IGARSS 2021 - 2021 IEEE International Geoscience and Remote Sensing Symposium | 978-1-6654-0369-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/IGARSS47720.2021.9554309

EVOLUTIONARY OPTIMIZATION OF NEURAL ARCHITECTURES IN REMOTE SENSING CLASSIFICATION PROBLEMS

Daniel Coquelin[1], *Rocco Sedona* [2,3], *Morris Riedel* [2,3], *Markus Götz*[1]

[1] Steinbuch Centre for Computing (SCC), Karlsruhe Institute of Technology (KIT), Germany
[2] Juelich Supercomputing Centre (JSC), Forschungszentrum Jülich (FZJ), Germany
[3] School of Engineering and Natural Sciences (SENS), University of Iceland (UoI), Iceland

ABSTRACT

BigEarthNet is one of the standard large remote sensing datasets. It has been shown previously that neural networks are effective tools to classify the image patches in this data. However, finding the optimum network hyperparameters and architecture to accurately classify the image patches in BigEarthNet remains a challenge. Searching for more accurate models manually is extremely time consuming and labour intensive. Hence, a systematic approach is advisable. One possibility is automated evolutionary Neural Architecture Search (NAS). With this NAS many of the commonly used network hyperparameters, such as loss functions, are eliminated and a more accurate network is determined.

***Index Terms*—** Neural Architecture Search, NAS, Evolutionary Algorithms, Remote Sensing, Classification

1. INTRODUCTION

Deep learning has reached unprecedented results across many domains in the last decade and Remote Sensing (RS) is no exception. As deep neural networks require a large amount of data to train, the constant stream of measurements from satellite-borne sensors makes RS an frequent use-case. In recent years, publicly accessible, labeled, RS-specific datasets [1, 2] have been released, allowing the RS community to train neural networks tailored to specific research questions.

ResNet-50 is a convolutional neural network (CNN) with skip connections which has been shown to be effective for computer vision tasks [3]. It has been tested extensively and is the bases for many CNNs used in a multitude of different fields including Remote Sensing.

There are multiple ways of evaluating the effectiveness of a neural network, many of which use the ideas of precision and recall. The F_1 score is one such measure. It is the harmonic mean of precision and recall and is often used in the multi-class multi-label case. The result can be calculated globally, i.e. the micro-F_1 score, or it can be the unweighted average of the F_1 scores for each class, i.e. the macro-F_1 score [4]. There are other methods, however they are not relevant to this work.

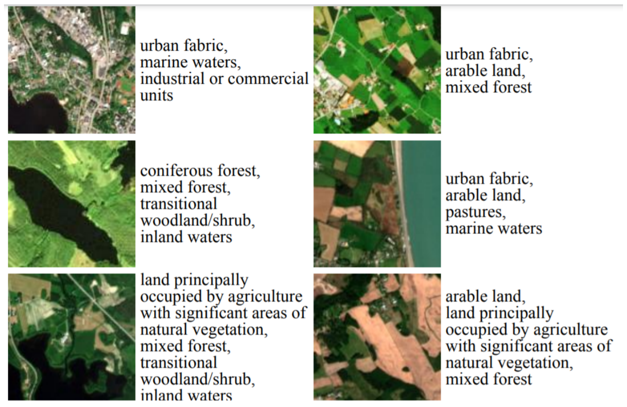

Fig. 1. Example patches and labels for Sentinel-2 tiles [6].

BigEarthNet [1], a large-scale remote-sensing dataset containing patches extracted from 125 Sentinel-2 tiles (Level2A) acquired from June 2017 to May 2018 [1]. The archive consists of 590,326 patches, each of which is assigned one or more of the 19 available labels. The label nomenclature is an adaptation of the CORINE Land Cover [5] which consists of labels from 10 European countries updated in 2018 [6]. Each patch has 12 spectral bands at various resolutions: 3 RGB bands and band 8 at $10\,\mathrm{m}$ resolution ($120\,\mathrm{px} \times 120\,\mathrm{px}$); bands 5, 6, 7, 8a, 11, and 12 at $20\,\mathrm{m}$ resolution ($60\,\mathrm{px} \times 60\,\mathrm{px}$); and bands 1 and 9 at $60\,\mathrm{m}$ resolution ($20\,\mathrm{px} \times 20\,\mathrm{px}$). The cirrus sensitive band 10 was omitted, as well as patches covered with snow or clouds [7]. Example patches are shown in fig. 1.

In [6], multiple network types were trained for the BigEarthNet dataset to various degrees of success. Those experiments excluded bands 1 and 9, utilized the Adam optimizer, the sigmoid cross entropy loss, an initial learning rate of 0.001, and were trained for 100 epochs. It was found that ResNet-50 was the most accurate network architecture, achieving a micro-F_1 score of 77.11 and a macro-F_1 score of 67.33.

[1]http://bigearth.net/

As a portion of the data was excluded for these experiments (bands 1 and 9), the inclusion of this data may increase the effectiveness of a network. This, combined with a different network architecture or training configuration has the possibility to greatly improve both training time and accuracy.

2. EVOLUTIONARY OPTIMIZATION

The structure of a neural network is determined by its hyperparameters. These include, the optimizer, the number of training epochs, the network architecture, the loss function, and many others. As the number of combinations of hyperparameters for a network can become extremely large, it is typically infeasible to determine effective combinations manually. Therefore, it is typically done automatically via selected a hyperparameter search and a Neural Architecture Search (NAS). A NAS typically takes a large amount of time to complete as it must train a network for each set of hyperparameters to determining the set's effectiveness or quality. The algorithms for finding the optimal hyperparameters within a NAS vary widely [8], however this work utilizes an evolutionary NAS to search for an optimal configuration for the ResNet-50 network used in [6].

The evolutionary NAS to be used begins with a population of random hyperparameter sets, known as offspring. After these networks are trained, they are sorted into the mating population. When an offspring is added to the mating population, the worst-performing network is removed. New generations of offspring are created by breeding two random members of the mating population, i.e. randomly sampling hyperparameters from each set. During the mating process mutations can occur, i.e. modifying a small amount of the offspring's set. These new networks are then trained and the process is repeated for either a specified number of generations or until an external factor stops it.

3. EXPERIMENTS

The hyperparameter search space was divided into six groups: optimizers, learning rate (LR) schedulers, activation functions, loss functions, the number of filters in each convolutional block, and the activation order. The search space is shown in table 1, excluding the number of filters which exponentially spans the space between two and 256. For this network, the number of filters in a convolutional block is dependent on a ratio with the first block's number of filters. The parameters of the learning rate schedulers and optimizers were also included in the search space.

It has been shown in [13] that the order of activation, batch normalization (BN), and convolution layers within residual building blocks can affect the accuracy of a network. The test configurations shown in fig. 2 will be used when referring to the activation order.

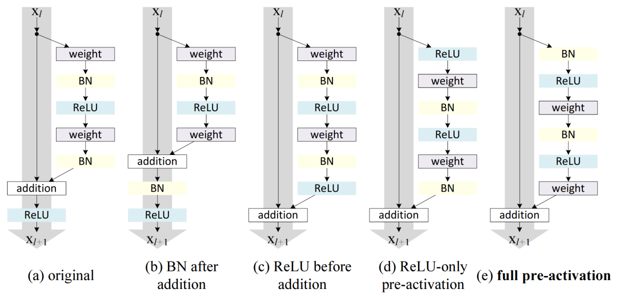

Fig. 2. The various orders of the activation, batch normalization (BN) and convolution layers within residual building blocks used in a network. ReLU is an example activation function, during the NAS the activation function is defined by the network hyperparameters [13].

The data is prepared analogously to [6]. Albeit, there is no image augmentation performed on the images. The network was implemented in TensorFlow [17] and the NAS was controlled by the open source package `propulate` [18]. `propulate` maintains the status of the population via MPI [19]. The experiments were performed on various numbers of NVidia A100 GPUs [20] on ForHLR 2 at KIT. The networks use early stopping to exit the training if the loss as measured on the validation set is not longer increasing for ten epochs.

Hyperparameter combinations which contained the optimizers Adam, Adamax, Nadam, and RMSprop, failed more frequently than their counterparts. These optimizers will return NaN values if the combination between the loss function, optimizer, and their parameters is unfavorable, most likely due to their adaptive algorithms. This behavior is commonly referred to as instability. It is important to note that an optimizer's stability is typically a poor indicator of its effectiveness. Since other optimizers were more stable, the more unstable optimizers are quickly excluded from the search space. To compensate for this effect, individual NASs were performed for Adam, Adamax, Nadam, and RMSprop.

As poorly performing networks are removed from the population, the relative frequency of hyperparameter selection can be used as a proxy for the stability and accuracy of the network with that hyperparameter. This measurement is conducted across all of the completed searches as to avoid excluding the metrics which perform better with the less stable optimizers. By this measure, the most successful loss functions were binary cross entropy and categorical hinge. The least effective loss functions were hinge, Kullback–Leibler divergence, and squared hinge. Exponential decay, inverse time decay, and polynomial decay were used at roughly the same frequency. Indicating that, when given fitting parameters, all three are stable and effective choices for learning rate schedulers. The use of this metric to determine the effectiveness of activation functions is flawed as activation function

Table 1. Neural architecture and hyperparameter search space. The Activation column header shows the activation functions. Activation order details are shown in fig. 2. ELU is the exponential linear unit, ReLU is the rectified linear unit, SELU is the scaled exponential linear unit, K-L divergence is the Kullback-Leibler divergence, and `tanh` is the hyperbolic tangent.

Optimizer [9, 10]	**LR Scheduler** [11]	**Activation** [12]	**Activation Order** [13]	**Loss** [14, 15, 16]
Adadelta	Exponential Decay	ELU	Original	Binary Cross Entropy
Adagrad	Inverse Time Decay	Exponential	BN after addition	Categorical Cross Entropy
Adam	Polynomial Decay	Hard sigmoid	Activation before addition	Categorical Hinge
AMSGrad		Linear	Activation-only pre-activation	Hinge
Adamax		ReLU	full pre-activation	K-L Divergence
Ftrl		SELU		Squared Hinge
Nadam		Sigmoid		
RMSprop		Softmax		
SGD		Softplus		
		Softsign		
		Swish		
		tanh		

effectiveness was found to be correlated with the network optimizer. I.e. Adamax was most effective with ELU whereas Adadelta was most effective with Softmax. Nonetheless, the most commonly chosen activation function was a linear function. It is theorized that this is due to its stability with many optimizers. Therefore it is chosen in the early stages of the NAS and slowly disregarded as other, more effective, activation functions are found. Similarly to the usage of the linear activation function, the number of filters chosen during the early stages of the NAS is primarily four. As the NAS progresses, the number of filters is predominantly chosen as either 32 or 64. The most effective activation order is full pre-activation (as shown in fig. 2).

As mentioned, some optimizers were sequestered into separate searches and thus the measure of their frequency is less useful. Therefore, they are judged on the F_1 scores which their optimizers produce. Ftrl did not fail more frequently than others, however it did not continue to produce competitive results and was slowly eliminated from the population in the general NAS. The individual searches with Adamax, RMSprop, and Adam all produced competitive networks with the collective NAS. Showing that the networks excluded for stability issues were unfairly removed for the population.

The most effective optimizers were Adadelta and SGD, both of which produced multiple configurations which had results greater than or equal to the target micro-F_1 score of 77.11. The most accurate configuration converged to a mirco-F_1 score of 77.25 and a macro-F_1 score of 69.57 in 10 epochs. Both of which are accuracy improvements over the networks presented in [6], and the found network converged in 90 fewer epochs. The final network configuration is as follows. The optimizer was `Adadelta`, with rho as 0.8831, and epsilon as $3.4771e - 06$; the learning rate scheduler was polynomial decay with an initial learning rate of 0.8666, 10585 decay steps, an ending learning rate of 0.0035, a power of 0.8693, and without cycling; 128 filters; the softmax activation function; the binary cross entropy loss; and with full pre-activation.

At a class level, the found network noticeably outperforms the networks used in [6] for certain classes. Namely, coastal wetlands; moors, heathland and sclerophyllous vegetation; and agro-forestry areas were 9.60%, 5.05%, and 6.86% more accurate then the presented ResNet50 implementation. However, the class "beaches, dunes, and sands" performed 17.4% worse. As the overall performance is very similar, the large differences in class accuracies are attributed to the inclusion of more spectral bands in the trained data.

4. CONCLUSION

The recent increase in availability of large RS datasets has allowed researchers to achieve unprecedented results in tasks such as land cover classification. However, this comes at the price of spending a considerable amount of time manually searching for optimal DL architectures and hyperparameters.

When utilized correctly, a NAS can determine network hyperparameters which result in accelerated convergence rates, improved accuracy, or both. However, if the search space of a NAS contains hyperparameters of functions with varying stability, some hyperparameters can be unfairly excluded. To avoid this, either careful selection of the search space hyperparameters or multiple NASs are required. This is exemplified by the various optimizers which required individual NASs to produce competitive results.

It has been shown that an evolutionary NAS can increase the accuracy and greatly increase the convergence rate of the target network. The use of Adadelta, polynomial decay, Softmax, and binary cross entropy has been shown to slightly outperform the initial network.

Some of the classes showed greatly improved accuracy with inclusion of more spectral bands. However, one category (beaches, dunes, and sands) was significantly less accurate than previous results. Therefore, the inclusions of more bands should be done based on the intended use-case of the network after training. More research is required to determine the effectiveness of each band for the accurate classification of an image.

This result, and the result of any NAS, is limited by its search space and as the functions within the search space were selected based on their commonality, there are many cutting edge functions which were not included. To greatly outperform the given network, a combination of implementing a more advanced network architecture and the selection of more useful loss functions is required.

Acknowledgments

This work was performed on the supercomputer ForHLR funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research. This work is supported by the Helmholtz Association's Initiative and Networking Funds under under the Helmholtz AI platform grant and the Helmholtz Analytics Framework collaboration.

5. REFERENCES

[1] Gencer Sumbul, Marcela Charfuelan, Begüm Demir, and Volker Markl, "BigEarthNet: A Large-Scale Benchmark Archive For Remote Sensing Image Understanding," *arXiv preprint 1902.06148*, 2019.

[2] Xiao Xiang Zhu, Jingliang Hu, Chunping Qiu, Yilei Shi, Jian Kang, et al., "So2Sat LCZ42: A Benchmark Data Set for the Classification of Global Local Climate Zones [Software and Data Sets]," *IEEE Geoscience and Remote Sensing Magazine*, vol. 8, no. 3, pp. 76–89, 2020.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 770–778, IEEE.

[4] Zachary Chase Lipton, Charles Elkan, and Balakrishnan Narayanaswamy, "Thresholding Classifiers to Maximize F1 Score," 2014.

[5] M. Bossard, J. Feranec, and J. Otahel, "CORINE land cover technical guide – Addendum 2000," Tech. Rep., European Environment Agency, 2000.

[6] Gencer Sumbul, Jian Kang, Tristan Kreuziger, Filipe Marcelino, Hugo Costa, et al., "BigEarthNet Dataset with A New Class-Nomenclature for Remote Sensing Image Understanding," 2020.

[7] "Scripts to Remove Cloudy and Snowy Patches," .

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter, "Neural Architecture Search: A Survey," 2019.

[9] Shiliang Sun, Zehui Cao, Han Zhu, and Jing Zhao, "A Survey of Optimization Methods from a Machine Learning Perspective," *arXiv preprint 1906.06821*, 2019.

[10] H. Brendan McMahan, "Analysis Techniques for Adaptive Online Learning," *arXiv preprint 1403.3465*, 2014.

[11] Yanzhao Wu, Ling Liu, Juhyun Bae, et al., "Demystifying Learning Rate Polices for High Accuracy Training of Deep Neural Networks," *arXiv preprint 1908.06477*, 2019.

[12] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning," *arXiv preprint 1811.03378*, 2018.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Identity Mappings in Deep Residual Networks," *arXiv preprint 1603.05027*, 2016.

[14] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 03 1951.

[15] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian, "A Comprehensive Survey of Loss Functions in Machine Learning," *Annals of Data Science*, Apr 2020.

[16] Shruti Jadon, "A survey of loss functions for semantic segmentation," *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, Oct 2020.

[17] Martín Abadi, Ashish Agarwal, Paul Barham, et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015, Software available from tensorflow.org.

[18] Oskar Taubert, "Propulate," `https://github.com/oskar-taubert/propulate`, 2020.

[19] Message Passing Forum, "MPI: A Message-Passing Interface Standard," Tech. Rep., University of Tennessee, USA, 1994.

[20] Yuhsiang Mike Tsai, Terry Cojean, and Hartwig Anzt, "Evaluating the Performance of NVIDIA's A100 Ampere GPU for Sparse Linear Algebra Computations," 2020.