Grammar-Guided Evolution of the U-Net

Mahsa Mahdinejad[1 [0000−0003−4288−3991]], Aidan Murphy[2 [0000−0002−6209−4642]], Michael Tetteh[1 [0000−0003−4288−3991]], Allan de Lima[1 [0000−0002−1040−1321]], Patrick Healy[1 [0000−0002−3824−7442]], and Conor Ryan[1 [0000−0002−7002−5815]]

[1] University of Limerick, Limerick, Ireland
{Mahsa.Mahdinejad,Michael.Tetteh,Allan.Delima,Patrick.Healy,Conor.Ryan}@ul.ie
[2] University College Dublin, Dublin, Ireland
Aidan.Murphy@ucd.ie

Abstract. Deep learning is an effective and efficient method for image segmentation. Several neural network designs have been investigated, a notable example being the U-Net which has outperformed other segmentation models in different challenges. The Spatial Attention U-Net is a variation of the U-Net, which utilizes *DropBlock* and an attention block in addition to the typical U-Net convolutional blocks, which boosted the accuracy of the U-Net and reduced over-fitting. Optimising neural networks is costly, time-consuming and often requires expert guidance to determine the best mix of hyper-parameters for a particular problem. We demonstrate that grammatical evolution (GE) can be used to create U-Net and Spatial Attention U-Net architectures and optimise its choice of hyper-parameters. Our results show improved performance over state-of-the-art models on the Retinal Blood Vessel problem, increasing both *AUC*, from 0.978 to 0.979 and *Accuracy*, from 0.964 0.966, from the base models. Crucially, GE can achieve these improvements while finding a model which is 10 times smaller than the base models. A smaller model would enable its use in smart devices, such as smart phones, or in edge computing.

Keywords: Deep Learning, Evolutionary Algorithm, Image Segmentation, Grammatical Evolution

1 Introduction

Specialists and clinicians are increasingly using image processing and machine learning technology to help them investigate and identify anomalies. Diabetes is a serious and widespread disease that, if untreated, can cause blindness [5] or death [19]. Damage and a swelling of blood vessels in the eye are the main contributors to diabetic retinopathy (DR), a form of blindness brought on by diabetes. One of the main causes of blindness is specifically the formation of harsh exudates (a type of fluid) around the fovea (a crucial portion of the neurosensory retin). Early diagnosis and laser photocoagulation, however, is able to slow the growth of DR in the retina. Early detection of DR requires a manual inspection of retinal images, as it does not become evident until after a diabetes diagnosis.

The Retinal Blood Vessel Segmentation task is the process of extracting blood vessels from retinal images to diagnose DR [27].

The identification of blood vessels in images of eyes is a difficult task, however cutting-edge Neural Networks (NNs) approaches are employed to resolve this issue. A convolutional NN (CNN) [13], an improved variation of the traditional NN, has emerged as the most successful method for tackling image segmentation problems. Due to the availability of powerful supercomputers for training larger, more complex CNNs, a wider range of image processing tasks are being effectively accomplished by CNNs. These achievements have come at the expense of complexity, despite the fact that many typical CNN architectures are so expensive to train. As a result, developing a CNN architecture, training it, testing it and selecting the appropriate hyper-parameters for a task are becoming increasingly challenging and costly. We use Grammatical Evolution (GE) to find the best CNN architecture and parameters automatically [22].

In contrast to our earlier work [20,12] and [16], we use GE to choose both the hyper-parameters and evolve the architecture of a CNN rather than a using a Genetic Algorithm (GA) [11] which optimises the hyper-parameters.

Our results show that GE prefers a Spatial Attention U-Net (SA-UNet) over a U-Net and that the SA-UNet evolved achieves higher *Accuracy* than manual designs. The SA-UNet representation improves the U-Net by adding an attention block and also using *DropBlock* instead of *DropOut*. Further, a SA-UNet reduces over-fitting and concentrates on important aspects of the image while suppressing unnecessary ones. In other words, the SA-UNet cleans up the images by removing noise and unimportant background elements.

To explore if we can improve the performance of our models one step more, we decide to change our fitness function in the second sets of experiments.

We first create a grammar and conduct experimentation on it. Results indicated that the performance of the SA-UNet exceeded that of the U-Net, so further experimentation was conducted with special attention fixed in each model. Lastly, with real world applications in mind, we investigate the use of different fitness functions, first *Val-accuracy* followed by *F2-Score*, on the evolution of models.

In Section 2, we provide background information on our methodology. In Section 3, the experimental setup is explained and discussed in depth , and Section 4 provides the results. Section 5 presents our conclusions as well as our ideas for further work.

2 Background

2.1 Image Segmentation

A significant field of research in computer vision and machine learning is image segmentation [17]. Medical image processing, autonomous driving, and urban navigation all demand precise, dependable, and efficient segmentation algorithms. In many clinical processes, medical image segmentation, such as diagnosis, planning, and executing treatments is a crucial step. These visual analyses

are often carried out by trained therapists. Therefore, creating reliable and strong image segmentation algorithms is a key requirement for medical image analysis. Image segmentation is the process of breaking down a digital image into many subgroups in order to reduce the image's complexity and make future processing or analysis easier.

2.2 Convolutional Neural Networks

A branch of machine learning known as deep learning (DL) focuses on artificial neural networks (ANNs), particularly networks with many complex layers. CNNs have made significant strides in a variety of pattern identification fields, from voice recognition to image processing. CNN's greatest benefit is their ability to decrease the number of parameters in ANNs by using filters and kernels in convolution blocks [2]. This success has encouraged academics and developers to think about ever-larger models in order to complete difficult tasks that were previously unsolvable with conventional ANNs. However, solving a problem a CNN requires that the images do not have spatially dependent features. Another key aspect of CNNs is their ability to identify and use abstract features when inputs pass through many deep layers.

2.3 Autoencoders

An autoencoder is an unsupervised learning strategy for NNs that trains the network to ignore signal "noise" in order to develop efficient data representations. This is referred to as encoding. Autoencoders have been used on many different segmentation tasks [7]. They can be divided into two main parts, an encoder and a decoder 1. The encoder is the portion which encodes input data from the train-validate-test set into a substantially smaller representation than the input data (often by several orders of magnitude). Convolutional blocks are used in the encoder, which are followed by pooling blocks. The middle section contains the compressed knowledge representations. The decoder, is the part which assists the network in "decompressing" knowledge representations and reconstructing data from its encoded form. Finally, the output is contrasted with a ground truth. Data denoising [9] and dimensionality reduction [26] for data visualization are two practical applications of autoencoders.

2.4 The U-Net and SA-UNet

The most recent strategies for addressing the blood vessel segmentation challenge are all NN-based methods. The original U-Net was developed by [21]. According to the authors, the U-Net was a completely CNN variant that could provide good prediction using a minimal training dataset. A U-net design consists of a U-shaped (Fig. 2) convolutional autoencoder that has extra connections between the encoder and decoder components. The contracting path and the expanding path, which are located on the left and right sides of a U-Net, respectively,

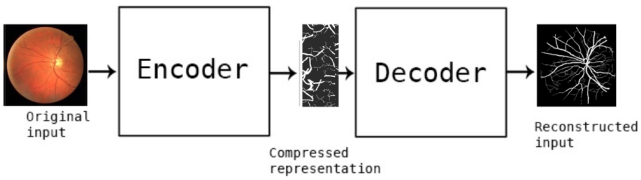


Fig. 1. An auto-encoder acting on an image.

separate a U-Net into two pieces. The expanding path uses up-sampling, whereas the contracting path uses down-sampling to extract features from input images.

We have previously optimized a U-Net design and produced a smaller model than the standard U-Net [12,20]. This allowed the model to be trained more quickly and affordably, and as a result, the models were more suited to running on compact or embedded systems. However, this more compact form came at the expense of accuracy. This trade-off is no longer justifiable when the accuracy gap between the models widens as newer, state-of-the-art models have improved performance on benchmarks to ever-higher levels. To improve the performance of the network, several U-Net modifications have been suggested. The design of a U-Net has undergone some changes, including the Dual Encoding U-Net (DEU-Net) [25], which employs two encoders and attention blocks. The U-Net++ [29] is another example, which seeks to close the feature map gap between the encoder and decoder, as well as other modifications.

In our experimentation, we also use a spatial attention U-Net (SA-UNet) [10], an improved version of a U-Net. An SA-UNet has two key differences compared to a U-Net, as shown in Fig. 2. First, it uses spatial attention at the bottom of a U-Net [28], in the middle section, and second, it uses *DropBlock* rather than *DropOut* [8] (see below).

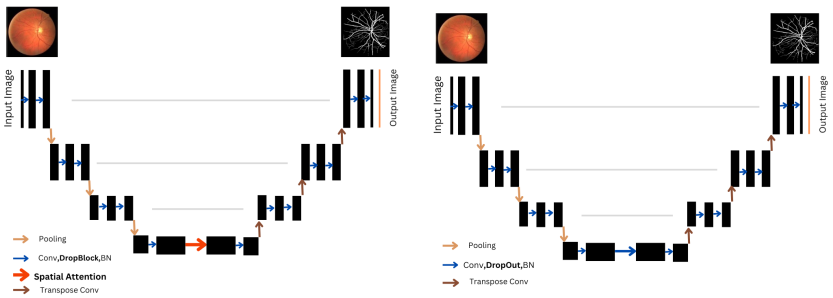


Fig. 2. Comparison SA-UNet (left) and U-Net (right) architecture.

Small training sets are frequently encountered in supervised machine learning applications. This might be due to a lack of available data, the cost of gathering, or a variety of other factors. Most CNN models require massive quantities of data to learn well to avoid over-fitting [6] and exhibit poor performance when trained on small datasets.

A regularization approach called *DropOut* [4] ignores or "drops out" some neurons stochastically during the training process. The *dropout rate* is a hyper-parameter dictating the chance of training a particular node in a layer. A value of 0.0 denotes no drop out, while 1.0 drops out the entire output layer. A suitable dropout value in a hidden layer is usually between 0.4 and 0.6.

DropBlock is another CNN regularization approach that, like *DropOut*, ensures that no units are dependant on each other throughout the training process by changing the input's units to 0. It differs from *DropOut* by removing continuous areas from a layer's feature map rather than individual random units. This difference is demonstrated in Fig. 3.

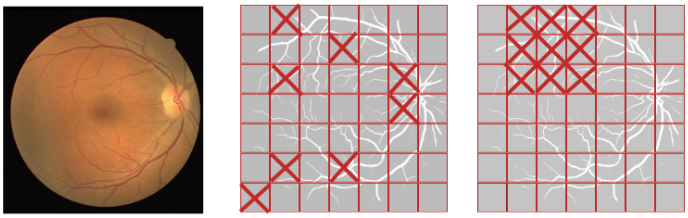


Fig. 3. An example contrasting *DropOut* and *DropBlock's* methods for disregarding units in an input image.

.

DropBlock has been shown to efficiently minimize network over-fitting, with small training datasets. *DropBlock* needs the user to provide the values of two parameters: *block-size*, which defines the amount of pixels in each block, and *keep-probability*, which determines the likelihood of the block being shut down.

The lack of contrast between the blood vessel region and the backdrop in retinal fundus images is a significant difficulty in retinal segmentation. Much effort has been done to address this lack of differentiation. *Attention* has been demonstrated to be one of the most powerful approaches [24] by translating a query and a collection of crucial data into output to help the network learn better. In order to provide effective outcomes, the model has to learn structural information, which is possible with spatial attention. This information is learned by emphasizing key units and minimizing background noise. The channel axis is used to apply the max-pooling and average-pooling blocks first, which are then concatenated to create an effective feature detector.

2.5 Evolutionary Algorithms

Population-based optimization methods called evolutionary algorithms (EAs) are motivated by natural selection.

A population of solutions is first created in an EA experiment along with a fitness function. This population is then evolved across a number of generations, with the strongest individuals surviving and carrying their traits to next generations, while the weaker individuals will disappear. An individual in our experiments is a CNN architecture and its corresponding hyper-parameters.

2.6 Grammatical Evolution

GE is popular variant of Genetic Programming (GP). It is an evolutionary method that uses a grammar to map a string, called a genotype, to an executable program, called a phenotype. GE has been used to a variety of test problems and real-world applications, with satisfactory results [15,3]. GE often uses a context free grammar, written in Backus-Naur-Form (BNF). A BNF grammar, G, is represented by the four-tuple $< N, T, P, S >$, where N is a set of non-terminals, T is a set of terminals, P is a set of production rules that map the components of N to T, and S (a member of N) is the start symbol.

Deciding on a suitable EA is challenging. Our previous works used a GA for optimising and tuning hyper-parameters of our CNN. GE has some advantages compared to the GA [18], as it gives more freedom in selecting hyper-parameters which can lead to more complex or simpler models, as seen in Figs.9 and 10, when compared to a fixed-length GA representation. Furthermore, GE can modify the architecture of CNNs and tune the hyper-parameters. The use of a grammar simplifies the addition and expansion of hyper-parameters, enabling solving more difficult tasks on complex networks. We explain our proposed grammar in detail in Section 3.4.

3 Experimental setup

Our experiments started first by using *Val-Accuracy* as the fitness function and asking GE to find both the best architecture and corresponding hyper-parameters.

After analysing the results of the first set of experiments, it was decided to run the next set of experiments with the attention layer fixed in the model, as a SA-UNet architecure. In the final sets of experiments, we modified the fitness function to *F2-Score* to see if it can improve the model's performance.

3.1 Dataset

In order to facilitate comparative study on blood vessel segmentation in retinal images for the diagnosis, screening, treatment, and assessment of many disorders including diabetes, the DRIVE (Digital Retinal Images for Vessel Extraction) [23] database was established. Experts identify and employ morphological characteristics of retinal blood vessels such as length, width, texture, and

branching patterns. Research into the relationship between vascular deformability and hypertensive retinopathy, vessel diameter measurement in connection to hypertension diagnosis, and computer-assisted laser surgery may all benefit from automatic identification and study of the vasculature. The retinal vascular branch has also been demonstrated to be particular to each individual and may be used for biometric identification. Two sets of 20 images each were created from the 40 total images, a training set (Fig. 4) and a test set (Fig. 5). After applying an augmentation method on the training set we had 260 total images which we randomly picked 26 to create the validation set.

For the training images, there is only one manual segmentation of the vasculature available. The test cases include two manual segmentations. While the other can be used to compare computer-generated segmentation to those made by a human observer, the first serves as the standard. Additionally, a mask image showing the region of interest is provided for each retinal scan. All human observers who manually segmented the vasculature were led and instructed by an experienced ophthalmologist.

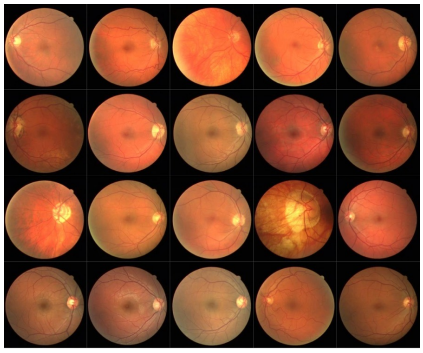


Fig. 4. DRIVE training images.

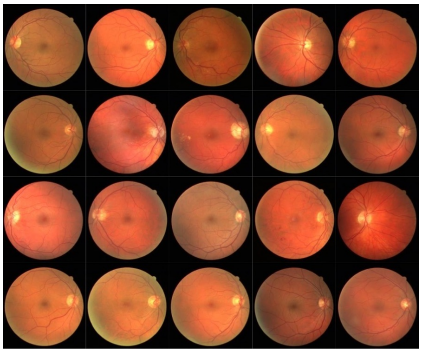


Fig. 5. DRIVE test images.

3.2 Hardware and Software

The experiments were carried out on a single system configured with an Nvidia Quadro RTX 8000 GPU. The networks were trained using TensorFlow [1] and GRAPE (Grammatical Algorithms in Python for Evolution) [14] was the GE framework used.

3.3 Evaluation Metrics

Our models were evaluated using *Recall*, *Specificity*, area under the curve (AUC), positive predictive value (*Precision*), negative predictive value (NPV), *F2-Score* and, Matthew's Correlation Coefficient (MCC). We demonstrate how different evaluation rates work in the segmentation process in Fig. 6.

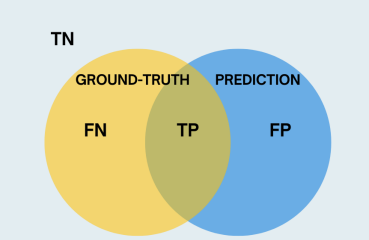


Fig. 6. Example of evaluations in segmentation.

For example, a True-Positive (TP) is when a pixel's value correctly matches the predicted value . False-Positive (FP) pixels are those that are found in the prediction but are not part of the ground truth, and so on. *Recall* is the rate of TPs which can also be considered to be the test's sensitivity to noticing small changes. *Specificity* is the rate of TNs. The relevant formulae are:

$$Recall = \frac{TP}{TP + FN}$$
$$Specificity = \frac{TN}{TN + FP}$$
$$Precision = \frac{TP}{TP + FP}$$
$$NPV = \frac{TN}{TN + FN}.$$

MCC 3.3, is a measure for calculating the difference between predicted and actual results. *MCC* is a reliable statistical rate that returns a high score only if the prediction was successful in each of the four sections of the confusion matrix (TP, FN, TN, and FP)

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FN)(TN + FN)}}.$$

F2-Score 3.3 is another evaluation metric which we considered in our experiments. *F2-Score*, is a special case of the *Fβ-Score* metric 3.3, where $\beta = 2$

$$F\beta\text{-}Score = \frac{(1 + \beta^2) \times Precision \times Recall}{(\beta^2) \times Precision + Recall}$$

With *F2-Score*, *Recall* becomes more significant and *Precision* becomes less important. Therefore, *F2-Score* is better suited in applications where correctly classifying as many positive samples as possible is more important than maximising the number of correct classifications. In other words, we want the blue circle in Fig. 6 to dominate the yellow circle. In real world settings, it may be preferable to have a very high TP rate, as an incorrect FN may have dire consequences.

3.4 Evolutionary Process

Fig. 7 shows our proposed grammar. GE determines the architecture of the model as well as its hyper-parameters. For instance, GE may choose whether to use *DropOut* or *DropBlock*, as well as what *rate* or *block-size* and *keep-prob* to use.

```
      <encoder> ::= <down> | <encoder>\n\t<down>
      <decoder> ::= <up> | <decoder>\n\t<up>
      <spatial> ::= <sp> | <conv>
        <layer> ::= <conv>\n\t<drop_type>\n\t<Batch>
         <down> ::= <layer>\n\t<pool>
           <up> ::= <transpose>\n\t<drop_type>\n\t<Batch>
           <sp> ::= x = spatial_attention(x)
   <transpose> ::= x = Conv2DTranspose(<filters>,(2, 2),<padding>)(x)
         <conv> ::= x = Conv2D(<filters>,<kernel>,<activation>, padding = <padding>)(x)
         <pool> ::= x = <pooling>((2, 2), padding='same' )(x)
        <Batch> ::= x = BatchNormalization()(x)
   <DropBlock> ::= x = DropBlock2D(<block_size>, <keep_prob>)(x)
     <Dropout> ::= x = Dropout(<rate>)(x)
   <drop_type> ::= <Dropout> | <DropBlock>
  <activation> ::= 'relu' | 'sigmoid' | ...
     <pooling> ::= MaxPooling2D | AveragePooling2D
        <rate> ::= 0.4 | 0.5 | 0.6
<optimisation> ::= 'Adam' | 'sgd' | 'adamax' | ...
     <filters> ::= 16 | 32 | 64 | 128 | 256
      <kernel> ::= (1, 1) | (2, 2) | ...
  <block_size> ::= 7 | 9
   <keep_prob> ::= 0.8 | 0.9
     <padding> ::= "valid" | "same"
```

Fig. 7. Proposed GE Grammar.

Table 1. List of the parameters used to run GE.

Parameter	**Value**
Fitness Function	*Val-Accuracy* and *F2-Score*
Runs	5, 5 and 3
Total Generations	15
Population Size	100
Crossover Rate	0.9
Mutation Rate	0.5
Epochs (Training)	15
Epochs (Best)	500

Table 1 shows our GE parameters. As we employ a non-semantic grammar, we had a large number of invalid individuals with mismatched output dimensions. Decoder layers were supposed to come up depending on how many layers our encoder had descended, however in some individuals, this didn't happen.

Therefore, each experiment had a population of 100, which is higher than in our prior studies, and it was run for 15 generations. One-point crossover and bit-flip mutation were utilized in each experiment, together with random initialization to create the initial population. During the evolutionary run, each individual was trained for 15 epochs with a batch size of 4. The top-performing model was trained for an additional 500 epochs.

We used two different fitness functions. *Validation-accuracy* was the fitness function in our first and second sets of experiments. We repeated first sets 5 times and then 5 more just for optimising hyper-parameters of SA-UNet. Validation-accuracy helps in preventing over-fitting, which is important for our small training dataset. Next, we used *F2-Score* as the fitness function to create a model which is optimised to maximise True-Positive rate. We performed 3 runs for the second fitness function.

4 Results

In Table 2 we show the hyper-parameters of our best models. *Exp1* and *Exp2* are the two best models found during our first set of experiments (5 runs). *Exp1* is a SA-UNet and *Exp2* is a U-Net. The results from our first set of experiments showed that GE chose a SA-UNet more frequently than a U-Net architecture.

Our second set of experiments used a SA-UNet as a base model for the evolutionary process. The hyper-parameters of the best model which GE discovered in this set of experiments is shown in *Exp3*.

The best model from the last set of our experiments, in which we used *F2-Score* as the fitness function, is seen in *Exp4*. Interestingly, this is a U-Net.

As can be seen, all of the models have the combination of *DropBlock* and *DropOut*, a feature rarely seen in human designed networks. We can see some similarity in U-Net and SA-UNet models. All the U-Net models have $Depth = 2$, $Kernel = (5, 5)$, $Padding = same$ and $Pooling = AveragePooling$. In SA-UNet models we have $Depth = 2$, $Kernel = (3, 3)$ and $Kernel = (5, 5)$, $Padding = same$ and $Padding = valid$, while both $AveragePooling$ and $MaxPooling$ were used in SA-UNet. $Filter\text{-}Size = 16$ is the most frequent one amongst all models. $'adam'$ and $'Nadam'$ are the most popular *optimisers*. We have a good combination among choosing an *Activation* function.

Table 3 shows the full results of our experimentation on test dataset, with the best results compared with the standard U-Net[3] and standard SA-UNet[4], as found online. *Exp1* obtained a higher AUC, (0.976), than *Exp2*, 0.963.

Exp3 achieved better AUC, 0.978, compared to the standard SA-UNet, 0.977. It's *size*, $3.8M$, is also much smaller than the standard SA-UNet, $17.1M$.

Indeed, GE evolved more accurate and much smaller models, the smallest of which, *Exp2*, was a comparatively tiny $0.9M$ when compared to the standard U-Net size, $42.4M$, and standard SA-UNet size, $17.1M$. This highlights the great benefits GE can provide.

[3] `https://github.com/zhixuhao/unet`
[4] `https://github.com/clguo/SA-UNet`

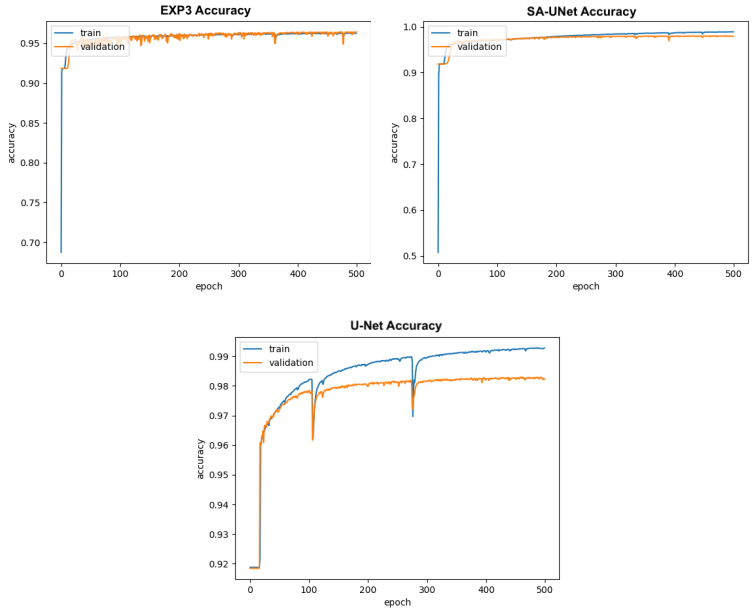


Fig. 8. Different models' training and validation plots for 500 epochs.

Exp4 achieved best *AUC*, 0.979, *Accuracy*, 0.966 and *Specificity*, 0.988 compared to the original models and those found during our evolutionary experiments.

It is also worth noting that our model has less over-fitting than the SA-UNet and U-Net, as demonstrated in Fig. 8.

We also demonstrate the arrangement of the hyper-parameters in two of our models in Fig. 9 and Fig. 10.

The results of our best models compare to U-Net and SA-UNet for two images are shown in Fig. 11. Our model discovered more details in both images.

Table 2. Hyper-parameters of the best models.

Name	**Exp1**	**Exp2**	**Exp3**	**Exp4**
Model	SA-UNet	U-Net	SA-UNet	U-Net
Depth	3	2	3	2
Filter	16-64-128	32-64-128	16-32-64-256	16-256
Kernel	3-5	5	3-5	5
Drop-type	Block-Out	Block-Out	Block-Out	Block-Out
Padding	same-valid	same	same-valid	same
Optimiser	adam	adam	Nadam	Nadam
Activation	selu-tanh-sigmoid	realu-tanh	relu-softsign-sigmoid	elu-sigmoid
Pooling	Avg-Max	Avg	Avg-Max	Avg

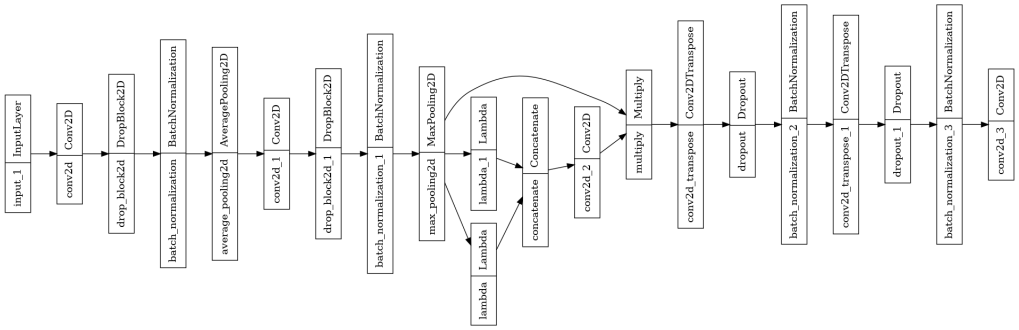


Fig. 9. Exp3 Architecture.

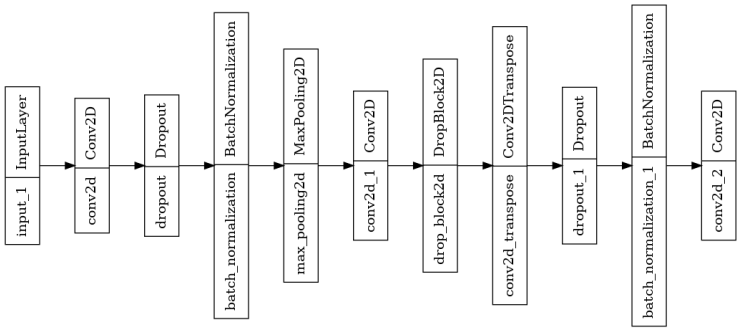


Fig. 10. Exp2 Architecture.

5 Conclusion

The architecture and hyper-parameters of a CNN used in image segmentation were optimized by GE. Previously, the U-Net and SA-UNet were employed as base models for a GA to optimize their hyper-parameters separately. In this study, we utilized GE to create and optimise diverse architectures, at the same time. We employed several drop types and attention blocks to decrease noise in order to solve over-fitting and enhance the performance of CNN models.

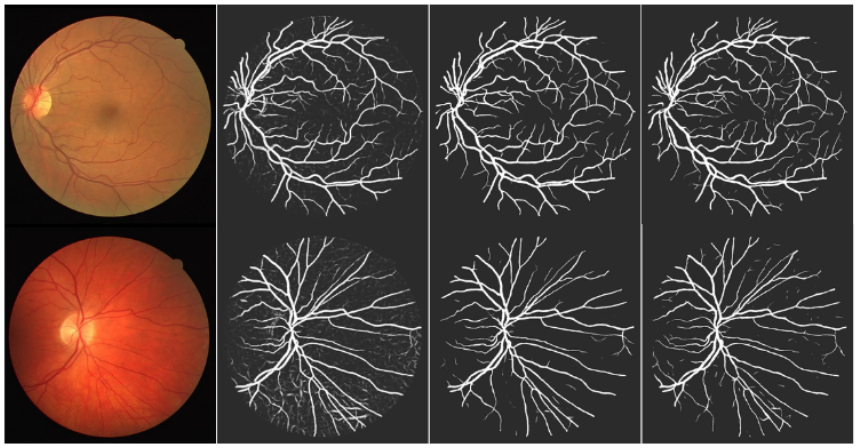
Our results demonstrated that we outperformed the original models in terms of *AUC* and *Accuracy*. Furthermore, our models are over ten times smaller than U-Net and SA-UNet. A smaller model has the advantage of requiring less training time and computation costs.

We plan to undertake further experimentation with semantic grammars in order to further increase the accuracy of our models; these will remove all morphologically invalid individuals. Further training for longer epochs may also lead

to improved performance. Including further modification that have shown to improve the performance of U-Nets is also an avenue for future research. Finally we wish to investigate the use of a multi objective fitness function when evolving U-Net architectures.

Table 3. Results of test set for both fitness functions. The bold numbers are the best results and underlined numbers are the best results which GE found.

Name	U-Net	SA-UNet	Exp1	Exp2	Exp3	Exp4
Accuracy	0.964	0.963	0.956	0.957	0.964	**0.966**
AUC	0.978	0.977	0.976	0.963	0.978	**0.979**
NPV	0.980	0.983	0.967	**0.993**	0.986	0.975
Specificity	0.980	0.977	0.984	0.961	0.974	**0.988**
Recall	0.791	0.760	**0.839**	0.583	0.734	0.737
Precision	0.799	0.812	0.707	**0.889**	0.836	0.853
F2-Score	**0.793**	0.770	0.768	0.627	0.752	0.757
MCC	**0.776**	0.766	0.747	0.700	0.764	0.774
Size	42.4 M	17.1 M	4 M	**0.9 M**	3.8 M	10.3 M



(a) (b) (c) (d)

Fig. 11. Comparison the results of 2 images. b is Exp4, c is U-Net and d is SA-UNet.

6 ACKNOWLEDGEMENTS

The Science Foundation Ireland (SFI) Centre for Research Training in Artificial Intelligence (CRT-AI), Grant No. 18/CRT/6223, and the Irish Software Engineering Research Centre (Lero), Grant No. 16/IA/4605, both provided funding for this study.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: a system for large-scale machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16). pp. 265–283 (2016)
2. Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a convolutional neural network. In: 2017 international conference on engineering and technology (ICET). pp. 1–6. Ieee (2017)
3. Assunçao, F., Lourenço, N., Machado, P., Ribeiro, B.: Automatic generation of neural networks with structured grammatical evolution. In: 2017 IEEE Congress on Evolutionary Computation (CEC). pp. 1557–1564. IEEE (2017)
4. Baldi, P., Sadowski, P.J.: Understanding dropout. Advances in neural information processing systems **26** (2013)
5. Ciulla, T.A., Amador, A.G., Zinman, B.: Diabetic retinopathy and diabetic macular edema: pathophysiology, screening, and novel therapies. Diabetes care **26**(9), 2653–2664 (2003)
6. Dietterich, T.: Overfitting and undercomputing in machine learning. ACM computing surveys (CSUR) **27**(3), 326–327 (1995)
7. Doersch, C.: Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908 (2016)
8. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks. Advances in neural information processing systems **31** (2018)
9. Gondara, L.: Medical image denoising using convolutional denoising autoencoders. In: 2016 IEEE 16th international conference on data mining workshops (ICDMW). pp. 241–246. IEEE (2016)
10. Guo, C., Szemenyei, M., Yi, Y., Wang, W., Chen, B., Fan, C.: SA-UNet: Spatial attention U-Net for retinal vessel segmentation. In: 2020 25th International Conference on Pattern Recognition (ICPR). pp. 1236–1242. IEEE (2021)
11. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975), second edition, 1992
12. Houreh, Y., Mahdinejad, M., Naredo, E., Dias, D.M., Ryan, C.: HNAS: Hyper Neural Architecture Search for image segmentation. In: ICAART (2). pp. 246–256 (2021)
13. LeCun, Y., Bengio, Y., et al.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks **3361**(10), 1995 (1995)
14. de Lima, A., Carvalho, S., Dias, D.M., Naredo, E., Sullivan, J.P., Ryan, C.: Grape: Grammatical algorithms in python for evolution. Signals **3**(3), 642–663 (2022)
15. Lima, R., Pozo, A., Mendiburu, A., Santana, R.: Automatic design of deep neural networks applied to image segmentation problems. In: European Conference on Genetic Programming (Part of EvoStar). pp. 98–113. Springer (2021)

16. Mahdinejad., M., Murphy., A., Healy., P., Ryan., C.: Parameterising the SA-UNet using a genetic algorithm. In: Proceedings of the 14th International Joint Conference on Computational Intelligence - ECTA,. pp. 97–104. INSTICC, SciTePress (2022)
17. Minaee, S., Boykov, Y.Y., Porikli, F., Plaza, A.J., Kehtarnavaz, N., Terzopoulos, D.: Image segmentation using deep learning: A survey. IEEE transactions on pattern analysis and machine intelligence (2021)
18. Nyathi, T., Pillay, N.: Comparison of a genetic algorithm to grammatical evolution for automated design of genetic programming classification algorithms. Expert Systems with Applications **104**, 213–234 (2018)
19. Ogurtsova, K., da Rocha Fernandes, J., Huang, Y., Linnenkamp, U., Guariguata, L., Cho, N.H., Cavan, D., Shaw, J., Makaroff, L.: Idf diabetes atlas: Global estimates for the prevalence of diabetes for 2015 and 2040. Diabetes research and clinical practice **128**, 40–50 (2017)
20. Popat, V., Mahdinejad, M., Cedeño, O.D., Naredo, E., Ryan, C.: GA-based U-Net architecture optimization applied to retina blood vessel segmentation. In: IJCCI. pp. 192–199 (2020)
21. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
22. Ryan, C., Collins, J.J., Neill, M.O.: Grammatical evolution: Evolving programs for an arbitrary language. In: European conference on genetic programming. pp. 83–96. Springer (1998)
23. Staal, J.: DRIVE: Digital retinal images for vessel extraction (2018), `http://www.isi.uu.nl/Research/Databases/DRIVE`
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
25. Wang, B., Qiu, S., He, H.: Dual encoding U-Net for retinal vessel segmentation. In: International conference on medical image computing and computer-assisted intervention. pp. 84–92. Springer (2019)
26. Wang, W., Huang, Y., Wang, Y., Wang, L.: Generalized autoencoder: A neural network framework for dimensionality reduction. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 490–497 (2014)
27. Winder, R.J., Morrow, P.J., McRitchie, I.N., Bailie, J., Hart, P.M.: Algorithms for digital image processing in diabetic retinopathy. Computerized medical imaging and graphics **33**(8), 608–622 (2009)
28. Woo, S., Park, J., Lee, J.Y., Kweon, I.S.: Cbam: Convolutional block attention module. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018)
29. Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J.: Unet++: A nested U-Net architecture for medical image segmentation. In: Deep learning in medical image analysis and multimodal learning for clinical decision support, pp. 3–11. Springer (2018)