Divided We Stand:  
A Novel Residual Group Attention Mechanism for Medical Image Segmentation

Chaitanya Kaul, Nick Pears  
Department of Computer Science, University of York,  
Heslington, York, United Kingdom, YO10 5DD  
{ck807,nick.pears}@york.ac.uk

Suresh Manandhar  
NAAMII, Katunje, Bhaktapur  
Kathmandu, Nepal  
suresh.manandhar@naamii.org.np

Abstract

Given that convolutional neural networks extract features via learning convolution kernels, it makes sense to design better kernels which can in turn lead to better feature extraction. In this paper, we propose a new residual block for convolutional neural networks in the context of medical image segmentation. We combine attention mechanisms with group convolutions to create our group attention mechanism, which forms the fundamental building block of FocusNetAlpha - our convolutional autoencoder. We adapt a hybrid loss based on balanced cross entropy, tversky loss and the adaptive logarithmic loss to create a loss function that converges faster and more accurately to the minimum solution. On comparison with the different residual block variants, we observed a 5.6% increase in the IoU on the ISIC 2017 dataset over the basic residual block and a 1.3% increase over the resneXt group convolution block. Our results show that FocusNetAlpha achieves state-of-the-art results across all metrics for the ISIC 2018 melanoma segmentation, cell nuclei segmentation and the DRIVE retinal blood vessel segmentation datasets with fewer parameters and FLOPs. Our code and pre-trained models will be publicly available on GitHub to maximize reproducibility.

1. Introduction

A combination of attention mechanisms and deep learning has been shown to learn better feature representations in recent researches, proving them to be better feature extractors. Learning better feature extractors is the most important task a network can do, especially for attention based architectures, as the attention mechanisms are learnt over features extracted via convolutions. Hence, the better the feature, the better the output. Recent research has placed a lot of emphasis on optimizing convolutions, and in-turn learning better feature extractors, but until very recently, a very important technique went unnoticed. The AlexNet [17] architecture, due to memory constraints, divided their convolution kernels in each layers over two different GPUs. This resulted in an interesting effect where each kernel on a particular GPU learnt a explicit feature type. For the first convolutional layer, the kernels on one GPU learnt grayscale edge features, while the kernels on the second GPU learnt only colour specific information. This led to the initial empirical results showing the affect of grouping convolutions in convolutional neural networks. Filter groups learn a sparsely correlated set of features in each group, where each group learns a specific distinct feature of the input volume [12]. Even though filter grouping has shown promising improvements in the accuracy obtained by CNNs for various tasks, their use in state-of-the-art approaches is at a very early stage. There are a lot of unanswered questions in research involving filter grouping. We focus on the two which we consider are the most important - 1) No research to date incorporates attention mechanisms inside filter groups, and 2) Existing research has no grounding work that shows the best way to combine information learnt in each group. This is an important step as filter groups in their most basic form do not interact with each other. In architectures that propose to have certain cross talk between these groups, they define a fixed permutation of the groups rather than learning the best way to combine the information together.

To this end, we propose FocusNetAlpha, a deep learning architecture for medical image segmentation, that harnesses the power of grouped convolutions and combines it with a FocusNet [15] style attention mechanism to get a better performance than FocusNet, with less than half the number of parameters. We enhance the decoding using fine grained information from each decoder scale which helps improve the network’s decoding ability. We show that instead of using a single permutation to aid filter group interaction, such as proposed in techniques like ShuffleNet [30] and IGCNet [29], creating an embedding using a shared weighted function learns to embed the features into a space that is invariant to all possible permutations of the feature maps. We compare with state of the art architectures, namely, Wide...
DenseNet [11], BCDU-Net [1] and FocusNet [15] architectures and outperform them all.

The rest of the paper is organized as follows. Section 2 describes the current literature relevant to our problem. Section 3 describes the attention mechanism used in FocusNet Alpha. Section 4 describes FocusNet Alpha as well as the novel residual group attention block that we propose. We describe our loss in section 5 and experiments in section 6. The results are presented in section 7. We analyze our model’s efficiency compared to the other existing architectures in section 8 and conclude with section 9.

2. Related literature

ResNets: ResNets introduced the concept of feature reuse in deeper convolution layers. The initial residual architecture [8] was the first network to beat human level performance on imagenet. Since their inception, resnets have been constantly optimized and refined to produce better results. Residual blocks can be formally defined as $y = F(x, W) + x$ where $F(x, W)$ is the residual mapping that is learnt by the network and x are the features that are propagated forward by the skip connection. Residual learning allows feature reuse from previous layers that helps fight overfitting in deep networks. This technique results in the ability to train extremely deep networks. The residual block was optimized to produce better results in [9]. They showed how using a bn-relu-conv style residual block performed much better than the conv-bn-relu block via a series of ablation studies. One of the first works combining group convolutions with residual learning was the resneXt [27] architecture. The authors defined a new form of aggregated residual transformation, $F(x) = \sum_{i=1}^{C} T_i \{x\}$ where $T_i \{x\}$ can be any function approximation, and is generally a set of neurons. $T_i$ transforms the input $x$ by projecting it into a low dimensional embedding. $C$ is the cardinality, which is the total number of transformations that the network learns for a network layer. This aggregated transformation serves as the residual connection learnt in this architecture. The output $y$ in this case is given by, $y = x + \sum_{i=1}^{C} T_i \{x\}$ where $x$ is the the value that the aggregated transformation works on. Another spin on designing efficient skip connections that learn to propagate the most effective features from the encoder to the decoder using dense convolution blocks and bidirectional convolutional LSTMs. [15] learns a dual encoder decoder structure and employs one convolutional autoencoder to generate attention maps, and another to do fine grained image segmentation.

Convolutions and filter groups: To the best of our knowledge, one of the first works to explicitly show that grouping filter groups leads to learning better representations is deep roots [12]. They use a sparse connecting structure that resembles a tree root to reduce the number of parameters without any significant affect on the network accuracy. The impact of group convolutions was made apparent with resneXt [27] employing it at the heart of its methodology and doing impressively at the ILSVRC 2016 tasks. Depthwise separable convolutions [3] are another form of grouped convolutions that process channels in a feature volume individually before combining it together via a 1D convolution.

Filter group interaction: Due to the nature of grouped convolutions, the filter groups only look at a particular part of the input at a time. Seeing this as an apparent drawback, techniques such as ShuffleNet [30], FLGC [25] and IGCNets [29] have been proposed to aid interaction between filter groups and different parts of the feature inputs. To the best of our knowledge, all (except one [27]) existing techniques learn fixed permutations of the features to aid interaction with filter groups. ShuffleNet looks at only one permutation of the features for interaction with the filters, IGCNets fix a permutation matrix to perform permutations following their single primary and secondary group convolution blocks.
Attention in CNNs: The simplest form of attention networks are the spatial transformer networks [13] that learn the regions of interest from images with random clutter or noise. One of the first major visual attention approaches was a two level approach proposed in [26] where the images were first passed through an RCNN and selective search algorithms to generate proposals. A gating operation using softmax over the imagenet classes was used to get rid of low probability proposals. The remaining patches were then passed through a classifier which in their case was an SVM. The approach worked well on a subset of the imagenet dataset, but requires a large amount of computation as well as hyperparameter tuning. [23] proposed a combination of CNNs and RNNs to accumulate high multi-resolution glimpses of an image to make a final prediction. A large amount of techniques combine either reinforcement learning [7] or recurrent neural networks [18] along with multistage pipelines to create or process attention information making these techniques slow. Self attention mechanisms aim to learn context beyond a networks receptive field. SE-Nets [10] proposed to global average pool feature map information into a single vector creating a global representation, that was then autoencoded and passed through sigmoid gating to generate attention weights for each feature map in an activation output. The maps were then scaled via multiplication with these attention weights. SE Blocks have been extensively used in object detection [2], image segmentation [22] and scene classification [14] to name a few applications.

3. Attention Module

We discuss the base attention methodology (shown in figure 2). The idea is to first learn trainable weights which predict the probability of a pixel’s importance in the context of the output, and then recalibrate the global feature maps to learn the most important ones in an activation volume. Let us assume \( F \) to be the overall input to this block. Let \( F_1 \) be an intermediate feature map volume outputted from the interaction of some intermediate input volume \( I_1 \) with a set of trainable weights \( W_1 \) of a convolution filter. This \( F_1 \) is passed through a series of conv-ReLU operations, followed by a final 1x1 conv operation with a sigmoid operation, that gives per pixel probabilities for the volume. The learned output \( F \) can be denoted as,

\[
F = \sigma(\delta(I_1 * W_1), W_2)
\]
where * denotes a convolution between the filters and the learnt weights and $\delta$ denotes the (ReLU) non linearity. $W_1$ denotes the weights for the convolutions corresponding to feature extraction, while $W_2$ are the weights corresponding to the 1x1 convolution. In parallel, the input is also processed by a series of different conv-ReLU operations. The output of this block of operations has its feature maps recalibrated via the self attention mechanism. This is done by first squeezing the information into a 1D vector to get a global characteristic of each channel and then autoencoding this vector to get a per-channel probability. Let $I_2$ be the intermediate feature volume to this operation and $F_2$ be the overall final output of this operation. The squeezing of information is achieved via a global average pooling operation, given by,

$$g = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} I_2 c(i, j)$$

for a volume $I_2$ with dimensions $H \times W$ and channels $c$. $g$ is the channel-wise statistic. This output is then autoencoded by compressing the channels $c$ to a latent representation given by $\frac{g}{2}$ and attempting to reconstructing the input 1D vector. This is given by,

$$f_x = \sigma(\delta(g, W_k), W_r)$$

Here $W_k$ are the weights learnt before compression and $W_r$ are the weights learnt to map the input to a compressed latent space. The output from the autoencoder is passed through a sigmoid gating to get a channel-wise probability, which is used to scale each channel. This scaling is given by,

$$s = (f_x \cdot I_2)$$

where $($ denotes each weight value being multiplied with the entire slice of the feature map. This output is then multiplied by the per pixel probabilities obtained earlier that reweights every pixel inside the volume based on how much it contributes to the output, i.e,

$$P = F_1 \odot s$$

where $\odot$ is the hadamard product. The output is then recalibrated via the self-attention mechanism (denoted as SA).

$$F_2 = SA(P)$$

following which we add the residual feature map volume from before these series of operations as,

$$F_{out} = F + F_2$$

The attention block is end-to-end trainable and can be incorporated inside any deep learning system.

4. **FocusNetAlpha**

The encoder decoder structure of FocusNetAlpha works like any convolutional autoencoder should. The input is processed by an encoder into a bottleneck, which is then decoded into an output segmentation mask by the decoder. The main building block of our architecture is the residual group attention block (see section 4.1) that employs our novel attention methodology inside group convolutions for effective feature extraction. We address the problem of the relatively inferior decoding ability of segmentation architectures in FocusNetAlpha. We do this by creating a scheme that combines the output from each decoder scale to the final output which leads to superior performance. The output from each scale passes through a conv-bn-LeakyReLU-conv-sigmoid operation to give intermediate outputs which are upsampled if needed to the output size, and then concatenated together. The concatenated volume is then passed through a conv-bn-LeakyReLU-conv-sigmoid block to get the final output segmentation map. Downsampling in our architecture is done using the max pooling operation. We add skip connections from the encoder to the decoder rather than concatenating them. We upsample via repeating values in a kernel from a lower scale into an upsampled scale and letting convolutions learn their correct values. We use dropout in the bottleneck layer with a rate of 0.5. The receptive field of the first convolution kernel is 5x5. Following that, all convolutions kernels have a receptive field of 3x3 when they’re used for feature extraction, and 1x1 when they are used to learn attention weights (preceeding the sigmoid gating). The architecture can be visualized in figure 1.

4.1. **Residual group attention block**

The residual group attention block is shown in figure 3. The input to the block is a feature volume that is processed by a $1 \times 1$ convolution operation. The general form of the identity mapping is

$$M = \mathcal{C}_{i=1} P_i(x)$$

where $M$ is the output of the residual block, $\mathcal{C}$ denotes concatenation, and $P_i(x)$ is some transformation learnt by $r$ separate stackings of trainable neurons transforming some input $x$. Here, $r = 4$, as we divide this input features into groups of 4, to be processed by 4 separate convolution groups. Each group, alternatively, is responsible for learning the attention weights for the group to it’s right, and the next group learns the features that need to be extracted. The attention weights for each attention group are obtained via two bn-LeakyReLU-conv operations followed by a conv-sigmoid operation to get the per pixel probabilities. Each attention group transforms its input in the form

$$A_r = \sigma(W_n, \delta(x_r, W_k))$$
Figure 3: Our novel residual block that first employs pixelwise attention inside filter groups, followed by combining the groups via a permutation invariant embedding. The squeeze and excitation block then recalibrates the feature maps which is followed by the residual mapping.

Here $W_k$ and $W_a$ are the convolution weights and the attention weights respectively. $x_r$ is the $r^{th}$ group that is input into this block, and $\delta$ denotes the leakyReLU activation. The structure of this block is shown in figure 4b. The residual block contains two bn-LeakyReLU-conv operations followed by a skip connection which adds the features from the previous step to the residual block features. If the residual mapping is given by

$$O_r = x_r + F(x_r)$$

then the network learns this $F(x_r)$ using some weights $W_k$ as $F(x_r) = \delta(x_r, W_k)$. The block is shown in figure 4a. The output from the residual block is multiplied pointwise with the output from the attention block as $A = A_r \odot O_r$, weighting the pixels with a higher importance more prominently. The $\odot$ here denotes the hadamard product. The attention infused output for each group propagates further in the block and is convolved with a convolution block with a $1 \times 1$ receptive field and twice the number of filters. These intermediate filter maps are then concatenated together and passed through a final $1 \times 1$ convolution. The feature maps are then recaliberated using a squeeze and excitation operation which is followed by a residual connection.

5. Hybrid adaptive logarithmic loss

Accuracy alone is not a good measure of a network’s performance. Hence, we try to create a loss in such a way, that we can optimise various metrics at the same time. Importantly, we want a network that can predict true positive as efficiently as possible, and reduce their mis-classifications. This effectively translates to having a high recall. In order to optimize our architecture to have better recall, we adapt the balanced cross entropy loss with the tversky loss in a novel way to create our hybrid loss function. The loss is defined as, $L = (k)BaCE + (1 - k)TL$, where,

$$BaCE = \Omega p \log(\hat{p}) + (1 - \Omega)(1 - p) \log(1 - \hat{p})$$

$$TL = \sum_c (1 - TI)$$

where the subscript indicates a summation over the number of classes $c$ and,

$$TI = \frac{|G \cap P|}{|G \cap P| + \alpha |P \setminus G| + \beta |G \setminus P|}$$

To create a higher emphasis on the true positives, we select $\Omega = 0.7$. Generally, $\alpha = 0.3$, $\beta = 0.7$ works as the most optimal setting in TL, adding higher weights to optimize over false positives and false negatives, so we stick with those hyperparameter values. We weight the influence of both losses equally by setting $k=0.5$. In order to optimize the loss further, we use a function whose derivative gives a non linear response closer to the global minimum leading to heavy penalty for mis-classification. Hence, to mitigate the problem of pixel class imbalance and poor convergence close to the minimum, we use the adaptive logarithmic loss [16] for our problem. The loss is defined as,

$\text{ALL} - \text{HL}(x) = \begin{cases} \omega \ln(1 + \frac{|HL|}{\epsilon}) & |HL| < C \\ |HL| - C & \text{otherwise} \end{cases}$

where $C = \gamma - \omega \ln(1 + \frac{\gamma}{\epsilon})$. We observe that the default hyperparameters of this loss work as the optimal ones for our experiments. Hence, we set $\gamma = 0.1$, $\omega = 10.0$ and $\epsilon = 0.5$. We compare ALL-HL and HL to show that ALL-HL gives a 1.85% higher recall and is hence used. Results are summarized in appendix A.
6. Experiments

In this section, we conduct a series of tests to test the effect of each component on our architecture. We also compare against the various residual blocks. All experiments in this section are conducted on the ISIC 2017 skin cancer segmentation dataset [5]. We use the same pre-processing and data augmentation as in [15]. The official data splits are used.

6.1. Ablation study

Our ablation studies test the influence of our decoder and permutation invariance on our architecture. We start with comparing the performance of our decoder with a standard decoder, that just upsamples the bottleneck without any multiscale refinement (FocusNetAlpha-MD (minus multiscale decoding)), in the style of the U-Net decoder, but with our group attention block. The rest of the blocks in the architecture remain the same. We then replace all filter groups with a residual attention block, computing features, and attention weights inside the same block (FocusNetAlpha + ResA). The structure of this block is the same as figure 4a. The difference is that the feature extraction is followed by a conv-sigmoid block and multiplied with the skip connection. The same skip connection is also added to the attention scaled value in parallel. Our final study deals with concatenating the features of a group depthwise with the features of the group to its left (FocusNetAlpha + CH (ConcatHorizontal)). This means using only residual blocks inside groups which are concatenated with the groups to the right, instead of multiplying as in our proposed block. Our experiments show that the multiscale decoding helps improve performance of the architecture. Combining feature extraction with attention in a single block has a degrading effect on the performance which we believe happens due to the lack of features extracted in the blocks. Horizontal concatenation gives comparable performance with our methodology, though it does not improve the recall sufficiently.

6.2. Permutation invariance vs channel shuffle

Combining information from the different filter groups into one feature is an active area of research and many techniques have been proposed in the past that fix permutation matrices to aid channel interaction. The most prominent of these techniques is ShuffleNet that proposes a fixed permutation across the different channels. We compare our methodology with ShuffleNet as it is the current state of the art in combining filter group information. A 1D convolution operation shares weights across its kernels, hence mapping the different filter groups on to a symmetric function. This embedding created is invariant to all permutation of the filter groups. Hence, we hypothesise that a permutation invariant embedding such as that created by a 1D convolution, will always outperform hand crafting permutations via a channel shuffle/fixed permutation. This concept has found great success in creating permutation invariant representations of 3D data in the recent past [20] and is the technique incorporated in resneXt to combine filter groups. We run two different networks to this end, where one is FocusNetAlpha, and the other is the same architecture, but with the 1 × 1 convolution before the SE block changed to the shuffle block from ShuffleNet (FocusNetAlpha + channel shuffle (CS)). The results are shown in table 2 where it can be seen that defining fixed permutations considerably decreases the performance of the network versus learning a permutation invariant mapping. The jaccard and dice index degrade considerably, and the recall is also affected. Results on other metrics can be found in appendix B.

| Method              | Recall | DI    | JI    |
|---------------------|--------|-------|-------|
| FocusNetAlpha + CS  | 0.7991 | 0.8169| 0.7612|
| FocusNetAlpha       | 0.8222 | 0.8404| 0.7817|

Table 2: Permutation invariance vs channel shuffle in FocusNetAlpha on the ISIC 2017 dataset.

6.3. Comparison with resnet variants

Our next series of experiments observe the effectiveness of the different residual blocks. We train 5 different architectures with a FocusNetAlpha backbone to this end. The first is the generic identity (Basic) mapping [8] proposed in the initial resnet paper. This is followed by the no bottleneck full preactivation (Identity) mapping[9]. We then train two architectures using the ResNext group convolution blocks (ResneXt) and ResneXt + Squeeze and excitation group convolution blocks. All residual blocks listed are placed in the FocusNetAlpha architecture and the results are observed. The results are summarized in table 3. Results from [15] are also provided for reference as it is effectively an optimization over the identity block. We observe that for such shallow encoding decoding schemes with just a few layers, there is only a marginal increase in performance from the basic residual block to the identity mapping block. ResneXt group convolution block has a marginally
better recall than when combined with squeeze and excitation. We believe it to be due to the global attention mechanism squeeze and excitation incorporates, suppressing entire channels of information that. Pointwise attention combined with squeeze and excitation gives a superior recall and fixes this problem, which is what FocusNetAlpha does. FocusNetAlpha achieves a significantly better dice score as well as F1 score compared to other blocks which leads us to believe that it handles the prediction of true positives and true negatives better than the other residual blocks.

| Method      | Recall | DI    | JI    | F1    |
|-------------|--------|-------|-------|-------|
| Basic       | 0.7519 | 0.8023| 0.7261| 0.8009|
| Identity    | 0.7576 | 0.8034| 0.7281| 0.8012|
| ResNeXt     | 0.8145 | 0.8214| 0.7689| 0.8196|
| ResNeXt + SE| 0.8123 | 0.8206| 0.7761| 0.8229|
| Focusnet    | 0.7673 | 0.8315| 0.7562| 0.8191|
| FocusnetAlpha| 0.8222 | 0.8404| 0.7817| 0.8336|

Table 3: Comparing with different residual blocks on ISIC 2017 dataset.

7. Results

To show the superior performance of FocusNetAlpha, we compare how it fairs on 3 different benchmark datasets against state of the art network architectures. For all the listed experiments, the train-val-test split is constant and no data augmentation is used. As a preprocessing step, we scale all pixel values to fall between [0,1]. Our initial experiments suggest that mean pixel subtraction worsens the performance of the networks, so we choose not to use it as a preprocessing step. We convert the segmentation mask to binary by setting every pixel above the threshold of 0.5 to 1. The training details can be found in the supplementary materials (section 1.3). For the results tables, we highlight the best results in the table by bold and red, while the second best results are shown in blue. The validation dice index, validation jaccard index and validation loss plots are shown in figure 5 in the appendix.

7.1. ISIC 2018 segmentation

The ISIC 2018 skin cancer segmentation dataset [6] has become a major benchmark dataset for the evaluation of medical imaging algorithms. We use the 2594 images with corresponding ground truths localizing lesions on skin images containing melanoma. We divide these images into a training set of 1815 images, a validation set of 259 images, and the rest of the 520 images. The images are preprocessed as described earlier. The images in the dataset are 700 × 900 in dimension. We resize every images to a smaller 256 × 256 size, via an antialiasing downsampling technique, to be processed by the networks.

Table 4 summarizes our results for the experiments. FocusNetAlpha outperforms every architecture across all metrics significantly for the dataset with considerably fewer parameters and FLOPs (see section 8). We get a 4.6% higher JI over the next best model. The F1 score is also considerably higher showing a superior and balanced segmentation output. The ROC curves and sample good and bad segmentation outputs can be seen in appendix D.

7.2. Cell nuclei segmentation

We now shift our focus to segmentation of smaller regions inside images. For this, we use the cell nuclei segmentation dataset which was a part of the Data Science Bowl 2018. It contains 670 images which we divided into a training set of 540 and a validation set of 130. We resize all images to 256 × 256. For this task, we test the performance of our architecture by reducing the number of parameters (via reducing the number of filters per layer) for it in a way that it has less than 1 million FLOPs. We call this lightweight architecture FocusNet-α-Lite. We also reduced the number of parameters for the other architectures to account for the smaller size of this dataset so that we don’t overfit on it. Our results are summarized in table 5. We get results competitive with BCDU-Net even though we use 2.5 times lesser parameters and 10 times lesser FLOPs than that architecture. It should be noted that without the constraint we set on our model to be under 1 million FLOPs, we outperform all existing architectures with a slightly ‘bigger’ version of FocusNet-α-Lite which contains 2.32 million parameters and 1.2 million FLOPs. Results on other metrics as well as sample results are shown in appendix E.

7.3. Retinal blood vessel segmentation

For the third task, we see the effect of our architecture on fine grained pixel segmentation. We use the DRIVE retinal vessel segmentation dataset for this task. The dataset is small and contains 40 high resolution images. But given the fine grained analysis needed for it, we can sample smaller patches from the images to be processed by the networks. We sample 200000 patches around the field of view of the fundus images, from which we use 170000 for training and 30000 for validation. The extracted patches are of size 64 × 64. We record an improvement of approximately 3%, 1.4% and 3.4% over the recall, F1 score and the jaccard index respectively. FocusNetAlpha outperforms every architecture across all three metrics significantly. The detailed results table can viewed in appendix F.

8. Model efficiency

We compare the trade off between performance and efficiency for our model in this section. We focus on the ISIC
| Method          | Precision | Recall | Accuracy | DI | JI  | F1  |
|-----------------|-----------|--------|----------|----|-----|-----|
| FCN [24]        | 0.7176    | 0.8966 | 0.9011   | 0.7861 | 0.7013 | 0.7832 |
| U-Net [21]      | 0.7398    | 0.9043 | 0.9187   | 0.8167 | 0.7268 | 0.8004 |
| Wide UNet [31]  | 0.7439    | 0.9167 | 0.9234   | 0.8224 | 0.7334 | 0.8039 |
| R2U-Net [4]     | 0.7381    | 0.9122 | 0.9172   | 0.8271 | 0.7511 | 0.8097 |
| BCU-Net [1]     | 0.7576    | 0.9272 | 0.9337   | 0.8637 | 0.7665 | 0.8138 |
| UNet++ [31]     | 0.7516    | 0.8889 | 0.9249   | 0.8437 | 0.7435 | 0.8145 |
| Attention U-Net [19] | 0.7526  | 0.9286 | 0.9330   | 0.8741 | 0.7813 | 0.8214 |
| FocusNet [15]   | 0.7805    | 0.9328 | 0.9395   | 0.8676 | 0.7751 | 0.8499 |
| FocusNetAlpha   | 0.8322    | 0.9471 | 0.9447   | 0.9014 | 0.8271 | 0.8717 |

Table 4: Segmentation results on ISIC 2018 dataset. The results in bold red are the best results obtained on the dataset. Blue results denote the next best results. FocusNetAlpha outperforms every architecture with fewer parameters and FLOPs.

| Method          | Recall  | F1   | JI   |
|-----------------|---------|------|------|
| U-Net [21]      | 0.9052  | 0.8994 | 0.8310 |
| BCU-Net [1]     | 0.9078  | 0.9101 | **0.8410** |
| Attention U-Net [19] | 0.9019 | 0.8899 | 0.7984 |
| FocusNet [15]   | 0.8981  | 0.8998 | 0.8176 |
| FocusNet-α-Lite | **0.9139** | **0.9106** | **0.8386** |

Table 5: Segmentation results on the data science bowl 2018 dataset. The results in bold red are the best results obtained on the dataset. Blue results denote the next best results.

2018 dataset for FocusNetAlpha (see table 6), and on the nuclei segmentation dataset for FocusNet-α-Lite (see table 7). It can be seen that FocusNetAlpha outperforms every architecture in terms of having the lowest number of parameters and floating point operations. In terms of benchmark evaluation metrics, we observe a considerable increase in performance compared to the state of the art. FocusNet-α-Lite gets extremely competitive results with an architecture that has almost 2.5 times more parameters than it’s nearest competitor (BCDU-Net), which it achieves with over 10 times less FLOPs. This leads us to believe that our architecture provides one of the best trade-offs between model complexity and performance given the set of architectures we have evaluated against.

9. Conclusions

We proposed an extremely efficient and accurate medical image segmentation architecture, FocusNetAlpha, based on our novel residual group attention block which outperforms state of the art architectures. We also propose an extremely lightweight variant of this architecture that performs at par with architectures almost 2.5 times its size. We adapt the tversky loss and balanced cross entropy loss in the adaptive logarithmic loss setting to boost performance over true positives and true negatives to get more well rounded segmentations. We show the trade-off between model complexity and performance for various architecture, where our architecture consumes the least parameters and FLOPs while giving the best results.

References

[1] Reza Azad, Maryam Asadi-Aghbolaghi, Mahmood Fathy, and Sergio Escalera. Bi-directional convlstm u-net with densely connected convolutions, 2019.
[2] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han
Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. CoRR, abs/1904.11492, 2019. 3

[3] Francois Chollet. Xception: Deep learning with depthwise separable convolutions, 2016. 2

[4] Md. Zahangir Alom et al. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. CoRR, abs/1802.06955, 2018. 2, 8

[5] Noel C. F. Codella et al. Skin lesion analysis toward melanoma detection: A challenge at the 2017 international symposium on biomedical imaging (isbi), hosted by the international skin imaging collaboration (ISIC). CoRR, abs/1710.05006, 2017. 6

[6] Noel C. F. Codella et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (ISIC). CoRR, abs/1902.03368, 2019. 7

[7] Jianlong Fu, Heliang Zheng, and Tao Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4476–4484, 2017. 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015. 2, 6

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In European conference on computer vision, pages 630–645. Springer, 2016. 2, 6

[10] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507, 7, 2017. 3

[11] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. CoRR, abs/1608.06993, 2016. 2

[12] Yani Ioannou, Duncan P. Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving CNN efficiency with hierarchical filter groups. CoRR, abs/1605.06489, 2016. 1, 2

[13] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. CoRR, abs/1506.02025, 2015. 3

[14] M. Juneja, A. Vedaldi, C. V. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In 2013 IEEE Conference on Computer Vision and Pattern Recognition, pages 923–930, June 2013. 3

[15] C. Kaul, S. Manandhar, and N. Pears. Focusnet: An attention-based fully convolutional network for medical image segmentation. In 2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019), pages 455–458, April 2019. 1, 2, 6, 8

[16] Chaitanya Kaul, Nick Pears, and Suresh Manandhar. Penalizing small errors using an adaptive logarithmic loss, 2019. 5

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. 1

[18] Xiao Liu, Tian Xia, Jiang Wang, Yi Yang, Feng Zhou, and Yuanqing Lin. Fully convolutional attention networks for fine-grained recognition, 2016. 3

[19] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kansaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glover, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018. 2, 8

[20] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. CoRR, abs/1612.00593, 2016. 6

[21] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention (MICCAI), volume 9351 of LNCS, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]). 2, 8

[22] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. Concurrent spatial and channel squeeze & excitation in fully convolutional networks, 2018. 3

[23] Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization, 2014. 3

[24] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. IEEE Trans. Pattern Anal. Mach. Intell., 39(4):640–651, Apr. 2017. 8

[25] Xijun Wang, Meina Kan, Shiguang Shan, and Xilin Chen. Fully learnable group convolution for acceleration of deep neural networks. CoRR, abs/1904.00346, 2019. 2

[26] Tianjun Xiao, Yichong Xu, Kuiyuan Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions for deep neural networks. CoRR, abs/1707.02725, 2017. 1, 2

[27] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Fully convolutional attention networks for fine-grained image classification. CoRR, abs/1411.6447, 2014. 3

[28] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. CoRR, abs/1611.05431, 2016. 2

[29] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. CoRR, abs/1605.07146, 2016. 2

[30] Ting Zhang, Guo-Jun Qi, Bin Xiao, and Jingdong Wang. Interleaved group convolutions for deep neural networks. CoRR, abs/1707.02725, 2017. 1, 2

[31] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. CoRR, abs/1707.01083, 2017. 1, 2

[32] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Izadi, Leonidas J. Guibas, and Kaiming He. Concurrent spatial and channel squeeze & excitation in fully convolutional networks, 2018. 3

[33] Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, Ben Glover, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas, 2018. 2, 8

A. Hybrid adaptive logarithmic loss vs hybrid loss

We compare the Adaptive logarithmic loss with our hybrid function formulation (ALL-HL), with the vanilla hybrid loss (HL). We run two experiments to this end the re-
sults for which are shown in table 8. In the results, of particular interest to us is the value of the recall that we are effectively optimizing over by weighting our true positives higher. Our loss improves the recall by 1.85% which in turn gives a higher dice index. We use the ALL-HL for all our experiments henceforth.

B. Permutation invariance vs channel shuffle

Further empirical results are presented in table 9.

C. Training details

All our experiments are trained with the adaptive logarithmic loss using our hybrid loss (ALL-HL) strategy. The experiments are conducted in keras using a tensorflow backend. The batch size for all experiments is kept constant at 8. 1 Nvidia GTX 1080Ti is used to train the FCN, U-Net, Wide U-Net, Attention U-Net, UNet++, R2U-Net and FocusNetAlpha architectures. BCDU-Net and FocusNet are trained on 2 Nvidia GTX 1080Tis. We have optimized every architecture trained for these experiments by first running them with different learning rates to see their behaviour, and then creating a customized learning rate schedule for each experiment, for each architecture. All architectures were trained for a maximum of 50 epochs and the best model weights were saved by monitoring the validation loss. No early stopping was used.

D. ISIC 2018 segmentation

The ROC curves for the architectures are shown in figure 6. We defined an accuracy threshold and sampled images higher and lower than the threshold to visualize our predictions and qualitatively evaluate the performance of our network. Figure 8 shows the results which FocusNetAlpha found easy to segment. Figure 9 shows the more challenging results. Even on the more challenging set of images, it can be seen that FocusNetAlpha (column 5) performs significantly better than Attention U-Net (column 3) and FocusNet (column 4). It is visually clear that even under conditions that FocusNetAlpha finds challenging, it provides better segmentations that the other architectures.

E. Cell nuclei segmentation

Figures 10 and 11 show the qualitative results, validating FocusNetAlpha’s competitive performance compared to BCDU-Net and FocusNet, and superior performance compared to Attention U-Net. The extended version of the table from the paper is shown in table 10.

F. Retinal blood vessel segmentation

The results are summarized in detail in table 11.
| Method               | Precision | Recall   | Accuracy | DI     | JI     | F1   |
|----------------------|-----------|----------|----------|--------|--------|------|
| FocusNetAlpha (HL)   | 0.7921    | 0.8037   | 0.9276   | 0.8214 | 0.7631 | 0.8262 |
| FocusNetAlpha (ALL-HL)| 0.8002    | 0.8222   | 0.9349   | 0.8404 | 0.7817 | 0.8336 |

Table 8: ALL-HL gives better results compared to HL. Dataset used is ISIC 2017.

| Method               | Precision | Recall   | Accuracy | DI     | JI     | F1   |
|----------------------|-----------|----------|----------|--------|--------|------|
| FocusNetAlpha + CS   | 0.7844    | 0.7991   | 0.9214   | 0.8169 | 0.7612 | 0.8184 |
| FocusNetAlpha        | 0.8002    | 0.8222   | 0.9349   | 0.8404 | 0.7817 | 0.8336 |

Table 9: Permutation invariance vs channel shuffle in FocusNetAlpha on the ISIC 2017 dataset.

(a) Val DI vs Epochs on (ISIC 2018 segmentation).
(b) Val JI vs Epochs on (ISIC 2018 segmentation).
(c) Val ALL-HL vs Epochs (ISIC 2018 segmentation).
(d) Val DI vs Epochs (cell nuclei segmentation).
(e) Val JI vs Epochs (cell nuclei segmentation).
(f) Val ALL-HL vs Epochs (cell nuclei segmentation).

Figure 5: Visualizing the different validation curves on ISIC 2018 segmentation (top row) and cell nuclei segmentation (bottom row) datasets.

| Method               | Precision | Recall   | Accuracy | F1     | DI     | JI   |
|----------------------|-----------|----------|----------|--------|--------|------|
| U-Net                | 0.8976    | 0.9052   | 0.9604   | 0.8994 | 0.9075 | 0.8310 |
| BCU-Net              | 0.9024    | 0.9078   | 0.9728   | 0.9101 | 0.9129 | 0.8410 |
| Attention U-Net      | 0.8782    | 0.9019   | 0.9672   | 0.8899 | 0.8876 | 0.7984 |
| FocusNet             | 0.9016    | 0.8981   | 0.9697   | 0.8998 | 0.8961 | 0.8176 |
| FocusNet-α-Lite      | 0.9173    | 0.9139   | 0.9768   | 0.9106 | 0.9077 | 0.8386 |

Table 10: Segmentation results on the cell nuclei segmentation dataset. The results in bold red are the best results obtained on the dataset. Blue results denote the next best results.
(a) The ROC curves for different architectures on ISIC 2018.  
(b) The same ROC curves zoomed in top left for better viewing.

Figure 6: Receiver operator characteristics (ROC) curves for the ISIC 2018 dataset.

(a) Cell nuclei segmentation ROC curves.  
(b) Same ROC curves zoomed in top left.

Figure 7: Receiver operator characteristics (ROC) curves for the cell nuclei segmentation dataset.

| Method       | Recall   | F1       | JI      |
|--------------|----------|----------|---------|
| U-Net        | 0.7614   | 0.7898   | 0.7219  |
| BCDU-Net     | 0.7991   | 0.8165   | 0.7503  |
| Attention U-Net | 0.7861 | 0.8033   | 0.7517  |
| FocusNet     | 0.7892   | 0.8097   | 0.7498  |
| FocusNetAlpha | 0.8288  | 0.8306   | 0.7854  |

Table 11: Segmentation results on the DRIVE retinal blood vessel segmentation dataset. The results in bold red are the best results obtained on the dataset. Blue results denote the next best results.
Figure 8: ISIC 2018 (good segmentation results). From left, column 1 is the input image, column 2 is the ground truth and columns 3, 4 and 5 are the outputs of Attention U-Net, FocusNet and FocusNetAlpha respectively.
Figure 9: ISIC 2018 (bad segmentation results). From left, column 1 is the input image, column 2 is the ground truth and columns 3, 4 and 5 are the outputs of Attention U-Net, FocusNet and FocusNetAlpha respectively. FocusNetAlpha provides better segmentation results even under challenging scenarios.
Figure 10: Cell nuclei segmentation (good segmentation results). From left, column 1 is the input image, column 2 is the ground truth and columns 3, 4, 5 and 6 are the outputs of BCDU-Net, Attention U-Net, FocusNet and FocusNetAlpha respectively.
Figure 11: Cell nuclei segmentation (bad segmentation results). From left, column 1 is the input image, column 2 is the ground truth and columns 3, 4, 5 and 6 are the outputs of BCDU-Net, Attention U-Net, FocusNet and FocusNetAlpha respectively.