training, the parameter values for all filters are put through a SoftMax and used as a weighted sum for the output.

As an analogy, picture an athlete training for a marathon. He is given a choice between 10 shoes. In the discrete case, he can choose a single shoe to train and compete with. He can do this

process many times, correcting for increased performance from training, to determine which shoe is for him. However, what if, during the training process, he wore each shoe in equal proportion. Over many training sessions he would break into each shoe, see his performance per-session, and form an opinion on every shoe. Eventually he would start prioritizing certain shoes that felt comfortable and allowed him to perform well. By the time the marathon came around, he would then feel confident in selecting the best shoe for him. While training may suffer in the beginning because there is a large amount of shoes to get used to, the final result of having information on all shoes in a single training cycle more than justifies it.

In other words, let $\mathbb{O}$ be the set of candidate operations, where each operation represents a function $o(.)$ to be applied to an input $x^{(i)}$. The operation mixing weights for a pair of nodes (i, j) are parameterized by a vector $\alpha^{(i,j)}$ of dimension $|\mathbb{O}|$. Thus, traversing from a node i to a node j is defined as:

$$o^{(i,j)}(x) = \sum_{o \in \mathbb{O}} \frac{exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathbb{O} exp(\alpha_{o'}^{(i,j)})}} o(x)$$

Inclusion of both trainable weights and alpha values presents itself as a nested optimization task. To avoid the expensive inner optimization, DARTs optimizes each of the weight categories in-turn. While there is no proof of convergence, empirically the idea is to sufficiently approximate $\Delta_\alpha \mathcal{L}(w^*(\alpha), \alpha)$.

$$min_\alpha \mathcal{L}_{val}(w^*(\alpha), \alpha)$$
$$s.t. w^*(\alpha) = argmin_w \mathcal{L}_{train}(w, \alpha)$$

Thus, the gradient of α is determined per-batch, updated, and then the gradient of w is determined and updated. The gradient of w uses a gradually updating α. Thus the procedure becomes:

Create a mixed operation $o^{(i,j)}$ parameterized by $\alpha^{(i,j)}$ for each edge (i, j).While not converged, perform the following:

1) Update architecture α by descending $\Delta_\alpha(w - \epsilon \Delta_w \mathcal{L}(w, \alpha), \alpha)$
2) Update weights w by descending $\Delta_w \mathcal{L}_{train}(w, \alpha)$

It is important to note that the alphas (α) are trained on a validation set, whereas the weights (w) are trained on a dedicated training set. There is an additional Hessian approximation included within the training gradient, but for this thesis, that has been excluded from the gradient update scheme.

Beta-Decay Regularization for Differentiable Architecture Search (β-DARTs)

A central problem with DARTs is that as network size grows, the risk for performance collapse increases substantially. Performance collapse happens when the network selects a local minimum of many weight-free operations instead of training weight-filled operations. In a large network, DARTs prefer the stability and predictability of weight-free operations in contrast to untrained weight-filled operations (Peng et al., 2022). To correct for this issue, regularization of the alpha weights after SoftMax is employed; these post-SoftMax parameters are referred to as a Mixed Operations' β parameters.

If regularization is done before the SoftMax, the actual alphas can be reduced close to zero, however, what is more essential is an alphas' value *relative* to other alphas within a Mixed Operation. Thus, even if all values are very close to each other, if they are sufficiently small, the SoftMax proportions can still explode and give an unfair advantage to weight-free operations.

Solving the performance collapse issue has been the center of many papers researching DARTs. This version was chosen for its theoretical underpinnings, simplicity, and performance. Below is an implementation:

$$\mathcal{L}_{Beta} = \text{torch.mean}(\text{torch.logsumexp}(\text{self.model.arch}_\text{p}\text{arams}, \text{dim} = -1))$$
$$\text{loss} = \text{self.val}_\text{l}\text{oss}(\text{self.model}, \text{input}, \text{target}) + \lambda \mathcal{L}_{Beta}$$

Where λ is a linearly increasing sequence. In this thesis it starts at 0.5 and increases 0.5 per-epoch until reaching a maximum of 15.0.

If using the runner's analogy from earlier, picture an athlete that has trained barefoot for all his life. If a shoe company, who has done years of research and development, comes to this runner and promises a faster mile time with a specific shoe, the athlete will likely try it. However, to his dismay, his training quality and mile tests, at first, go down. He prefers to train without any shoes on because it is what he is used to. If he worked in the shoes, there would be a rebound in performance, and he would exceed his past personal records. So, instead, the shoe company signs a contract with him. In turn for a salary, he must train in the shoe for some given time period.

This is the logic of the regularization technique above. It changes the loss surface to avoid the local, weight-free minimum.

Reproducibility

This section details important information in reproducing the results presented in this thesis. This includes hyperparameters, datasets, and logging files. The purpose of a reproducibility document is to reduce the friction other researchers may go through to replicate an experiment within a research paper. This thesis already contains all code within a copyable collab notebook. The github repository itself will link to an open, shared repository – via google drive – of utilized training data.

Libraries

A requirements.txt will be included within the github for use in reproducing the following experiments. Since the requirements.txt is taken from a collab notebook, it will likely contain many additional packages. One can easily import the requirements into their own collab notebook in accordance with this public in-notebook tutorial.

Set Random Seeds

To reproduce results exactly, the main modeling notebook and data augmentation notebook will both include set seeds that can be modified by other researchers. The aim of having set seeds is to allow other researchers to exactly reproduce the conditions that led to the output detailed in the thesis.

Function	Seed
np.random.seed	2024
torch.manual_seed	2024
torch.cuda.manual_seed	2024

Logging

In addition to set seeds, there is a significant amount of logging and model saving integrated into the notebooks. This will allow other researchers to interrogate the weights – most likely alpha weights – of the network at any training epoch.

Saved information includes: current time, epoch, step within epoch, learning rate, loss, accuracy, IoU, Dice, and the alpha values associated with all blocks within the skip connection cells.

Hyperparameters

Weight Optimizer: SGD
Learning Rate: 0.025, Minimum of 0.003
Weight Decay: 0.0003
Momentum: 0.9
Gradient Clip: 5.0

Alpha Optimizer: RAdam, Betas = (0.5, 0.999)
Alpha Learning Rate: 0.0003

Beta-Regularization: Linear Increase 0.5 → 15

Criterion: BCEWithLogitsLoss
Epochs: 30
Data Length: 2048
Batch Size: 24

File Storage

All files, logs, models, and graphs are produced from code written and saved in a publicly available GitHub. See https://github.com/CGUCB/darts-unet for this information.

Methodology

Data Processing

The original images and masks were in a 650 x 450 pixel format. For the purposes of this thesis, the images and masks were first center-cropped to 450 x 450 pixels. Then, the images and masks were then scaled down to a size of 128 x 128.

Data Augmentation

While the dataset chosen has attempted to robustify itself via incorporating large amounts of varied dermatoscopic images, an augmentation step can further help to ensure a more generalizable network. Architecture search via DARTs is, by nature, dependent on the dataset used. If the goal is to find a highly generalized architecture, then the data that facilitates this search must also be sufficiently general. In the worst case, the assumptions and problems in discretization of the network can be further compounded by utilization of that architecture for data that is completely unseen during the Network Architecture Search (NAS) stage. There are

conflicting objectives in this case; a simpler the dataset leads to a faster runtime, but lower generalizability. Balancing these objectives is an ongoing field of research. (Moser et al., 2022)

For this thesis, an additional 10,000 images and masks were added to the training dataset. They are equally composed of the following augmentation techniques:

- Salt & Pepper Noise
- Salt & Pepper Noise + Random Crop
- Salt & Pepper Noise + Random Flipping, Rotation, and Scale
- Salt & Pepper Noise + Random Brightness & Contrast

The Salt & Pepper noise was added at a rate of 0.005; this means 0.5% of pixels were colored white and 0.5% were colored black. The Random Crop uniformly took random chunks of 256 x 256 pixel parts from the cropped 450 x 450 images. From here, the images were downscaled to 128 x 128. Random flipping uniformly chose between no flip, horizontal flip, and vertical flip. Random rotation chose a uniformly random angle between 0 and 360 degrees. Random scale picked a value between 0.8x and 2.0x. Random brightness and contrast was handled by the OpenCV function convertScaleAbs; this function took in an alpha and beta value that corresponded to brightness and contrast. These values were chosen uniformly in the intervals -75 to 75 and 0.7 to 1.3. The details of the implementation are included within a collab notebook within the linked github project.

Operations

In total there are 10 operations that compose each connection from one node to another. Each of them preserves the input and output image size, and is flexible to the input and output feature map dimensions.

- Zero (None)
- Skip Connection (Identity)
- Max Pool 3x3
- Average Pool 3x3
- Dilated Convolution 3x3, 5x5
- Depth-Wise Separable Convolution 3x3, 5x5
- Spatially Separable Convolution 5x5, 7x7

When connecting feature maps with different dimensions, special considerations have to be made for the zero, skip, max pools, and average pool operations. In the case of the zero operation, a blank feature map is replicated to match the output dimensions. For the skip connection if the input feature map is larger than the output, a group-wise average is done for each pixel. If the input feature map is smaller than the output feature map, the input features are concatenated

together until they meet the dimensionality of the output. For both the max and average pool, if the input dimension is larger than the output dimension, a group-wise three-dimensional kernel is applied to the feature maps. If the output dimension is larger than the input, the input feature map is concatenated to itself until output dimensionality is reached.

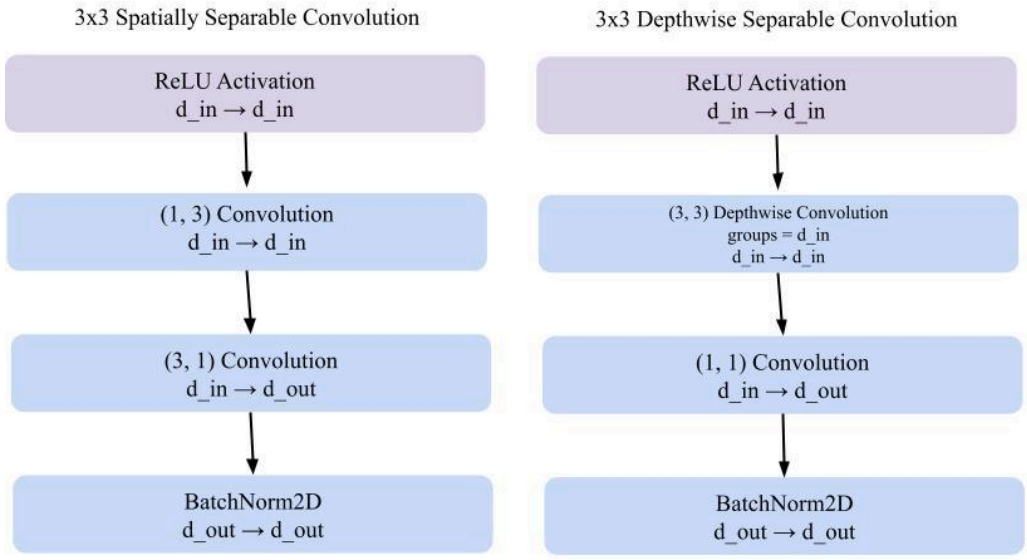

Figure 3: illustration of Depthwise and Spatially Separable Convolutions

The operation choice itself took heavy inspiration from sharpDARTs: Faster and More Accurate Differentiable Architecture Search (Hundt et al., 2019). Along with a new suite of operations, when compared with the original DARTs paper, this paper integrated a new operation block called a SharpSepConv. This block contained both a Depth-Wise Separable convolution and mid–filter choke operation. The main goal of this block and the candidate operation pool in general, is to increase the per-parameter efficiency of the possible filters. DARTs itself trains all candidate filters at the same time; this can pose large computational and runtime limitations. Through this perspective, the candidate operations for this thesis were derived. Both Depth-Wise and Spatially Separable filters divide a traditional 3x3, 5x5, or 7x7 convolution into multiple parts. When compared, these filters have less parameters than their vanilla convolution counterparts. Additionally, Dilated Convolutional blocks were integrated to allow for a parameter-efficient, large receptive field filter. Lastly, even though both MaxPool and AvgPool aim to distill down information without use of any parameters, they behave differently. Based on the vision task at hand, including both gives the network more flexibility.

Cell Search Space

In the original DARTs paper, the cell structure (and thus the search space) followed in the footsteps of NasNet (Zoph et al., 2018). In this scheme, each node was connected to all nodes previous to it. There would typically be four nodes within a single cell. To allow for more varied

connections, the input to the cell would draw on the output of the previous two cells. The output for the current cell would be the output of the final node. For nodes that had multiple mixed-operations leading in, the output of the node would be a concatenation of all incoming feature maps (Liu et al., 2019).

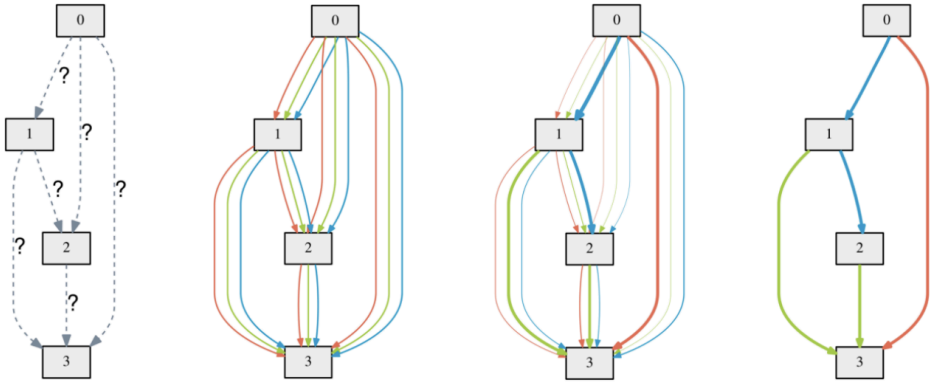

Figure 4: An overview of the cell structure of DARTs. Initially weights are unknown. Mixed operations are connected from a node to each previous one. Bilevel optimization trains both filter and alpha weights. During final network discretization, the filter with the highest weight is used.

This thesis follows a modified cell search space proposed by Rethinking DARTs Search Space and Renovating a New Benchmark (Zhang et al., 2023). A large issue with the original DARTs search space is that the possible architectural choices are limited to reflect the meta-structure of the search space. In other words, because cells in DARTs require picking the filter with the maximum alpha value for every mixed operation, any resulting cell will have the same general structure as any other; the only differentiating factor is the filter choice itself. However, with the aforementioned new search space, this is not the case; the network has more flexibility. The large changes to this search space include:

Applying the SoftMax over all incoming operations into a cell, not every Mixed Operation.

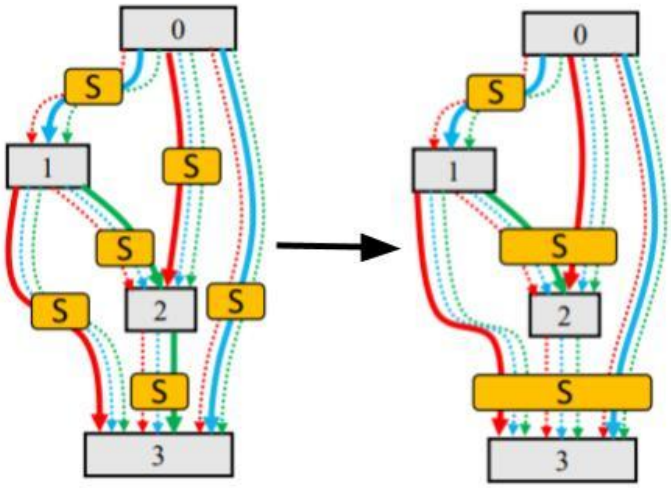

Figure 5: Conversion of SoftMax by Mixed Operation to SoftMax by all incoming operations into a node

Instead of the output of the cell being the output of the last node, output is composed of two polynary operation blocks.

Polynary operation blocks are a method by which multiple inputs, and their associated alpha values, are combined into a single output. In the paper, two main polynary operation blocks are used: the Sum Block (SSB), and the Concatenation Block (CSB).

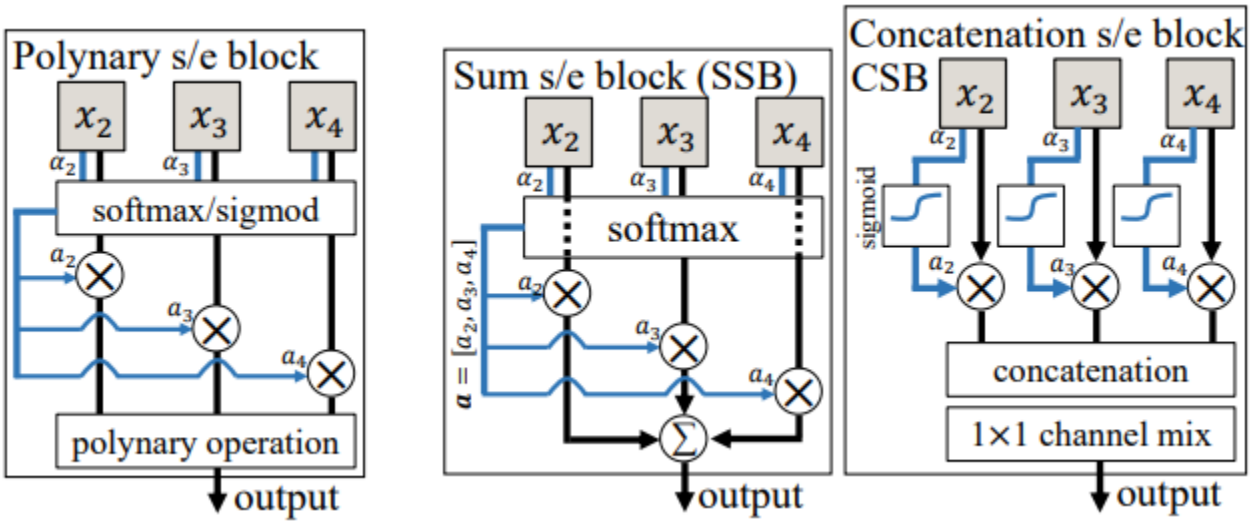

Figure 6: One parameter attached to each path is both optimizable and explains the significance of the path.

It is important to note that both a SoftMax and Sigmoid version are used. This allows for an output block in which different inputs are dependent on each other and independent of each other. Just as in the original DARTs, there are two inputs per cell, however, instead of being

composed of the output of the previous two cells, the inputs are the SSB and CSB outputs of the previous cell.

This thesis takes heavy inspiration from the modified search structure featured above. However, unlike the cells above, there will be a single input and output. This choice is made to ensure the skip connections are compatible with the Encoder/Decoder backbone and as to not add too much additional complexity.

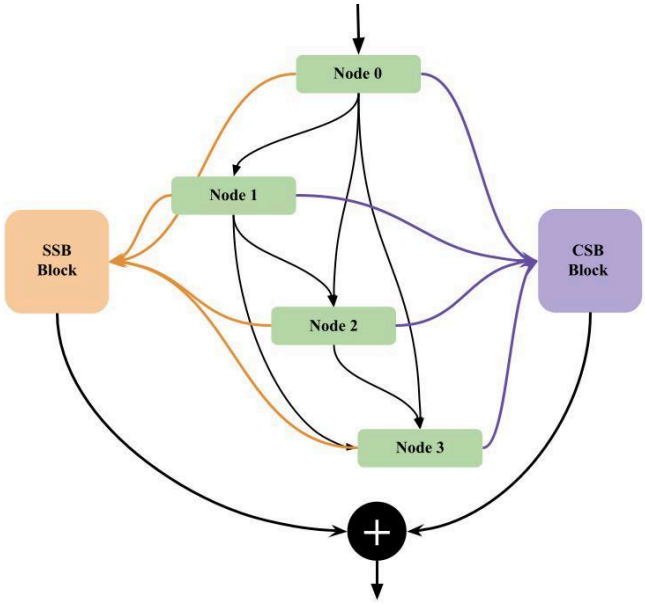

Figure 7: Skip Connection Cell Anatomy. Each black line represents a mixed operation. Each of the colored lines represents a connection paired with an alpha value. At the end both SSB and CSB feature maps are concatenated to form the output.

Network Structure

A large influence for this thesis was the network architectures of UNet 3+ (Huang et al., 2020) and ELU-Net (Deng et al., 2022). Both these variations of the U-Net architecture had full-scale deep skip connections. This thesis takes a similar approach in its network construction. While the encoder and decoder backbones are static and replicate the aforementioned papers, all skip connections are composed of dynamic DARTs-based cells.

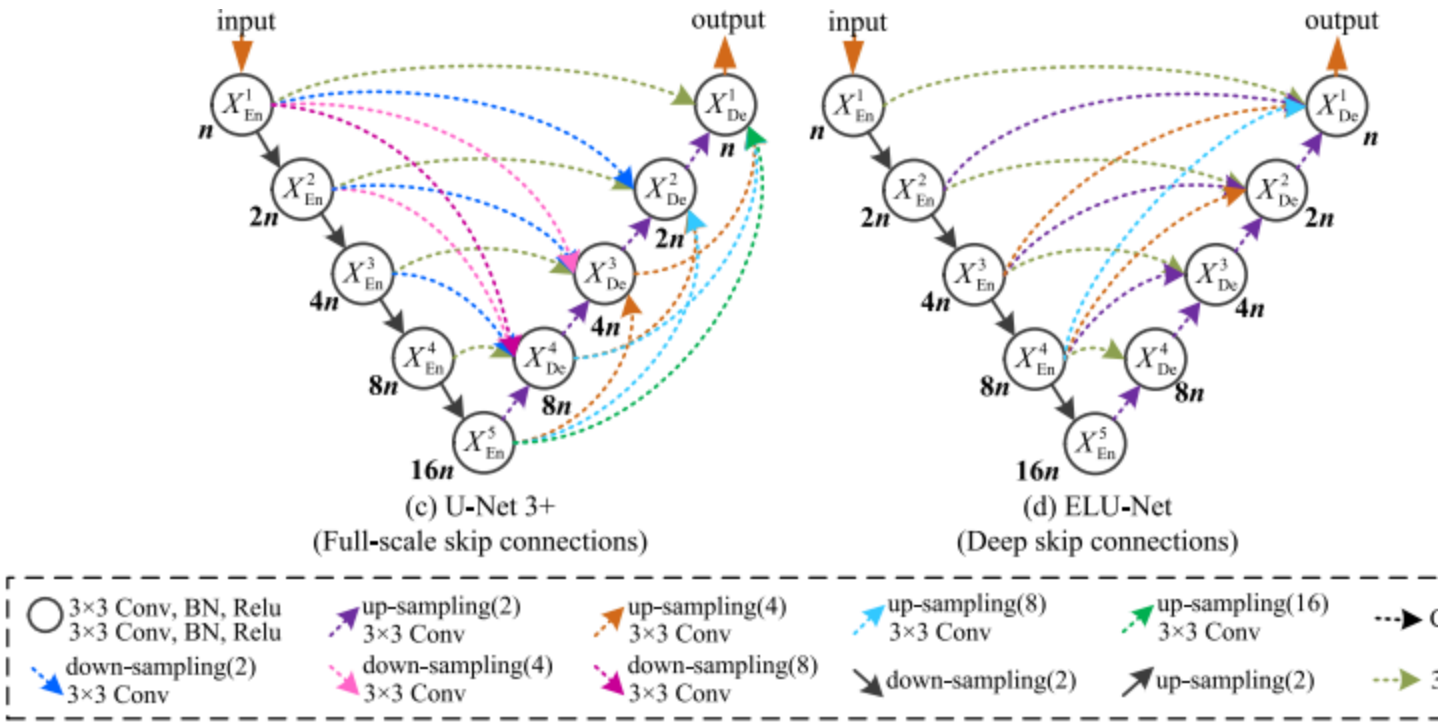

Figure 8: Excerpt from the ELLU-Net paper that details the network architectures

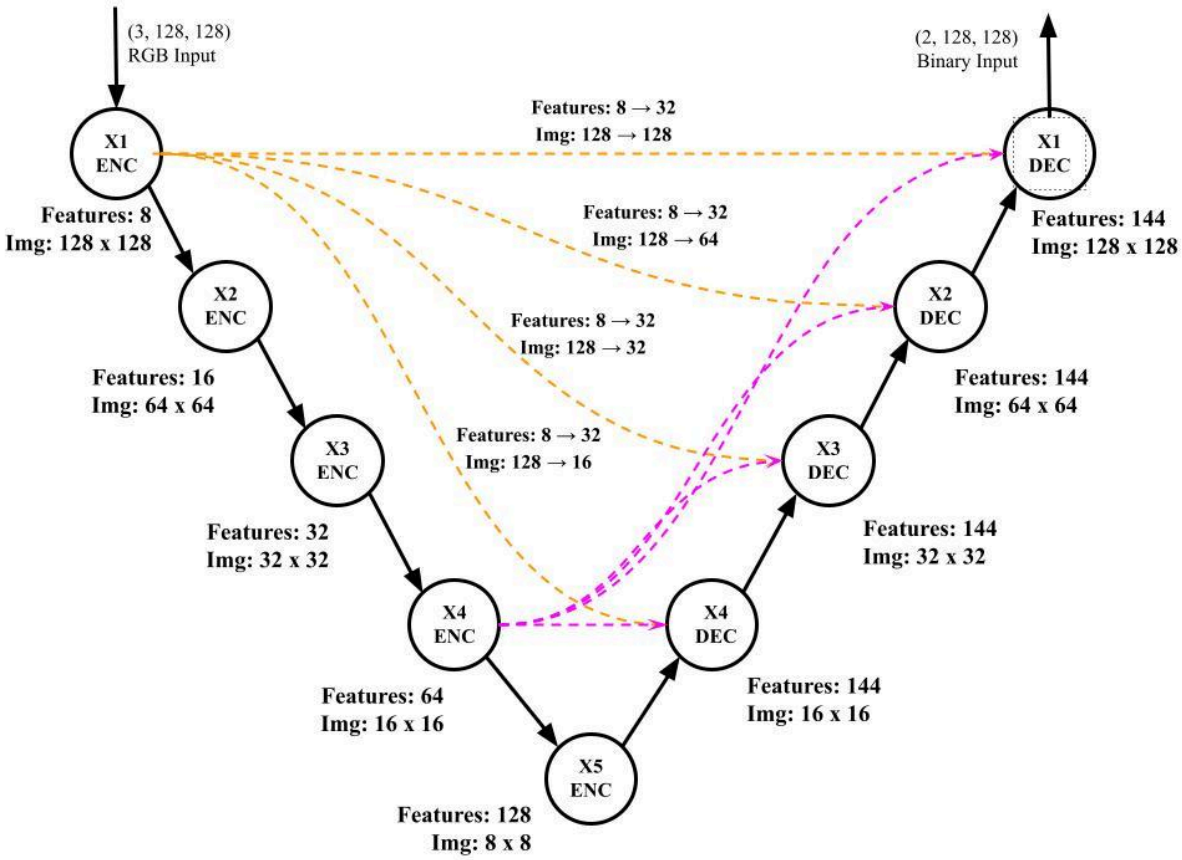

Figure 9: Thesis network architecture. Skip connections from X1_ENC and X4_ENC are included. Feature and image transformations are featured for X1_ENC.

All skip connections output a feature map of thickness 32; this is a concatenation of feature maps for CSB and SSB that are both 16 features thick. Image size differs between the input and output tensors for the given skip connections. If downsampling is needed, a MaxPool2D operation is performed over the images. If upsampling is needed, Bilinear Interpolation is used. Differing input and output features is much less of a problem because filters can easily adjust the output dimension.

Two binary output masks are utilized instead of one to allow the network to learn the target and background classes independently. During inference, the evaluated output will only concern the binary mask for the target class.

Discretization Policy

Going from alpha values to a final network architecture requires a well-defined ruleset. The policy used for this thesis closely follows Rethinking DARTs Search Space and Renovating a New Benchmark (Zhang et al., 2023). For each cell with input operations, the operation with the highest corresponding alpha value is chosen. For both the CSB and SSB blocks, all connections with alpha values above the mean alpha values for alphas within those blocks are kept. The SSB block in particular will retain a weighted combination scheme for discretized inputs.

The above paper details how discretization of DARTs architectures is an ongoing combinatorics research problem. They test a path tuning strategy detailed in Wang et al. (2021), which proposed a method that adds an additional Path Tuning (PT) phase after neural architecture search. The researchers found this discretization step incurred non-trivial computational time, and could even exceed resources needed in the search phase. Therefore, just as the authors of the above paper concluded, this thesis will not delve into path-tuning methods.

Results

Architecture Search Phase

Training was done on an A100 GPU with 40GB of VRAM. Throughout the training process, the VRAM usage never rose above 30 GB. Total training time is 2h 19m or ~0.1 GPU days. There are two output maps, one for the target class and one for the background class. Metrics like accuracy, IoU, and Dice are calculated on both of these feature maps. This is done for a more holistic view of network performance. However, during the discretion phase, only the target class will be used. The search network was evaluated on a held out test dataset.

Accuracy: 84.1%	IoU: 0.694	Dice: 0.804	Params: 3,105,752

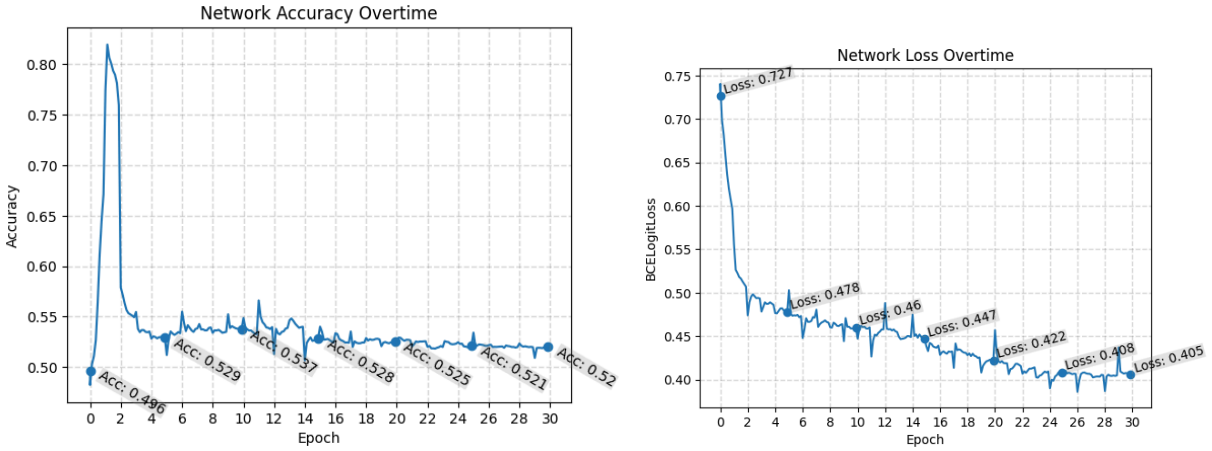

Figure 10: Search Network Accuracy and Loss Overtime

Initially, it looks like the accuracy and network loss is very unstable epoch-to-epoch. This is by virtue of the inner-epoch metric tracking. At the start of an epoch, the only data available to calculate metrics is the initial batch, however, as one gets further into the epoch, results are averaged out with all past in-epoch data. When taking this into account, network loss has a reasonable shape.

When looking at training accuracy, not only does it seem to decrease overtime, but it is substantially lower than the test accuracy. This is likely due to the Beta-Decay Regularization as it increases over the course of training.

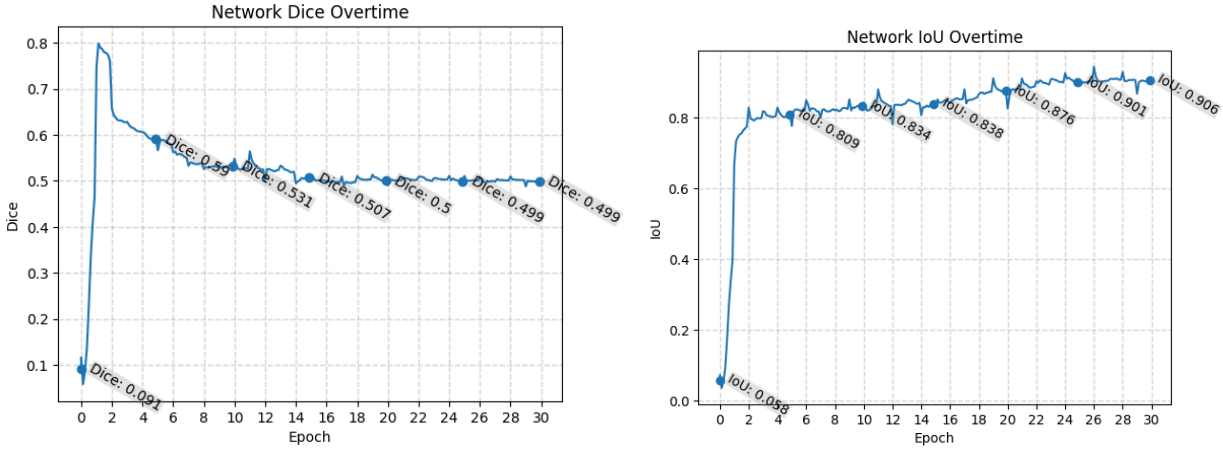

Figure 11: Search Network Dice and IoU Overtime

The Dice and IoU results are significant because the formulas of both imply the search network struggles with identifying true positive pixels. The Beta-Regularization looks to affect Dice in a more suppressive way than IoU. Both this and the accuracy dropoff can be attributed to Beta-Regularization because it forces the network to more heavily use weight-filled filters. If they are initially untrained, they will perform worse than their weight-free counterparts.

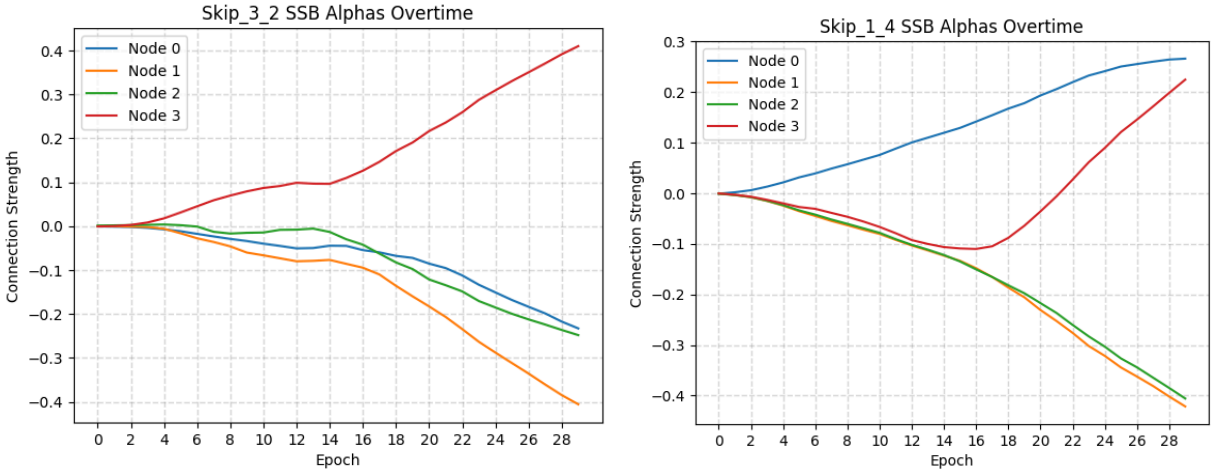

Figure 12: Skip_3_2 and Skip_1_4 SSB Alphas

Skip_1_4 connects the 1st Encoder node to the 4th, and most deep, decoder node. Thus, the high weight of the identity connection likely serves to communicate unaltered information about the original image that was made absent through the myriad of transformations along the encoding path.

Skip_3_2 connects the 3rd Encoder to the 2nd Decoder. This connection is between two fairly deep encoding and decoding nodes. Thus it makes sense Node 3, the node with the most transformations, would be prioritized. Priorities lay in transforming the input for use over anything else.

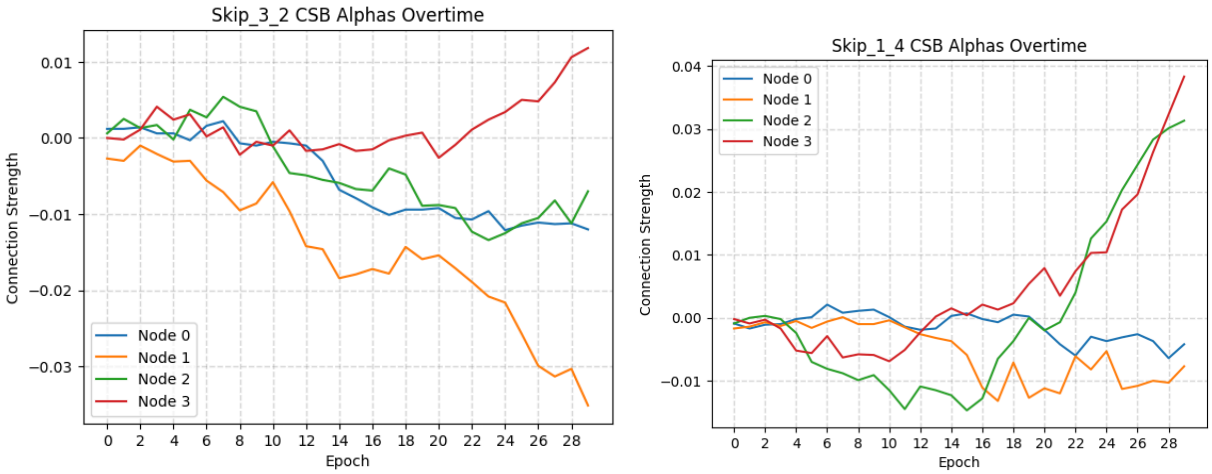

Figure 13: Skip_3_2 and Skip_1_4 CSB Alphas

Both the graphs for the CSB alphas are significantly more jagged compared to the SSB graphs. This could be due to the 1x1 convolution, which also features a trainable weight, being utilized as a combiner. Skip_3_2 replicates the same ordering of its SSB counterpart, while Skip_1_4 has a completely different order. This is good as it communicates these Polynary Operators are sufficiently different; however, it begs the question of what structural differences lead to a difference in output selection. CSB weights are also a magnitude smaller than the SSB weights.

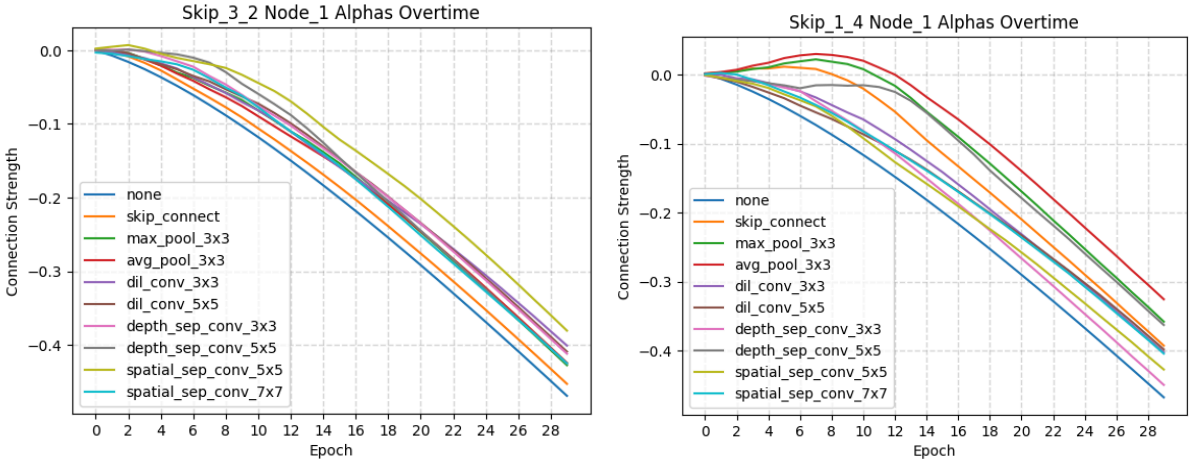

Figure 14: Skip_3_2 and Skip_1_4 Node 1 Alphas

The sharp decrease after initial increase is likely due to the Beta-Regularization, and the negative feedback it has with large operation values. Still, intuitively, the operation ordering makes sense. When going from a large image size to a small image size, max and average pooling help to preserve information. We see in Node 1 of the Skip_1_4 connection, average and max pool are the top choices. Once deep inside the network, preferred operations likely allow for a significant

amount of transformation of the input. This is also seen in Node 1 of the Skip_3_2 connection. In fact, the zero and identity operations are dead last in this cell.

Discretization Phase

When discretizing the network, it is interesting to note that as skip connections had inputs of deeper and deeper encoding layers, the operational choices leaned more towards weight-filled versus weight-free operations. This can potentially be due to the following reasons:

1) The inability for Beta–Regularization to sufficiently penalize weight-free operations.

2) The necessity to shrink dimensionality of a given image size, while still retaining sufficient original contextual information. This is further compounded by the usefulness, to deep layers, of untransformed image data. This untransformed image data can augment the already transformed image data received from the encoding backbone.

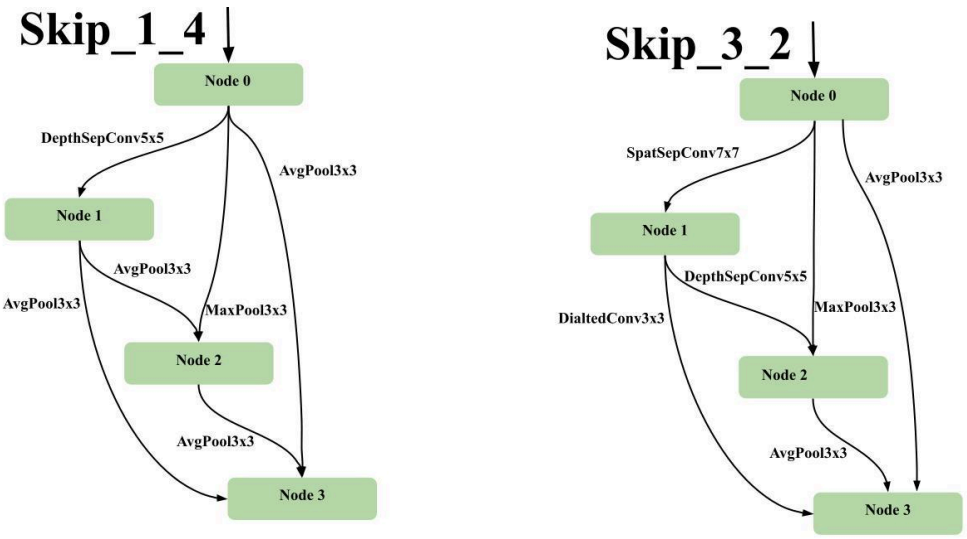

Figure 15: Top-1-Operation Selected Discrete Skip Connection Cells for Skip_3_2 and Skip_1_4. SSB and CSB blocks have been removed to decrease visual clutter.

The same hyperparameters were used for discrete training as was used for the search phase. The only exception is for an increase in epochs from 30 to 40. Additionally Rectified Adam with a decoupled weight decay of 0.0001 was used as an optimizer. Below are networks and papers also tuned on the HAM10000 lesion segmentation task for comparison. The top 3 operations were selected from every input node to be discretized in the final network. The features maps are twice as thick when compared to the search phase. In total, the final network has 3,646,080 parameters; this is almost 10x less than the base U-Net model.

Model Name	IoU	Dice	Acc	Source
U-Net	0.822	0.892	0.947	(Patil et al., 2024)
KNN	0.778	0.675	0.854	(Patil et al., 2024)
SVM	0.798	0.702	0.875	(Patil et al., 2024)
DermoSegDiff		0.943		(Bozorgpour et al., 2023)
MFSNet	0.902	0.906		(Basak et al., 2022)
SkinSAM	0.701	0.836		(Hu et al., 2023)
DARTs-UNET	0.51	0.65	0.67	This Paper

Unfortunately, just as with the search space training metrics on a validation set, during training, performance gradually reduced on the validation set. That would be classified as overfitting were it not for the performance of the network on the test set; this has almost twice the performance of the Accuracy, IoU, and Dice of the last training epoch.

Performance is significantly below the other methods mentioned in this paper. This could be due to a multitude of reasons such as DARTs performance collapse, overfitting, or to frigid operation selection that would need to be further investigated.

Discussion

Limitations

Path Tuning for discretization of the architecture is still an ongoing field of research and it is almost certain the policy here is nonoptimal.

While attempts have been made here to robustify the architecture searching process, if this result is to be used, the target dataset for a given task must be similar to the dataset used in this thesis.

Due to the automated nature of the parameter tuning, architectures as a result of DARTs are even more uninterpretable than traditional, manually tuned models.

Future Work

Computational complexity was by far the largest limiting factor for performing the architecture search. In an effort to fit the model training on a single GPU, both the image size and feature

depth had to be reduced substantially. In the future, distributed compute resources could allow a more full-featured search stage.

Discovering more properties about how the SSB and CSB blocks operate. A large majority of the behavior, due to the recent nature of the paper, is hard to characterize.

Explanation of metric and performance degradation could be due to the performance collapse problem of DARTs, an artifact of beta-regularization, or an unknown separate cause. It is hard to say at this moment.

Deriving a method to the madness. How can the selection of filters and weights be understood in an intuitive way? Does selection of architecture by DARTs reflect human intuition about architecture construction?

Application to path tuning of other networks, not just the traditional NasNet search space. A large majority of DARTs papers are constricted to improving DARTs as an algorithm versus application – and possible improvement – to other models.

Conclusion

This paper aims to tune the skip connections within U-Net using DARTs. DARTs relaxed the filter selection problem into a weighted combination of filters based on trainable parameters. To address the performance collapse issue, this thesis adds beta-regularization to the search process. To allow more flexibility in structure, this thesis both modifies the operation selection and cell structure. In the search phase, alphas are tuned via a bi-level optimization problem. These alphas can then go on to inform the discretization phase. During this phase the features were doubled in depth and the top 3 operations that came as inputs to any node within a cell were kept. Overall performance is disappointing, but may be due to an aspect of training rather than the selected architecture itself.

References

B. Moser, F. Raue, J. Hees and A. Dengel, "Less is More: Proxy Datasets in NAS approaches," in 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), New Orleans, LA, USA, 2022 pp. 1952-1960. doi: 10.1109/CVPRW56347.2022.00212

Bozorgpour, A., Sadegheih, Y., Kazerouni, A., Azad, R., Merhof, D. (2023). DermoSegDiff: A Boundary-Aware Segmentation Diffusion Model for Skin Lesion Delineation. In: Rekik, I., Adeli,

E., Park, S.H., Cintas, C., Zamzmi, G. (eds) *Predictive Intelligence in Medicine*. PRIME 2023. Lecture Notes in Computer Science, vol 14277. Springer, Cham. https://doi.org/10.1007/978-3-031-46005-0_13

C. Wang, A. Bochkovskiy and H. -Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, BC, Canada, 2023, pp. 7464-7475, doi: 10.1109/CVPR52729.2023.00721.

H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search,” in ICLR 2019, 2019.

Hritam Basak, Rohit Kundu, and Ram Sarkar. 2022. MFSNet: A multi focus segmentation network for skin lesion segmentation. Pattern Recogn. 128, C (Aug 2022). https://doi.org/10.1016/j.patcog.2022.108673

Huang H., Lin L., Tong R., Hu H., Zhang Q., Iwamoto Y., et al. . (2020). “Unet 3+: a full-scale connected unet for medical image segmentation,” in ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (Barcelona: IEEE), 1055–1059.

Hu, Mingzhe & Li, Yuheng & Yang, Xiaofeng. (2023). SkinSAM: Empowering Skin Cancer Segmentation with Segment Anything Model.

Liu D, Zhang D, Wang L, Wang J. Semantic segmentation of autonomous driving scenes based on multi-scale adaptive attention mechanism. Front Neurosci. 2023 Oct 19;17:1291674. doi: 10.3389/fnins.2023.1291674. PMID: 37928734; PMCID: PMC10620498.

O. Ronneberger, P. Fischer and T. Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation,” Medical Image Computing and Computer-Assisted Intervention, pp.234241, 2015.

Patil, S. N. ., & Patil, H. D. . (2023). Performance and Analysis of a U-Net Model for Automated Skin Lesion Segmentation. *International Journal on Recent and Innovation Trends in Computing and Communication*, *11*(11s), 508–518. https://doi.org/10.17762/ijritcc.v11i11s.8181

Wang, R., Cheng, M., Chen, X., Tang, X., and Hsieh, C.-J. Rethinking architecture selection in differentiable nas. In International Conference on Learning Representations, 2021b.

Y. Deng, Y. Hou, J. Yan and D. Zeng, "ELU-Net: An Efficient and Lightweight U-Net for Medical Image Segmentation," in *IEEE Access*, vol. 10, pp. 35932-35941, 2022, doi: 10.1109/ACCESS.2022.3163711.

Ye, Peng, et al. "β-darts: Beta-decay regularization for differentiable architecture search." *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022.

Zhang, Jiuling, and Zhiming Ding. "Rethink DARTS search space and renovate a new benchmark." *International Conference on Machine Learning*. PMLR, 2023.

Zoph, Barret, et al. "Learning transferable architectures for scalable image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2018

Z.W. Zhou, M.M.R. Siddiquee, N. Tajbakhsh and J.M. Liang, "UNet++: A Nested U-Net Architecture for Medical Image Segmentation," Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp: 3-11, 2018.