Insights from the Use of Previously Unseen Neural Architecture Search Datasets

Rob Geada* David Towers*, [1] Matthew Forshaw[1,3] Amir Atapour-Abarghouei[2] A. Stephen McGough[1]

[1]Newcastle University, UK — [2]Durham University, UK — [3]The Alan Turing Institute, UK

rob@geada.net, amir.atapour-abarghouei@durham.ac.uk

{d.towers2, matthew.forshaw, stephen.mcgough}@ncl.ac.uk

Abstract

The boundless possibility of neural networks which can be used to solve a problem – each with different performance – leads to a situation where a Deep Learning expert is required to identify the best neural network. This goes against the hope of removing the need for experts. Neural Architecture Search (NAS) offers a solution to this by automatically identifying the best architecture. However, to date, NAS work has focused on a small set of datasets which we argue are not representative of real-world problems. We introduce eight new datasets created for a series of NAS Challenges: AddNIST, Language, MultNIST, CIFAR-Tile, Gutenberg, Isabella, GeoClassing, and Chesseract. These datasets and challenges are developed to direct attention to issues in NAS development and to encourage authors to consider how their models will perform on datasets unknown to them at development time. We present experimentation using standard Deep Learning methods as well as the best results from challenge participants.

1. Introduction

One of the main appeals of Deep Learning (DL) was its ability to democratise Machine Learning. No longer would you require a domain expert to develop an optimal solution to a given problem since DL models are capable of learning the required patterns and features from the data themselves and would not need the domain expert to identify and extract the required features. Unfortunately, rather than removing the need for an expert, the new paradigm has just shifted where the expert is required. These experts now spend their time identifying the most optimal neural network architecture for a given problem. Those who are not that proficient would select between pre-existing off-the-shelf networks such as ResNet [10], VGG [23] or Inception [25]. However, to get the best results for a particular problem, one should search across all possible solutions, not just those which may have shown good results in other problem domains, as no network is optimal for all problems. This idea of the 'no free lunch' [34] is backed up by our findings here, with some networks giving far poorer performance than what would be expected.

A DL problem can be seen as the combination of a particular task – such as classification – one wishes to apply in a particular data domain. The data domain is normally realised as a dataset for training the DL network.

Neural Architecture Search (NAS) is a fast-developing field of DL – which can be seen from the increase of published papers in the domain over the last few years [3]. The aim of NAS is to remove the need for an expert's time and knowledge to generate state-of-the-art competitive Neural Networks from a particular dataset [40]. NAS methods search through millions (if not billions) of candidate architectures from a pre-defined search space to find optimum networks. These search spaces comprise not only different network topologies but also the types of nodes which make up these networks – such as fully connected layers, convolutions and pooling. Although NAS can be applied to any data modality, the majority of work to date has focused on image-based datasets, with modern NAS methods achieving this feat in a few GPU hours [19, 37]. In addition to a pre-defined search space, a NAS approach will have a search strategy. In terms of the task, the majority of work in NAS has focused on classification problems.

NAS techniques are often evaluated by their ability to find networks that perform well on benchmark datasets, such as CIFAR-10 [12] and ImageNet [5]. These datasets, combined with standard search spaces such as NAS-Bench [6], fill an important role, allowing a direct comparison between NAS methods. While there is a clear benefit in using common benchmark datasets, we argue this is against the ethos of NAS – removing the need for domain experts, required to develop optimal architectures for a given problem.

Developing against common datasets can be seen as over-engineering a process to reach a solution we already

*These authors performed equal contribution.

know. For example, for datasets such as CIFAR-10, extensive experimentation with various neural network architectures has led to extremely high levels of accuracy [1, 8, 20], rendering this dataset easily solvable. In essence, since we already know what achieves high performance, NAS strategies may be optimised to reach these solutions, which may not generalise to other datasets. Prior work [18] has found that just because NAS methods perform well on benchmarks such as NAS-Bench-101 [38], this does not mean they will match other benchmarks.

If NAS is to truly replace architecture design experts, it must be able to find optimal networks for datasets that it has never seen before – the unseen data challenge. There is a dichotomy here in that proposing a new 'unseen' dataset for NAS may lead to that dataset being adopted as one of the core datasets for NAS. We, therefore, see our work as having two values. In the first case, we are diluting the effect of overfitting to the core datasets – having more core datasets is less likely to create a NAS approach, which can overfit to all of them. For the second case, we see this as a 'call to arms' to encourage the development and use of more datasets for NAS, which we will continue to do ourselves.

In this work, we present eight new datasets created and used for unseen dataset NAS challenges. We define how these datasets were constructed and illustrate benchmark performance metrics that have been achieved, with the intention of encouraging others in the NAS community to develop approaches that will work on more general datasets.

2. Motivation

To understand how well NAS methods might generalise to novel datasets and drive the direction of NAS research towards more impactful real-world applications, we organised a challenge tasking participants with creating NAS methods that perform well on unknown datasets. Participants were provided with a small number of "development" datasets to develop and test their NAS algorithm(s). After development, the algorithms were submitted to the competition servers, where the algorithms would be run over novel "evaluation" datasets. The datasets were created bespoke for the challenge and withheld from the participants throughout its duration, such that no participant could gain an advantage by knowing or guessing the evaluation datasets. As such, the participants are encouraged to create NAS approaches that generalise well to multiple problems rather than ones specifically targeted at one dataset.

These unseen datasets explore one of two concepts:

- Type-1: A problem an expert could solve themselves or create a program, which may include basic DL, that can outperform a naive DL approach on the raw datasets. An example of this type of data is the AddNIST dataset Sec. 4.1, one of the datasets outlined later in this paper, which consists of images with three channels, with each channel being an image from the MNIST dataset [14] having the class label as the sum (the numbers are chosen so the sum is less than twenty) of the individual MNIST digits. With prior knowledge of this dataset, a solution that attains the highest accuracy is splitting the colour channels into three separate images, using a model trained on MNIST to identify the individual numbers, and then hardcoding in the necessary arithmetic to get the final answer.
- Type-2: A problem that would be almost impossible for a human to solve or create a program without task-specific tools. An example is the Language dataset Sec. 4.2, which encodes words from ten languages into an image. The models needs to identify which letter frequency is attributed to the correct language. Without prior knowledge of what is encoded within the images and which language is represented by each class label, correctly labelling each image to the correct language would be nearly impossible. The motivation for using a type 2 dataset is to see if bespoke models can perform better than random guesswork over datasets that appear human-impossible at face value.

We believe that both types of tasks are essential for NAS algorithms to make a meaningful impact in real-world applications. First, if NAS remains incapable of solving problems that are effortlessly solvable by humans, it would be premature to assert that NAS can remove the requirement for expert knowledge. The second type relates to how machine learning is often deployed to find solutions to problems humans find complicated. Essentially, NAS methods should be able to perform at a level at least commensurate with other conventional machine learning techniques to effectively supplant manually designed models.

To further obfuscate the true nature of the datasets from the challenge participants, within our datasets, data shapes were deliberately chosen that *appeared* to correlate to normal images or well-known datasets, such as $3\times64\times64$, and data split sizes to align with well-known benchmark datasets, such as 50,000 training images to align with CIFAR. We also asked participants for their guesses as to what each development dataset was: most participants correctly described the Type-1 datasets. In contrast, none of the Type-2 datasets have been correctly identified.

3. Related Work

We consider related work within the NAS literature and the datasets commonly used in NAS research.

3.1. Neural Architecture Search

NAS aims to automate the process of neural network architecture design. Traditional network design requires extensive human expertise and significant time investment.

NAS seeks to streamline this process by employing algorithms that efficiently identify architectures tailored to specific tasks and datasets from a vast search space.

Initially, the work by Zoph and Le [40] applied NAS using reinforcement learning to CIFAR-10, achieving an architecture slightly outperforming manually designed models. NASNet [41] addressed the challenge of learning architectures directly on large datasets by transferring a building block designed for a small dataset to a larger one using ImageNet. ENAS [21] focused on computational efficiency by learning to search for an optimal sub-graph within a large graph, thus reducing computational resources. Evolutionary algorithms [22] have emerged, providing comparable results but with a distinct evolutionary overview. Weight-sharing methods [15] define over-parameterised super-networks (one-shot models), reducing computational costs, allowing the training of one super-network encompassing many different sub-architectures rather than training numerous networks independently.

Differentiable NAS [9, 16] marks another significant advancement, allowing gradient-based optimisation methods to explore the neural architecture search space efficiently. Each of these developments in NAS represents a significant stride towards the more efficient, automated and effective design of neural network architectures; however, the effectiveness of these approaches is often only tested on common benchmark datasets, some of which we will briefly cover in the following sections. The real-world applicability of these approaches needs to be tested on "unseen" datasets, which better simulate what is needed of NAS in practice.

3.2. Common NAS datasets

Although any dataset can be used to evaluate NAS methods, CIFAR-10 [12], and ImageNet [5] are highly popular.

CIFAR-10 [12] comprises 60,000 images from ten classes evenly distributed in training and testing sets. Accuracy on CIFAR-10 is often used to evaluate performance, and it is also used in NAS, with the very first NAS algorithm [40] highlighting the potential power of NAS by demonstrating its ability to achieve better CIFAR-10 performance than human-designed models. While CIFAR-10 is a good dataset for comparing performance, it is an easy problem, with modern architectures and NAS methods easily achieving accuracies over 90%. Furthermore, since CIFAR-10 is used as a standard benchmark, new methods are encouraged to focus on this dataset, which may come at a cost of generalisability across other datasets.

ImageNet [5] is another popular dataset for evaluation and benchmarking. It contains 1,000 classes and presents a more challenging problem, with the results often presented in tandem with CIFAR-10 results to demonstrate model performance. ImageNet poses similar issues to CIFAR-10 due to its similar prevalence in benchmarking DL methods. While including ImageNet and CIFAR-10 as datasets for NAS comparison demonstrates that the NAS approach is not overfitting to just one dataset, it does not solve the problem of generalisation. We argue that both more datasets and datasets that the developers are unaware of are needed.

3.3. NAS Benchmark Suites

To make NAS development more comparable and easier, benchmark suites such as NAS-Bench-101 [38], NAS-Bench-201 [7], and NATS-Bench [6] have been developed. In these benchmarks, the performance achieved for each network in the search space has been pre-computed, thus removing the model training step in NAS approaches.

Benchmark suites allow developers to focus on the development of search strategies and allow for easy comparison between different approaches as they work directly on the same search space. Thus, performance gains over other methods cannot be attributed to different search spaces.

These benchmark suites suffer from the same issues of reliance on CIFAR-10 and ImageNet. The search spaces used by these benchmarking tools have been fully explored on CIFAR-10 and ImageNet. These results are stored so a NAS method can look up an architecture's final result without having to retrain fully. These lookup tables do not exist for other datasets, meaning that NAS methods developed using these tools can quickly generate results for CIFAR-10 and ImageNet but not easily for other datasets.

4. Dataset Descriptions

The code to generate our datasets is available at our GitHub page[1]. Each dataset comprises six NumPy files, X (images) and Y (labels) files for the training, validation, and testing sets, and a metadata file. This metadata file includes the shape of the training data, the ResNet-18 [10] benchmark result, and the number of classes in the data. Further details and examples are provided in the Appendices.

4.1. AddNIST

AddNIST [27], is a Type-1 dataset of 70,000 images with an image shape of $3\times28\times28$ (channels first). The training, validation, and testing split is 45,000, 15,000, and 10,000, respectively. Each colour channel is an image from MNIST [14]. An example AddNIST image is depicted in Fig. 1 (left), the larger image, shows the image as is, with the three channels laid on top of each other, and three images to the right show each channel more clearly.

AddNIST has twenty classes (0 - 19), with the class derived from the MNIST label of each channel, such that $l = (r + g + b) - 1$, where l is the image label and r, g, and b are the respective MNIST labels of each colour channel.

[1] github.com/Towers-D/NAS-Unseen-Datasets

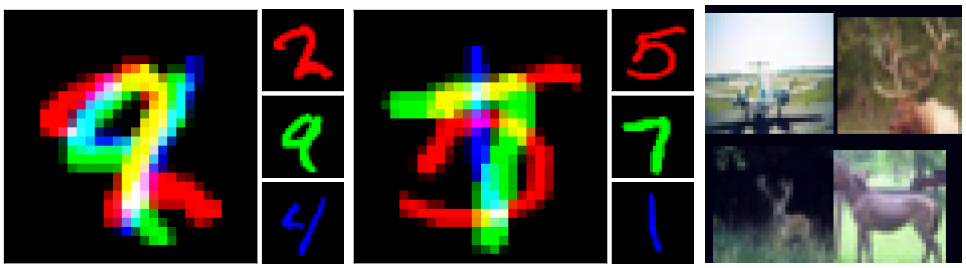

Figure 1. *Left:* AddNIST image - the sum of the channels adds up to 15 (r = 2, g = 9, b = 4), implying a label of 14. *Middle* MultNIST image - the product of the channels equals 35 (r = 5, g = 7, b = 1), which implies a label of 5 (35 %10 = 5). *Right* CIFARTile image - two deer, one aeroplane, and one horse, meaning there are three unique labels among the sub-images, which equates to a final label of 2

AddNIST adds complexity to the MNIST data. In MNIST, the goal is to identify what numerical digit is seen within the image, while in AddNIST, not only does the model have to identify the digit in each channel, but it must also learn to sum up these values in a specific manner to identify the class correctly. In this manner, AddNIST was designed around the research question of whether NAS could "figure out" a calculation was required and apply it.

While the task this dataset puts forward is still rather simple, which is expected for a Type 1 dataset, it requires high-level inference capabilities within the model, which would mean the NAS algorithm searching for the optimal architecture would have to consider the capacity of the model to encompass the required function.

4.2. Language

Using the open-source, public spell checker ASPELL, which can be found at aspell.net, we created Language [32], a Type-2 dataset.

Using dictionaries from 10 languages that use the Latin alphabet (English, Dutch, German, Spanish, French, Portuguese, Swedish, Zulu, Swahili, and Finnish), all six-letter words within each language are extracted, and any words with letters that use diacritics (such as é or ü) or include 'y' or 'z' are subsequently removed.

Each image is generated by first selecting one of the ten languages. We then randomly select four words that we filter from the selected dictionary and concatenate them into a 24-character string. Since we eliminated letters with diacritics and y and z, we have a remaining alphabet of twenty-four letters. This allows for a 24×24-grid encoding the string as a 1×24×24 image. Using the y-axis to denote the index of the string and the x-axis to denote the character, we construct the image so each black pixel denotes the letter used in that position. See Fig. 2 (top) for an example. The training, validation, and testing split is 50,000, 10,000, and 10,000 images, respectively.

The Language research question pertained to whether a linguistic character encoding contained enough information for DL to correctly identify the original language, requiring the model to perform linguistic analysis. To ensure no leakage between the training and testing data, when selecting a word for the training data, we ensure that this word does not appear in the validation or testing data and vice versa.

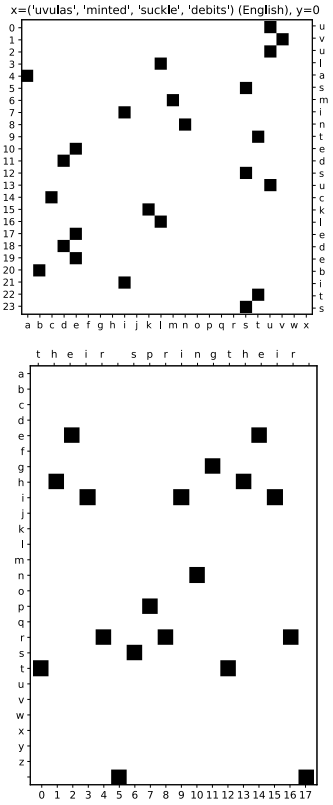

Figure 2. *Top:* An example of a Language image with a readable axis included. The right-hand axis contains the four six-letter words "Uvulas", "Minted", "Suckle", and "Debits", which are all English words given the label 0. *Bottom:* An example Gutenberg image, the words "their", "spring", and "their" have been encoded from one of Shakespeare's works which give the label 4

4.3. MultNIST

MultNIST [33] is a Type-1 dataset similar in concept to AddNIST (Sec. 4.1), originating from the same research question. Like AddNIST, MultNIST images have a

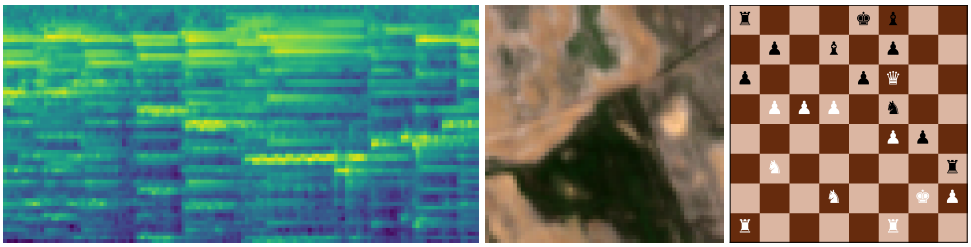

Figure 3. *Left:* An example of an Isabella generated using a piece of music labelled as "20th Century"(0). *Middle* An example of the GeoClassing dataset showing a photo taken over Portugal (9). *Right:* An example rendering of a board position in the Chesseract dataset wherein white goes on to eventually win and is thus given a label of White Wins (0)

3×28×28 channel first shape, and each channel is an image from the MNIST dataset [14]. See Fig. 1 (middle) for an example. The training, validation, and testing split is 50,000, 10,000, and 10,000 images, respectively.

MultNIST has only ten classes (0 - 9). The label of each MultNIST image is calculated such that $l = (r * g * b) \mod 10$ where l is the image label and r, g, and b are the respective MNIST labels of each colour channel.

In AddNIST, the class was directly affected by how large the numbers were in each channel. The only way for an image to have a label of 0 was if one colour channel contained an MNIST with label 1 and the remaining channels contained MNIST images for 0. MultNIST introduces the additional complexity over MNIST by retaining a calculation but removes the bias of larger numbers to higher classes.

4.4. CIFARTile

CIFARTile is a Type-1 dataset where each image in the CIFARTile [29] dataset is a compilation of four CIFAR-10 [12] images tiled together in a grid resulting in images which are of size 3×64×64. See Fig. 1 (right) for an example. The training, validation, and testing split is 45,000, 15,000, and 10,000 images, respectively.

There are four classes (0 - 3), which represent the number of CIFAR-10 classes in each grid minus one. For example, a grid consisting of CIFAR-10 images with labels [*horse*, *horse*, *frog*, *car*] has three distinct classes, and thus a label of $3-1=2$. As such, CIFARTile is a Type-1 dataset; any human should be able to identify and solve this task.

CIFARTile requires a model to identify multiple CIFAR-10 classes simultaneously and determine which are similar and which differ. Whether models could adequately do this is the driving research question behind this dataset.

4.5. Gutenberg

Gutenberg [31] is a Type-2 dataset named after the source of the data, Project Gutenberg, found at www.gutenberg.org, which provides free ebooks of literary works that are no longer under US copyright protection. We selected six authors (Thomas Aquinas, Confucius, Hawthorne, Plato, Shakespeare, and Tolstoy) and downloaded several books from each author – see the Appendices for further details and examples.

The selected works are English translations that represent a variety of cultures and time periods. We perform basic text preprocessing, including removing punctuation, converting letters with diacritics to the base letter, and removing "structure" words (e.g., 'Chapter', 'Scene', 'Prologue'). We then extracted consecutive sequences of three words between 3 and 6 letters long. In each sequence, the three words were padded up to 6 characters with spaces. Then, the three words were concatenated together to produce an 18-character string. These strings were used as the base for image creation. Training, test, and validation sequences were chosen such that there was no overlap between any sequence across any data split.

Similar to the Language dataset (Sec. 4.2), we converted these strings into images. On a 27×18 grid, we created a mapping of each character in the string. See Fig. 2 (bottom) for an example. The x-axis represents the position in the string, and the y-axis represents the alphabetical letter (or space) located at that position. This results in the shape being 1×27×18. The task is then to predict which of the original six authors the original word sequence came from. The training, validation, and testing split is 45,000, 15,000, and 6,000, respectively, to prevent participants from exploiting the standard distribution we used for the other datasets.

The research question behind this dataset was whether specific spatial patterns of letters were sufficient information for a neural network to identify the authors from which they originated, which seems impossible from face value.

4.6. Isabella

Isabella[2] is a Type-2 dataset that uses musical recordings from the Isabella Stewart Gardner Museum, Boston.

The four classes of this dataset refer to the era of composition (Baroque, Classical, Romantic, and 20th Century) attributed by the museum. The recordings are split into five-second snippets converted into 64-band spectrograms, creating a dataset of 1×64×128 shaped images. See Fig. 3

[2]Code to generate Isabella is available at github.com/Towers-D/NAS-Unseen-Datasets.

Dataset	ResNet-18	AlexNet	VGG16	ConvNext	MNASNet	DenseNet	ResNeXt	Random
AddNIST	92.08%	**94.87%**	92.06%	38.06%	90.51%	93.52%	91.42%	5%
Language	87.00%	85.71%	84.54%	83.40%	84.63%	84.57%	**93.97%**	10%
MultNIST	91.55%	**94.01%**	90.43%	64.20%	87.70%	92.81%	90.57%	10%
CIFARTile	45.56%	48.88%	24.43%	31.06%	48.49%	**51.28%**	46.23%	25%
Gutenberg	40.98%	**45.53%**	44.00%	31.93%	38.00%	43.28%	40.30%	$16.\dot{6}\%$
Isabella	62.02%	61.37%	58.13%	57.18%	60.69%	**63.27%**	60.46%	25%
GeoClassing	80.33%	92.49%	83.67%	72.76%	86.00%	**94.21%**	89.99%	10%
Chesseract	57.83%	57.45%	55.69%	52.74%	56.26%	**59.60%**	55.15%	$33.\dot{3}\%$

Table 1. Experiments on commonly-used CNN-based classification models [10, 11, 13, 17, 23, 26, 35] using the presented datasets.

(left) for an example. The training, validation, and testing split is 50,000, 10,000, and 10,000 images, respectively.

The task for the models is to predict the era of composition from the spectrogram. No recording used in a training snippet appeared in the validation or test sets. The research question behind this dataset was to explore whether era-defining characteristics appear at the spectrogram level and whether CNNs could reliably identify such patterns.

4.7. GeoClassing

GeoClassing [30], known as Sadie in the competition, is a Type-2 dataset that takes advantage of the BigEarthNet dataset [24] as a foundation. The BigEarthNet dataset, as originally published, consists of satellite photography with ground-cover classification labels, i.e., "airports" or "vineyards". An example image is shown in Fig. 3 (middle).

The GeoClassing dataset uses these images, but instead of using the given ground-cover labels, we identified the European Space Agency Sentinel patch from which the BigEarthNet image was sourced. We cross-referenced the coordinates of that patch within a map of Europe to identify the corresponding country depicted in the image. This gave us ten classes (Austria, Belgium, Finland, Ireland, Kosovo, Lithuania, Luxembourg, Portugal, Serbia, and Switzerland).

Each image has a shape of $3\times60\times60$, and the training, validation, and testing split contains 43,821, 8,758, and 8,751 images, respectively. The research question is whether NAS can find models that identify countries from differences in topology and ground coverage, which, without specific knowledge of geography and vegetation type, would be extremely difficult for a human to do manually.

4.8. Chesseract

Chesseract [28], a Type-1 dataset wherein we accessed public chess games from eight grandmasters (Bobby Fischer, Garry Kasparov, Magnus Carlsen, Viswanathan Anand, Hikaru Nakamura, Anatoly Karpov, Fabiano Caruana, and Mikhail Tal) extracting the final 15% of board states. These positions were then one-hot encoded by piece type and colour, creating a $12\times8\times8$ image. As a 12-channel image is hard to render, we provided a flattened 2D image of one of the chess boards we used Fig. 3 (right). A 3D visualisation can be seen in the Appendices.

The training, validation, and testing split is 49,998, 9,999, and 9,999 images, respectively. Each image in the Chesseract dataset is one of three classes (White wins, Draw, Black wins). No individual positions from the same game appeared across the data splits. This dataset requires the model to identify the game's final result from the position. Chesseract presented a problem for some participants of the challenge. This was because Chesseract uses twelve instead of the usual 1 or 3 channels applied for image datasets, which often caused errors for hardcoded algorithms that only accepted 1 or 3 channel dimensions.

Machine learning is already prominent in chess, with competing learning-based chess engines that beat grandmaster players, such as AlphaZero [4]. Our research question for this dataset was based on the understanding that any position can be analysed to show whether black or white has the advantage. Is NAS able to develop a DL network to identify this from just an encoding of the board position?

5. Baseline Experimentation

Here, we provide some baseline results for our datasets to motivate future NAS experimentation. We provide a set of baseline results across several well-known CNNs and three NAS methods across two search spaces.

5.1. CNN Experiments

To generate baseline results of our datasets for future work comparison, we have performed experiments using commonly used off-the-shelf CNNs, which can be seen in Tab. 1. In all our CNN experiments, we have followed a similar experimental setup. We use stochastic gradient descent as our optimiser with an initial learning rate of 0.01, momentum of 0.9, and a weight decay of 3×10^{-4}. We have used a Cosine Annealing Learning rate with the max number of iterations equaling 64, the number of epochs. Cross Entropy is used as the Loss function.

ResNet-18 is used as the baseline throughout our competition, and any NAS method would be expected to easily outperform this architecture. As you can see in Tab. 1,

				Random Search		
Dataset	**PC-DARTS**	**DrNAS**	**Bonsai-Net**	**DARTS**	**Bonsai**	**Random**
Addnist	96.60%	97.06%	**97.91%**	97.07%	34.17%	5%
Language	**90.12%**	88.55%	87.65%	90.12%	76.83%	10%
MultNIST	96.68%	**98.10%**	97.17%	96.55%	39.76%	10%
CIFARTile	**92.28%**	81.08%	91.47%	90.74%	24.76%	25%
Gutenberg	**49.12%**	46.62%	48.57%	47.72%	29.00%	16.$\dot{6}$%
Isabella	**65.77%**	64.53%	64.08%	66.35%	58.53%	25%
GeoClassing	94.61%	**96.03%**	95.66%	95.54%	63.56%	10%
Chesseract	57.20%	58.24%	**60.76%**	59.16%	68.83%	33.$\dot{3}$%

Table 2. Experimental results on PC-DARTS [36], DrNAS [2], Bonsai-Net [9] as well as random search.

ResNet-18 produces competitive results across our datasets, though it does not excel on any particular dataset. ResNet-18 particularly struggles with the GeoClassing dataset compared to the others, only beating ConvNext in our trials.

AlexNet [13] shows decent performance across all datasets, achieving the highest accuracy on three of the eight datasets (AddNIST, MultNIST, and Gutenberg). AddNIST and MultNIST are similar, both being derived from MNIST. This may suggest that AlexNet is particularly good at working with MNIST Images. AlexNet captures spatial hierarchies in images well and is thus suitable for the clear, structured layouts of MNIST digits. VGG16 [23], on the other hand, does not achieve particularly impressive levels of accuracy on any of the datasets, though it does not perform badly, either. The only exception is the CIFARTile dataset, where VGG suffers and only achieves roughly the same accuracy as random chance, as seen in Tab. 1.

ConvNext [17] performs poorly across our datasets, most likely since our datasets may not exhibit the specific spatial complexity that ConvNext is optimised for. However, it is important to note that for purposes of generating our CNN baselines, we have kept the experimental setup the same across CNNs. Thus, hyperparameter tuning may be beneficial to ConvNext. We have used the base version of ConvNext in our experiments.

We also perform the same experiments on MnasNet [26] using a depth multiplier of 1.0. Despite being a lightweight architecture designed for mobile efficiency, MnasNet achieves competitive results across our datasets. It has strong generalisation capabilities since it uses a reinforcement learning approach to automate architecture design, thus performing well on unseen datasets [26].

DenseNet [11] performs significantly well on the latter three datasets (Isabella, GeoClassing, and Chesseract) as well as CIFARTile, achieving the highest accuracy of the CNNs over these datasets. With the exception of Chesseract, the datasets that DenseNet excels on are the most memory-intensive datasets. This is due to the DenseNet mechanism allowing efficient re-use of features through dense connectivity and ensuring maximal information flow between layers in the network [11]. For memory-intensive datasets, which often contain complex and rich information, this feature reuse allows DenseNet to exploit the data more thoroughly and efficiently, leading to better performance. The DenseNet-161 architecture is used in our experiments.

Finally, ResNeXt [35] performs fairly well across our datasets, but especially well on the Language Dataset, scoring almost 7% higher than the other methods. The particular version of ResNeXt we used is ResNeXt-50 (32 x 4d), which has a high level of cardinality (the size of the set of transformations) [35]. This high cardinality allows ResNeXt to learn more complex and diverse representations, which is crucial for distinguishing subtle patterns in the Language Dataset, such as the frequency and arrangement of characters encoded in images.

Our analysis across these CNN architectures reveals distinct performance trends tied to the nature of each dataset. ResNet-18 serves as a versatile baseline, while AlexNet is especially capable of handling structured patterns. VGG16's mixed results and ConvNext's overall underperformance emphasise the need within conventional deep learning for matching architectural strengths to dataset characteristics, which is not desirable for evaluating generalisation capabilities in general and NAS algorithms in particular. DenseNet and ResNeXt are good at handling memory-intensive datasets and those with nuanced patterns, respectively. These insights confirm that the effectiveness of a CNN architecture is highly contingent on the specific demands of the dataset, which underscores the need for unseen datasets for effective evaluation of NAS methods.

5.2. NAS Experiments

In addition to the experiments performed on the CNNs, we have also applied the NAS methods of PC-DARTS [36], DrNAS [2] (DARTS-space), and Bonsai-Net [9] to our datasets. Furthermore, as a baseline of comparison, we perform a random search in both the DARTS and Bonsai search spaces, using the provided methods from PC-DARTS and Bonsai-NET. These results can be found in Tab. 2.

During our random search experiments using the DARTS search space, it is noted that random search often outperforms either PC-DARTS or DrNAS and outper-

Dataset	CNN	NAS	Competition
AddNIST	94.87%	**97.91%**	95.06%
Language	**93.97%**	90.12%	89.71%
MultNIST	94.01%	**98.10%**	95.45%
CIFARTile	51.28%	**92.28%**	73.08%
Gutenberg	45.53%	49.12%	**50.85%**
Isabella	63.27%	**65.77%**	61.42%
GeoClassing	94.21%	96.03%	**96.08%**
Chesseract	59.60%	60.76%	**62.98%**

Table 3. Results on each dataset from our CNN and NAS experiments, and Competition Submissions

forms both models on AddNIST, Chesseract and Isabella, tying PC-DARTS on Language while outperforming DrNAS. These results support findings by Yu et al. [39] that searches using these search spaces often do not significantly outperform random search.

We can see in Tab. 2 that PC-DARTS [36] performs well, scoring the highest accuracies over four of the eight datasets and gives competitive results over the others. DrNAS [2], however, scores the highest on two of the eight datasets but struggles with CIFARTile, where it scores 10% lower than the other methods. We used the DARTS search space during our DrNAS experiments; while DrNAS is also available to work on the NAS-Bench-201 [7] search space, it employs the NAS-Bench API to look at the performance of architectures fully trained on CIFAR-10, which would not reflect the performance these architectures found on our datasets. In most cases, Bonsai-Net [9] outperforms random search, with the exception of Chesseract. It returns the best results on AddNIST and Chesseract (though it is significantly beaten by random search on the latter dataset).

The varied performance of these NAS methods across our datasets highlights their ability to challenge and assess the adaptability and effectiveness of different architecture search strategies. This underscores the strengths of our datasets in providing a comprehensive benchmark for evaluating NAS methods' capability to generalise across a diverse range of tasks and complexities.

6. Discussion

Tab. 3 shows the best-reported performance of the CNN and NAS experiments on each dataset, as well as the best result found by competition participants; for further information, see the Appendices. From these results, we can see that NAS found the best model for seven datasets, supporting the usage of NAS methods in finding good models for given datasets. Three of these architectures were found by competition submissions, NAS methods designed for unseen data; given the competition's time restrictions this may suggest that methods designed in this way can generalise to other datasets better.

It is important to mention that we did not test the deepest versions of these CNN architectures or perform hyperparameter optimisation. Similarly, while we allowed the traditional NAS methods to run in their preset configuration, the competition participants were only given twenty-four hours to evaluate across three datasets. However, even with these limitations, CNNs and NAS architectures from the competition outperformed traditional NAS methods across several datasets, reinforcing our belief that current NAS methods do not necessarily generalise properly to unseen datasets.

7. Rights and Reproduction

These datasets have been created under the licence agreements of the original data. Where available, we have made the datasets publicly accessible under an CC BY 4.0 Licence. The Isabella dataset uses data from the Isabella Stewart Gardner Museum, which withholds the right to share modifications to the music they have made available. Instead of providing the dataset, we provide a Python script to convert music files obtained from the Museum[3] into the format of the competition dataset on our GitHub [4].

8. Conclusion

Machine learning is a valuable and fast-growing tool that is quickly becoming part of the tools we use daily. For small businesses to stay competitive, they need to be able to easily include machine learning techniques in their products or business strategy. NAS promises to be a tool that removes the cost of an expert's time and knowledge to create bespoke neural networks while remaining competitive. NAS is currently evaluated primarily on a few benchmark datasets and developed based on these datasets as well. This does not reveal how generalisable NAS methods are when given a dataset, especially when the problem is distinctly different.

This paper introduces eight new datasets to be used when testing NAS methods. These datasets represent problems that are either simple or difficult for humans to solve and provide difficulty outside of normal image classification. While we believe using these datasets will improve the generalisability of NAS methods, the problem of not knowing whether NAS is good in general or only good on benchmark datasets is a rolling problem. Simply including these datasets for upcoming NAS works will mean they become part of the datasets developed in mind. To solve this problem, more datasets will be continually needed.

In the future, we seek to develop further datasets to enable people to work on 'unseen' data for NAS. We will also evaluate these datasets against pre-existing, and NAS approaches we are developing.

For a video overview of the paper, please follow this link: youtu.be/YdYHdxNZUIw

[3] Available here https://www.gardnermuseum.org/experience/music
[4] github.com/Towers-D/NAS-Unseen-Datasets

References

[1] Andrea Gesmundo and J. Dean. An Evolutionary Approach to Dynamic Introduction of Tasks in Large-scale Multitask Learning Systems. *arXiv preprint arXiv:2205.12755*, 2022. 2

[2] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. DrNAS: Dirichlet Neural Architecture Search. In *International Conference on Learning Representations*, 2021. 7, 8

[3] Colin White, Mahmoud Safari, Rhea Sanjay Sukthanker, Binxin Ru, Thomas Elsken, Arbër Zela, Debadeepta Dey, and Frank Hutter. Neural Architecture Search: Insights from 1000 Papers. *arXiv preprint arXiv:2301.08727*, 2023. 1

[4] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm. *arXiv preprint arXiv:1712.01815*, 2017. 6

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1, 3

[6] Xuanyi Dong, Lu Liu, Katarzyna Musial, and Bogdan Gabrys. Nats-bench: Benchmarking nas algorithms for architecture topology and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3634–3646, 2022. 1, 3

[7] Xuanyi Dong and Yi Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020. 3, 8

[8] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 2

[9] Rob Geada, Dennis Prangle, and Andrew Stephen McGough. Bonsai-Net: One-Shot Neural Architecture Search via Differentiable Pruners. *arXiv preprint arXiv:2006.09264*, 2020. 3, 7, 8

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 3, 6

[11] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 6, 7

[12] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009. 1, 3, 5

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 6, 7

[14] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010. 2, 3, 5

[15] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. In Ryan P. Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 367–377. PMLR, 2020. 3

[16] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 3

[17] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11976–11986, 2022. 6, 7

[18] Yash Mehta, Colin White, Arber Zela, Arjun Krishnakumar, Guri Zabergja, Shakiba Moradian, Mahmoud Safari, Kaicheng Yu, and Frank Hutter. NAS-bench-suite: NAS evaluation is (now) surprisingly easy. In *International Conference on Learning Representations*, 2022. 2

[19] Joseph Mellor, Jack Turner, Amos Storkey, and Elliot J. Crowley. Neural architecture search without training. In *International Conference on Machine Learning*, 2021. 1

[20] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. 2

[21] Hieu Pham, Melody Y. Guan, Barret Zoph, Quoc V. Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. In *ICML*, 2018. 3

[22] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2902–2911. JMLR.org, 2017. 3

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 6, 7

[24] Gencer Sumbul, Marcela Charfuelan, Begüm Demir, and Volker Markl. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 5901–5904, 2019. 6

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with

convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 1

[26] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2815–2823, Long Beach, CA, USA, 2019. IEEE. 6, 7

[27] David Towers, Rob Geada, Amir Atapour-Abarghouei, and Andrew Stephen McGough. AddNIST Dataset, 2023. https://data.ncl.ac.uk/articles/dataset/AddNIST_Dataset/24574354/1. 3

[28] David Towers, Rob Geada, Amir Atapour-Abarghouei, and Andrew Stephen McGough. Chesseract Dataset, 2023. https://data.ncl.ac.uk/articles/dataset/Chesseract_Dataset/24118743/2. 6

[29] David Towers, Rob Geada, Amir Atapour-Abarghouei, and Andrew Stephen McGough. CIFARTile Dataset, 2023. https://data.ncl.ac.uk/articles/dataset/CIFARTile_Dataset/24551539/1. 5

[30] David Towers, Rob Geada, Amir Atapour-Abarghouei, and Andrew Stephen McGough. GeoClassing Dataset, 2023. https://data.ncl.ac.uk/articles/dataset/GeoClassing_Dataset/24050256/3. 6

[31] David Towers, Rob Geada, Amir Atapour-Abarghouei, and Andrew Stephen McGough. Gutenberg Dataset, 2023. https://data.ncl.ac.uk/articles/dataset/Gutenberg_Dataset/24574753/1. 5

[32] David Towers, Rob Geada, Amir Atapour-Abarghouei, and Andrew Stephen McGough. Language Dataset, 2023. https://data.ncl.ac.uk/articles/dataset/Language_Dataset/24574729/1. 4

[33] David Towers, Rob Geada, Amir Atapour-Abarghouei, and Andrew Stephen McGough. MultNIST Dataset, 2023. https://data.ncl.ac.uk/articles/dataset/MultNIST_Dataset/24574678/1. 4

[34] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997. 1

[35] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6, 7

[36] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial Channel Connections for Memory-Efficient Architecture Search. In *International Conference on Learning Representations*, 2020. 7, 8

[37] Peng Ye, Baopu Li, Yikang Li, Tao Chen, Jiayuan Fan, and Wanli Ouyang. b-darts: Beta-decay regularization for differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10874–10883, 2022. 1

[38] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. NAS-bench-101: Towards reproducible neural architecture search. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7105–7114. PMLR, 2019. 2, 3

[39] Kaicheng Yu, Christian Sciuto, Martin Jaggi, Claudiu Musat, and Mathieu Salzmann. Evaluating the search phase of neural architecture search. In *International Conference on Learning Representations*, 2020. 8

[40] Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*, 2017. 1, 3

[41] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3