Designing a New Search Space for Multivariate Time-Series Neural Architecture Search

Christopher MacKinnon[1] and Robert Atkinson[1]

University of Strathclyde, Glasgow, Scotland
christopher.mackinnon@strath.ac.uk
robert.atkinson@strath.ac.uk

Abstract. With the rise of edge computing and the Internet of Things (IoT), there is an increasing demand for models with low memory footprints. These models must be adaptable to embedded system applications, while being able to leverage the large quantities of data recorded in these systems to produce superior performance.
Automatic Neural Architecture Search (NAS) has been an active and successful area of research for a number of years. However, a significant proportion of effort has been aimed at finding architectures which are able to effectively extract and transform the information in image data. This has lead to search space design which is heavily influenced by the heuristics of image classifiers.
We review and incorporate the characteristics of successful time-series methods, while seeking to address traits of conventional NAS search-space design which may be detrimental to performance on time-series.
This paper provides an in-depth look at the effects of each of our design choices with an analysis of time-series network design spaces on two benchmark tasks: Human Activity Recognition (HAR) using the UniMib-SHAR dataset and Electroencephalography (EEG) data from the BCI Competition IV 2a dataset.
Guided by these design principles and the results of our experimental procedure, we produce a search space tailored specifically to time-series tasks. This achieves excellent performance while producing architectures with significantly fewer parameters than other deep learning approaches. We provide results on a collection of datasets from the UEA Multivariate time-series Classification Archive and achieve comparable performance to both deep learning and state-of-the-art machine learning time-series classification methods, using a simple random search.

1 Introduction

Neural Architecture Search (NAS) is a method of automatic architecture discovery and has been an active area of research for a number of years. Through this process, a large space of possible models is traversed and evaluated in an attempt to discover the optimal network architecture for a given domain task. While automated architecture design is well studied with regard to image data, its application to time-series problems has only recently begun to be investigated.

NAS search spaces have evolved over time to contain a high density of strong models. This has been achieved by adopting the principles that guide manual architecture design and applying them to search space design. These choices are easily seen when looking at cell-based search spaces such as NasNet [1] and DARTS (Differentiable Architecture Search) [2], where repeating cells are stacked often with residual connections between cells. When applying these methods to a new domain, however, the heuristics which guided search space design in one application may not be useful in another.

NAS search spaces can be considered restrictive in the diversity of models contained within them. While these spaces are "large" (DARTS for example is of order 10^{18}), many design choices are made to remove poor models from the space, reducing its overall diversity. While this likely improves the convergence speed and anytime performance of an architecture search, it limits the discovery of truly novel architectures which do not conform to traditional architecture design rules [3].

While deep learning approaches have begun to show promising results, particularly in multivariate time-series classification, the fidelity of these approaches is still dependent on the quality of manually designed architectures. In the domain of image classification NAS has achieved great success discovering architectures which outperform human designed architectures. This is the case despite the fact that top-performing architectures - and even parameterisations - often exhibit greater transfer-ability across tasks as evidenced by the relative ease of transfer learning compared with time-series classification [4]. The wide variety of problem characteristics in time-series tasks — such as dataset size, signal length, or discriminatory features — makes designing an optimal 'one-size-fits-all' architecture a challenging proposition. This highlights the opportunities for automatic architecture design methods which can discover architectures tuned to the specific characteristics of a dataset.

This paper introduces a novel search space for time-series NAS, which achieves competitive results compared with state-of-the-art (SOTA) time-series classification methods across a diverse set of multivariate time-series classification challenges, with only the most rudimentary search algorithm. We draw on concepts from successful time-series classification methods, as well as incorporating the characteristics of modern convolutional vision networks, integrating them into the design of a deep learning search space which produces highly efficient architectures with strong performance for time-series classification.

2 Related Work

2.1 Time-Series Classification

Time-series classification tasks are dominated by models which can generate a multitude of representations. A representation being a transform or encoding of the raw time-series which may reveal useful patterns or information. Unlike image classification models which often use deep repeating structures to extract

complex features, successful time-series models - whether they are deep learning or more traditional machine learning methods - are frequently characterised by an emphasis on the 'breadth' of representations. The Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) [5] is a good example of this, achieving SOTA performance by ensembling the predictions of a broad range of classifiers. Rocket [6] and Canonical Interval Forest (CIF) [7] are other examples of a focus on a diverse set of representations. These feature based methods, apply a large collection of randomly generated transforms to produce features. These features can then be used by a relatively simple classifier to achieve impressive results.

2.2 Deep Learning for Time-Series

The application of deep learning techniques to time-series classification has gained increasing attention in recent years. One of the earliest works in this area is the study by *Wang et al* [8] who proposed a deep learning framework for time-series classification and compared convolutional neural networks (CNN) with Multi-Layer Perceptrons (MLP), specifically looking at a ResNet architecture and Fully Convolutional Networks (FCN). The authors demonstrated the effectiveness of their proposed method on the set of UCR uni-variate time-series problems.

In the study by Fawaz, Lucas, Forestier, *et al.* [9], the authors proposed the use of InceptionTime, a modification of the Inception architecture designed specifically for time-series classification. This achieved strong performance in multivariate time-series classification tasks. The novel feature of InceptionTime is its use of parallel 1D convolutional filters. The kernels of these convolutions vary in length from 10 to 40, allowing the model to capture and extract patterns at various time scales in the signal.

The success of this approach shows again the importance of a broad set of representations but also the value in extracting information at different time scales for successful performance across diverse problems. We aim to leverage these insights in the application of NAS to discover effective architectures tailored to specific problems.

2.3 Neural Architecture Search

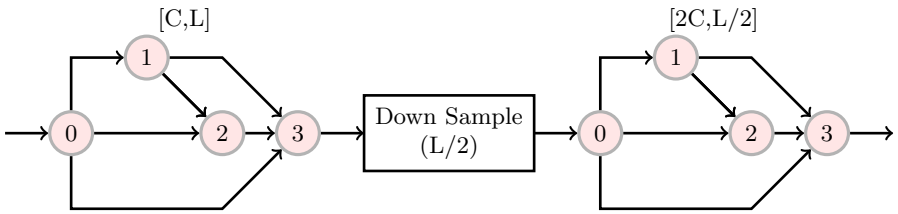
NAS can be considered as a subset of the general problem of hyperparameter optimisation. In a general sense, it frames any machine learning task as a bi-level optimisation problem, wherein both a set of parameters weights and hyperparameters settings are optimised w.r.t to the training and validation losses respectively. This is given by Equations 1 and 2, where θ is the parameters of a model and λ is the architecture configuration, with θ^* and λ^* being the optimal value of each respectively.

$$\lambda^* = \arg\min_{\lambda} \mathcal{L}_{validation}(\theta^*(\lambda), \lambda) \tag{1}$$

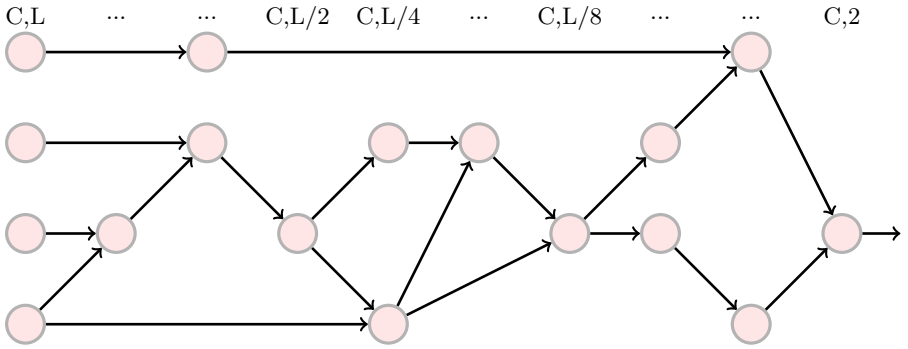
$$\text{s.t. } \theta^*(\lambda) = \arg\min_{\theta} \mathcal{L}_{train}(\theta, \lambda) \tag{2}$$

Initial approaches to architecture search within deep learning typically optimised the number of layers and the operation on each layer using a fixed model topology. This formulation is easily mapped to general hyperparameter optimisation methods with the number of layers, channels or kernels sizes being individual hyperparameters.

Although some early work in neuro-evolution searched for simple models in topological spaces such as NEAT [10], a significant innovation in the development of NAS was to search for models in a topological space. Many of these approaches used 'factorised' spaces such as hierarchical spaces or more famously cell-based spaces and achieved significant success [11][1]. In this type of approach architectures are constructed from repeating blocks which follows the characteristics of successful human designed architectures. The overall aim of this is to remove a large quantity of poorly performing architectures from the space while retaining the majority of strong networks, which conform to these heuristics.



(a) Cell Macro Structure: Down-sampling happens at fixed intervals in-between cells



(b) Our Graph Space Macro Structure: Dynamic down-sampling through-out model.

Fig. 1: Comparison of down-sampling and network structure in a cell-based search space and in our graph space.

3 Designing a new Search Space

In this section, we introduce a novel search space designed specifically for the characteristics of time-series data, which aims to effectively handle a wide range of signal lengths and produce a diverse set of transforms akin to what we see in other successful time-series methods. Figure 1 shows the traditional cell-based space compared with our approach.

In conventional cell-based network structure, down-sampling of the input signal is coupled to the network depth occurring at fixed intervals between cells. In contrast, our approach incorporates this down-sampling into the architecture search. The advantages of this are twofold: firstly, it allows for the extraction of features at multiple signal resolutions in parallel, where computation can be performed at the most effective granularity for different discriminatory features. Secondly, it avoids the need for very deep networks when dealing with long signal lengths or manual tweaks to the down-sampling operations for specific tasks.

We define a model topology where configurations are described in terms of a Directed Acyclic Graph (DAG) that defines the edges and connectivity of an architecture, as well as a set of operations with one corresponding to each of the defined edges. A valid architecture can be constructed from any DAG where the start and end nodes are connected via all paths. In order to generate a wide range of topologies, which conform to this specification, we propose an iterative method. Starting with a simple DAG containing 3 vertices (S,1,T) each connected by a single edge, we randomly apply one of two operations, "edge insertion" or "edge split". Alg. 1 describes a single iteration of this process with Fig. 2 showing this process over multiple iterations.

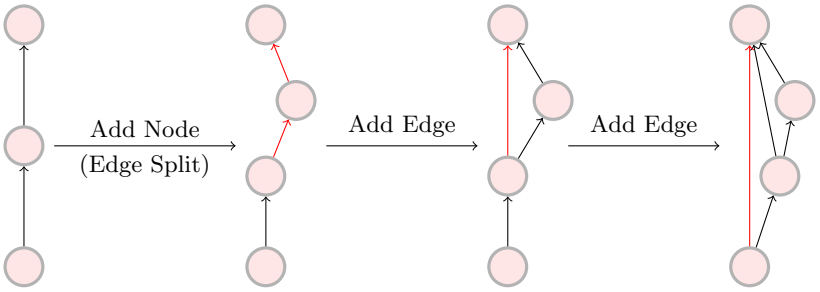


Fig. 2: Three iterations of the graph generation algorithm adding a new node in the first step, and a new edge in the subsequent iterations. Changes are highlighted in red.

3.1 Operation Chain

To further enhance the range of architectures that can be expressed in the search space, aspects that are often part of the primitive operations or simply built into

a fixed macro architecture are brought into the search space. This work innovates on the standard DAG representations of neural architectures, with the inclusion of node attributes.

Previous implementations like DARTS or NasNet primarily focused on edge attributes, i.e., the primitive operations. Our approach expands on this by associating attributes such as the normalisation, activation function, channel width (number of channels) and down-sampling (stride) to the node rather than an edge. An example of this is an edge (1,3) and (2,3), which will have the same node attributes, due to their common destination at node 3.

Fig. 3 provides a visualisation of this operation chain and the assignment of attributes to edges or nodes.

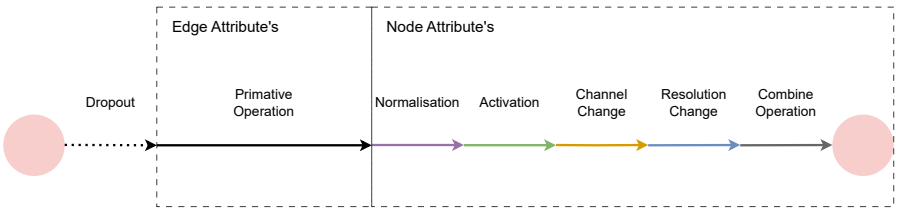


Fig. 3: Chain of operations in a compiled edge

Adjustments in the number of channels or resolution are implemented through the use of point-wise and depth-wise convolutions, respectively. Where the latter has a kernel size equal to its stride. These changes in resolution and channel width are propagated downstream to all subsequent edges and nodes. Figure 4 shows an example, if node A contains a down-sample of signal length L by a factor of two, and there exists an edge (A, B), then the operation of this edge will act on and produce data of size $(C, L/2)$ assuming no changes to the channel width. This means that B will be of size $(C, L/2)$. If another node C connects to B and has a signal length of L, then a down-sample operation will be added here also to produce two signals of the same length. This feature increases the flexibility of the generated architecture, allowing it to adapt to signals of varying lengths.

4 Benchmarking Tasks

In order to draw broadly useful conclusion about NAS search spaces for time-series problems we look at two time-series classification tasks from disparate domains.

4.1 Human Activity Recognition (HAR)

Human Activity Recognition is the task of classifying actions such as walking, running, jumping, as well as falls through the use of gyroscopic data. Among

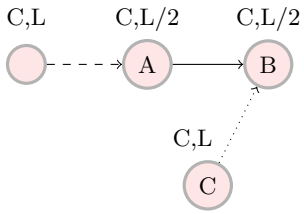


Fig. 4: Example of down-sampling operations and the propagation of signal length to subsequent nodes. Dashed lines indicate where a node contains a down-sample operation, dotted line indicates when a down-sample operation is induced.

members of the population who are over 65, falls are a common risk and can have severe consequences with the frequency only becoming larger with age. Undetected these falls can result in hospitalisation or, in the worst case, be fatal.

The *UniMib-SHAR* dataset is a benchmark for human activity recognition (HAR) collected by the University of Milano-Bicocca [12]. The dataset contains data from 30 subjects, both male and female aged 18 - 60, performing activities of daily living (ADL), such as standing, walking and sitting, as well as a collection of falls, while wearing a smartphone on their waist. The smartphone's accelerometer and gyroscope sensors capture three-axial linear acceleration and three-axial angular velocity, respectively. This dataset can be used for 4 classification tasks:

- (AF-17) - Distinguishing all 17 fine-grained classes from the ADL and FALL categories
- (AF-2) - Binary classification of ADLs and FALLs
- (A-9) - Classifying the 9 ADL fine-grained classes
- (F-8) - Classifying the 8 FALL fine-grained classes

In this paper we look specifically at the AF-17 scheme which is considered the most challenging task, with data being split based on subjects unless specifically stated otherwise.

4.2 Electroencephalography (EEG)

The *BCI Competition IV 2a* dataset is a widely used electroencephalography (EEG) dataset, collected as part of the Brain-Computer Interface (BCI) Competition IV. It comprises EEG recordings from 9 subjects, each performing 4 different motor imagery tasks: left hand, right hand, both feet and tongue movements. The dataset contains 22 EEG channels and 3 electrooculography (EOG) channels, sampled at 250 Hz, with each trial lasting approximately 8 seconds. In this paper we use only the 22 EEG channels.

Algorithm 1 Generate Graph Iteration

```
procedure GEN_ITER(edges, rate)
    g ← DiGraph(edges)
    if random() > rate then                                  ▷ Add an edge
        sorted_nodes ← TopologicalSort(g)
        number_valid ← len(sorted_nodes) − 1
        source_index ← RandInt(0, number_valid)
        new_source ← sorted_nodes[source_index]
        existing ← Neighbors(g, new_source)
        valid ← sorted_nodes[source_index + 1 :]
        valid ← valid.remove([existing, "S"])
        new_end ← RandomChoice(valid)
        edges.append((new_source, new_end))
    else                                          ▷ Add a node (Edge split)
        edge ← random.choice(edges)
        new_id ← len(edges) + 1
        idx ← edges.index(edge)
        edges[idx] ← (edge[0], new_id)
        edges.append((new_id, edge[1]))
    end if
    return list(set(edges))
end procedure
```

Table 1: Summary of the dataset properties

Dataset	Samples	Dimensions	Timesteps
EEG Train Data	2328	22	1750
EEG Test Data	2368	22	1750
SHAR Train Data	10541	3	151
SHAR Test Data	1230	3	151

5 Method

In order to make comparisons of different search spaces we adopt a method utilised in the analysis of design spaces for image classifiers [13] [14]. We perform a random sampling of architectures (Random Search) in each search space on a set of tasks, comparing the Cumulative Distribution Function (CDF) — which gives the probability that a random observation will be less than or equal to a certain value. This gives us a more robust comparison between two spaces than a single point estimate, such as the single best performing model. By performing a series of experiments following this methodology, the goal is to find a search space of strong performing architectures for time-series classification by iteratively improving the search space at each step.

This is conducted as a random search evaluating 500 models over our two datasets. Each model has a fixed stem size of 32 channels, with a Global Average Pooling (GAP) and fully connected linear layer as the output. We define both spaces, the cell-based and graph search-spaces to have 32 edges for which

operations are searched. Each model is trained with a batch size of 256 for 200 epochs, with the learning rate decaying from 0.01 to 1×10^{-5} throughout the training process, based on the cosine annealing strategy.

5.1 Experiment 1: Understanding the Role of Width Multipliers During Down Sampling in Graph and Cell-Based Topologies

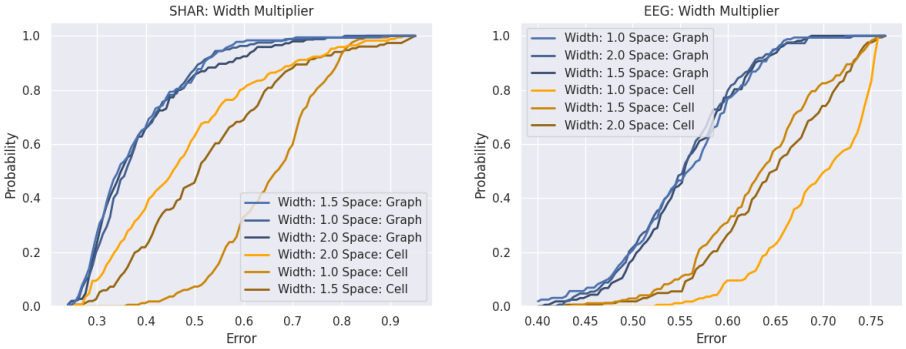


Fig. 5: Effects of different width multipliers on graph and cell based topologies across search spaces

As we have seen in Section 2.1, in comparison with Image tasks, time-series models generally make use of shallower networks favoring a wide variety of representations. However, time-series tasks can also have a large variety of signal lengths, adding an additional challenge to architecture design. Cell-based NAS architectures down samples the resolution or signal length at the end of specific 'reduction cells', increasing the number of channels to maintain the capacity. This approach however can lead to challenges in the context of time-series data, due to the coupling of signal down-sampling and network depth. We compare the effects on a cell based search space, based on implementation of [2] with our graph based search space. We use 4 cells, with down-sampling occurring after each cell, maintaining the same number of operations (32) in both search spaces.

Figure 5 shows the CDF of architecture error of the DARTS search space compared with our graph based search space over a range of down-sample width multipliers. We can see that in the case of the cell search space the optimal width multiplier seems to vary according to dataset, with a constant width performing consistently poorly. In contrast, this setting seems to have little effect on the graph search space.

The effect of these settings on the number of model parameters is also significant with a difference of around 1 order of magnitude between models of different width multipliers.

Moving forward we use the graph based architecture with a wide multiplier of 1 as the basis for further experiments.

5.2 Experiment 2: Breaking out of the Separable Convolution (Depth-wise and Point-wise Operations)

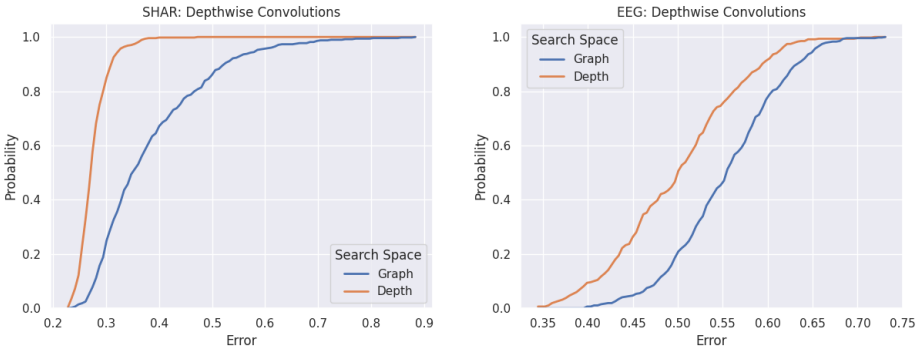


Fig. 6: Effects of breaking out the separable convolution operation across datasets

While depth-wise and point-wise convolutions are commonly used in NAS as constituent parts of the separable convolution to build more efficient architectures, we propose breaking the fundamental convolution operation down to the depth-wise and point-wise convolution. These operations are significantly less computationally demanding as each operation now contains a single convolution operation rather than the 4 convolutions in a separable convolution. In ViT architectures, we also see a separation of spatial and channel-wise information mixing, which may be a beneficial trait. Specifically in the context of multivariate time-series data, the input features may contain information which is not temporally aligned across channels, which could make channel mixing to be counter productive at certain stages in the model.

In order to have this approach be effective we also break out the activation and normalisation functions from the primitive operations. Taking inspiration for the ConvNeXt [15] results, we include 2 'none' operations for each activation and normalisation function, to produce networks with roughly 1 activation and normalisation in every 3 primitive operations.

We can adjust for the reduction in model capacity by increasing the number of edges in our architecture, allowing for a more diverse set of architectures at the same or lower computational cost. We also maintain the same distribution of operations for random sampling by introducing each of the depth-wise and point-wise convolutions into the operation pool four times for each kernel. The dropout rate was also reduced by a factor of 4 to maintain the same total dropout across each network.

Figure 6 shows the effects of these changes on the two datasets. Here we can see improvements across the board.

5.3 Experiment 3: Kernel Size and Dilation

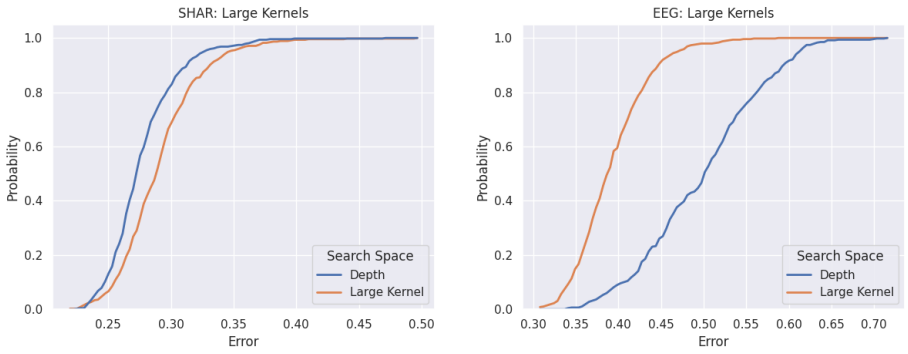


Fig. 7: Effects of using large kernel sizes on model performance across datasets

The use of dilation in convolutional networks for time-series is well established as an effective method of increasing the receptive field without exploding the number of parameters. This has shown success in a broad range of tasks and model types pertaining to time-series, with WaveNet [16] showing its application to deep learning and more recently ROCKET showing the effectiveness of dilated convolution kernels as feature extractors for more traditional machine learning approaches. We introduce kernels with a set of exponential dilation's (2,4,8,16,32) to the search space.

To further expand the search space we also add larger kernel sizes which has shown success in both image and time-series applications. We again add kernels of exponential sizes (reduced by one for padding) at 15,31,63. We also add pooling at these kernel sizes. Again we maintain the proportion of pooling, skip and convolution operations when adding the new operations to the pool of primitives.

Figure 7 shows the results of these large kernels on our datasets. Unlike with the previous change the results here are more mixed with a significant improvement on one dataset and a slight deterioration in another. These results are understandable due to the significantly shorter signal length of the SHAR dataset. This means a significant number of the kernels are larger than the entire signal even before down-sampling.

6 Results

We now show the performance of our new search space across a number of datasets. We randomly partition the dataset into a train, validation and test dataset. We use the simple random search algorithm [17], run for 500 iterations to select the architecture for each task. Each architecture is evaluated twice to improve quality of the evaluation, with the hyperparameters described in Section 5. The results of the final architecture on a holdout test set is reported as the mean accuracy of 10 training runs. Each search was run across 4 GPUs with all the architectures trained from scratch. Each search took between 3 and 24 hours to complete, depending on the characteristics and size of the dataset. Note that the search algorithm itself is intentionally rudimentary without any proxy evaluation methods to produce baseline performance estimates.

6.1 UniMib-SHAR

Table 2 presents the results obtained from two different schema: a 'subject-based split' and a 'random split'. In the 'subject-based split', no subject appears in more than one of the train, validation, or test sets. The random split is a standard random division of the data. The search was run for each approach separately.

The results compare the state of the art methods on this task with the discovered architecture. An improvement in both categories can be seen in terms of accuracy. A new SOTA is achieved with 95.7% accuracy in the random split, improving the accuracy by over 3% on the previous best, as well as a more modest 77.6% accuracy in the subject-based split. This was achieved while also producing models with a significantly lower number of parameters than reported by other approaches.

Table 2: UniMib-SHAR: Discovered architecture vs SOTA

Method	Subject-Based Split	Random Split	Parameters
Gao, Zhang, Teng, *et al.* 2021[18]	-	79.03	2.40M
Mukherjee, Mondal, Singh, *et al.* 2020[19]	-	92.60	-
Al-qaness, Dahou, Elaziz, *et al.* 2023[20]	77.29	84.99	2.40M
Helmi, Al-qaness, Dahou, *et al.* 2023[21]	-	86.08	-
Teng, Wang, Zhang, *et al.* 2020[22]	-	78.07	0.55M
New Search Space	**77.63**	**95.70**	0.10M

6.2 EEG

Next, we look at the EEG dataset; here we look at a random split, looking for general accuracy over the entire dataset. In the literature, it is more common to use subject-specific models as they tend to provide improved performance on

the particular subject, however, this involves the training of many different specialised models. We compare our approach with ResNet [8] and InceptionTime [9], both providing a strong baseline performance for deep learning models on time-series classification. These are implemented with 32 channels and an additional 64 channel version for InceptionTime. Table 3 shows the accuracy of our approach compared with the benchmark models. We see a significant disparity in performance here, with ResNet in particular struggling to find a good solution.

Table 3: EEG: Discovered architecture vs Deep Learning Approaches

Method	Random Split	Parameters
ResNet	35.89	0.95M
InceptionTime (32)	49.08	0.48M
InceptionTime (64)	50.22	1.89M
New Search Space	66.98	0.12M

6.3 UEA Multivariate time-series Classification Archive

We compare our search space to the SOTA time-series classification methods on a subset of the time-series classification archive. Due to the additional complications involved with performing NAS with small dataset sizes - which is out of scope for this paper - we look specifically at the 4 largest datasets in terms of number of samples. These are specifically the four largest multivariate datasets with equal length in the time-series classification archive to allow for comparison with the baseline experiments [23]. We conduct a search on each dataset separately. Table 4 shows the results of our new search space compared with SOTA time-series methods evaluated in Middlehurst, Large, Flynn, *et al.* [5] as well as deep learning approaches from Ruiz, Flynn, Large, *et al.* [23]. Our search space produces models which perform well compared with current state of the art, losing in average rank to only InceptionTime.

Table 4: UEA (Resamples) - Multivariate time-series Classification Archive: SOTA methods compared random sampling of new search space

Method	FaceDetection	LSST	PenDigits	PhonemeSpectra	Average Rank
HC2	71.35	**63.70**	99.56	29.43	3
ROCKET	69.38	61.85	99.57	27.03	4.25
HC1	69.17	53.84	97.19	32.87	4.25
ResNet	62.97	42.94	99.64	30.86	5.25
InceptionTime	**77.24**	33.97	**99.68**	**36.74**	2.5
TapNet	52.87	46.33	93.65	-	6.33
New Search Space	75.01	**63.68**	99.6	29.84	2.75

7 Further Work

This work outlines a new search-space for NAS which is both performant and adaptable to the diverse set of characteristics presented in time-series classification problems. We present results based on a simple random search of architectures. This gives an indication of baseline performance, which can be build upon in subsequent research. One main direction for further work is the design of optimisation algorithms for this space. Another avenue for future research is the discovery of efficient proxies for evaluation. Common methods such as super-networks are challenging to apply due to the unbounded nature of the search-space, as well as the low-data environments common to this domain.

8 Conclusion

We introduce a search space specifically designed for time-series classification tasks, which not only achieves competitive results compared with the SOTA but also produces highly efficient architectures with fewer parameters than other deep learning approaches. We search a more granular space than previous approaches, describing a larger diversity of models, which are dynamic to the task signal length. We introduce the use depth-wise and point-wise convolution as the primitive convolution operations in NAS and show that very large kernels with extensive dilation are effective for some problems.

Without the use of an advanced search algorithm or efficient proxy evaluations we achieve strong results on a range of problems. We set a benchmark for further work showing that with a well designed search-space NAS has strong potential as a time-series classification approach.

References

[1] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, *Learning transferable architectures for scalable image recognition*, 2018. DOI: 10.48550/ARXIV.1707.07012. arXiv: 1707.07012 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1707.07012.

[2] H. Liu, K. Simonyan, and Y. Yang, *DARTS: Differentiable architecture search*, 2019. arXiv: 1806.09055 [cs.LG].

[3] S. Schrodi, D. Stoll, B. Ru, R. Sukthanker, T. Brox, and F. Hutter, *Towards discovering neural architectures from scratch*, 2022. arXiv: 2211.01842 [cs.LG].

[4] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Transfer learning for time series classification," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 1367–1376. DOI: 10.1109/BigData.2018.8621990.

[5] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "Hive-cote 2.0: A new meta ensemble for time series classification," *Machine Learning*, vol. 110, no. 11-12, pp. 3211–3243, 2021.

[6] A. Dempster, F. Petitjean, and G. I. Webb, "Rocket: Exceptionally fast and accurate time series classification using random convolutional kernels," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1454–1495, 2020.

[7] M. Middlehurst, J. Large, and A. Bagnall, "The canonical interval forest (cif) classifier for time series classification," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 188–195. DOI: 10.1109/BigData50022.2020.9378424.

[8] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 1578–1585. DOI: 10.1109/IJCNN.2017.7966039.

[9] H. I. Fawaz, B. Lucas, G. Forestier, *et al.*, "InceptionTime: Finding AlexNet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, Sep. 2020. DOI: 10.1007/s10618-020-00710-y. [Online]. Available: https://doi.org/10.1007%2Fs10618-020-00710-y.

[10] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002. DOI: 10.1162/106365602320169811.

[11] R. Miikkulainen, J. Liang, E. Meyerson, *et al.*, *Evolving Deep Neural Networks*, 2017.

[12] D. Micucci, M. Mobilio, and P. Napoletano, "Unimib shar: A dataset for human activity recognition using acceleration data from smartphones," *Applied Sciences*, vol. 7, no. 10, 2017, ISSN: 2076-3417. DOI: 10.3390/app7101101. [Online]. Available: https://www.mdpi.com/2076-3417/7/10/1101.

[13] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollar, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[14] I. Radosavovic, J. Johnson, S. Xie, W.-Y. Lo, and P. Dollár, *On network design spaces for visual recognition*, 2019. arXiv: 1905.13214 [cs.CV].

[15] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 11 976–11 986.

[16] A. v. d. Oord, S. Dieleman, H. Zen, *et al.*, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[18] W. Gao, L. Zhang, Q. Teng, J. He, and H. Wu, "Danhar: Dual attention network for multimodal human activity recognition using wearable sensors," *Applied Soft Computing*, vol. 111, p. 107 728, 2021.

[19] D. Mukherjee, R. Mondal, P. K. Singh, R. Sarkar, and D. Bhattacharjee, "Ensemconvnet: A deep learning approach for human activity recognition using smartphone sensors for healthcare applications," *Multimedia Tools and Applications*, vol. 79, pp. 31 663–31 690, 2020.

[20] M. A. A. Al-qaness, A. Dahou, M. A. Elaziz, and A. M. Helmi, "Multi-resatt: Multilevel residual network with attention for human activity recognition using wearable sensors," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 144–152, 2023. DOI: 10.1109/TII.2022.3165875.

[21] A. M. Helmi, M. A. Al-qaness, A. Dahou, and M. Abd Elaziz, "Human activity recognition using marine predators algorithm with deep learning," *Future Generation Computer Systems*, vol. 142, pp. 340–350, 2023.

[22] Q. Teng, K. Wang, L. Zhang, and J. He, "The layer-wise training convolutional neural networks using local loss for sensor-based human activity recognition," *IEEE Sensors Journal*, vol. 20, no. 13, pp. 7265–7274, 2020.

[23] A. P. Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. Bagnall, "The great multivariate time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 35, no. 2, pp. 401–449, 2021.