DDS-NAS: Dynamic Data Selection within Neural Architecture Search via On-line Hard Example Mining applied to Image Classification

Matt Poyser[1], Toby P. Breckon[1,2]
[1]Department of Computer Science, Durham University, UK.
[2]Department of Engineering, Durham University, UK.

1. Introduction

Following the emergence of big data and the ever-increasing public availability of datasets, each with tens of thousands of data points, research within the deep learning domain is accelerating [1]. Consequently, there are two key factors that need to be addressed. Firstly, the process by which we present data to the deep learning model is paramount. It is not uncommon for models to be trained for thousands of epochs, and thus any superfluous data within the dataset will have an increasingly negative impact on training speed. This phenomenon has given rise to hard example mining [2], which attempts to identify *hard* images (i.e. images that contribute highly to loss, upon which the model performs poorly). By only considering these hard images, we can not only sample from a minimal dataset, therefore minimising the duration of a training epoch, but also reduce the number of iterations required for model convergence, as the contribution of each image sample is maximised in every iteration.

Similarly, the images sampled by the model in any given training iteration are controlled via curriculum learning [3] and self-paced learning [4]. Contrary to hard example mining, in which commonly only a subset of the global dataset is considered during the entire training process, curriculum learning and self-paced learning forces the initial iterations to sample one fraction of the global dataset, and subsequent iter-

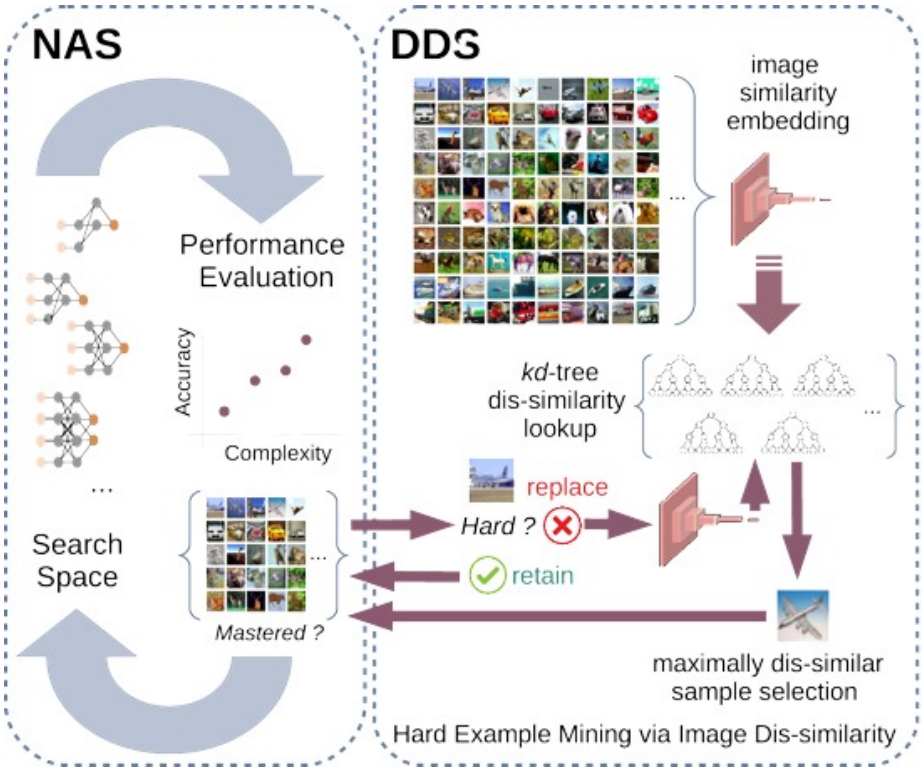

Figure 1: Overview of the DDS-NAS search phase. After a given training iteration, we determine whether a sufficient percentage of the data in the current subset is correctly classified, according to some *a priori mastery* threshold. If the subset has been mastered, we reformulate it dynamically. *Hard* images in the current subset are retained, according to some *a priori hard*-ness threshold, while easy images are replaced with the most different image from the same class. To determine the most different image, we employ an (approximate) furthest-neighbour *kd*-tree whereby each image is represented by the auto-encoded representation of its features within the latent space.

ations to sample from different fractions, until the entire global dataset is considered. Generally, curriculum learning introduces harder images (pre-defined by prior knowledge) as training progresses, while self-paced learning determines the current model performance as feedback to the controller to determine which images to sample next.

The second challenge arising from data accessibility is the evolution of the architecture search space. As research within the domain continues, newer, and often more complex network architectures are presented. To overcome this notion Neural Architecture Search (NAS) has emerged, which automatically traverses the architecture

search space for a given task and generates models that are competitive alongside hand-crafted, state-of-the-art models [5]. We can divide the NAS domain into evolutionary, reinforcement-learning, prediction-based, and gradient-based NAS frameworks. This paper primarily considers gradient-based NAS frameworks.

More precisely, the seminal gradient-based DARTS [6] framework constructs a super-network, in which each layer consists of all possible operations in the search space, followed by a softmax layer across said operations, such that operation selection can be represented as (continuous) operation-magnitude optimisation. After training the super-network, the best-performing subset of operations are extracted, thus formulating a cell (sub-network). A series of these cells is then trained to generate a final 'searched' model, fine-tuned upon a given challenge dataset.

We propose a strategy that incorporates a novel combined hard example mining and curriculum learning approach to enable Dynamic Data Selection (DDS) within a NAS framework, denoted as DDS-NAS. By using image similarity as a proxy metric for image difficulty (on an easy to hard performance axis), we can select *hard* images for processing within a given NAS training iteration in logarithmic time without compromising image diversity (Fig. 1). This process allows us to significantly improve the NAS search phase speed. Whilst this paper specifically addresses image datasets, there is no reason not to apply identical techniques to other application domains such as natural language processing (NLP).

On this basis, our main contributions are as follows:

- a novel framework, DDS-NAS, that incorporates both hard example mining and curriculum learning in order to minimise the training duration of a given epoch within NAS, demonstrated to be effective across a variety of commonplace NAS approaches (DARTS [6], P-DARTS [7] and TAS [8]).
- an efficient and novel approach for hard example mining within the image domain, that considers image dissimilarity an alternative metric to hardness, and

employs an autoencoder architecture that enforces an image similarity embedding in latent space. This yields efficient dissimilar image look-up from a *kd*-tree structure.

– generation of models in a manner intrinsically robust to biased datasets, and 10 times quicker than existing NAS techniques, whilst retaining competitive, near state-of-the-art accuracy with minimal memory footprint over common benchmarks.

2. Prior Work

In this section we introduce the related NAS, hard example mining, and curriculum learning approaches, from which we draw our methodology. We restrict our NAS literature survey to only a brief overview of NAS techniques, since DDS-NAS can be deployed upon any NAS approach that iteratively processes images (evolutionary, reinforcement-learning, and gradient-based). Through this review of the literature, we highlight our contribution to the field, outlining the ways in which our framework works alongside the current NAS approaches to optimise performance and reduce computational requirements.

2.1. Neural Architecture Search

With the rise of NAS, a multitude of recent literature has addressed the scalability challenge which occurs due to the resultant large search space and training cost. Following the seminal work of Zoph et al. [9] and other reinforcement learning [10, 11], and evolutionary [12, 13] approaches to NAS, weight-sharing techniques [14] reduce the need to train each architecture in the search space separately.

2.2. NAS Strategies

Gradient-based approaches [6, 7, 15] enable the application of stochastic gradient descent and other well-used deep learning techniques by relaxing the search space so

that it is continuous, thereby drastically improving the convergence rate of the architecture. One-shot NAS approaches employ the weight-sharing super-network training stage of DARTS, with an alternative sampling strategy, tending to consider only one path through the super-network in a given training iteration [16].

Progressive DARTS (P-DARTS) [7] address the optimisation gap within DARTS between the sub-network and final model. This is achieved by simultaneously limiting the prevalence of skip-connections within a generated cell and by progressively reducing the operation search space available to the super-network. This in turn enables progressively increasing network depth.

Network Pruning via Transformable Architecture Search (TAS) [8] crafts a loss function to directly minimise the complexity of the searched network. To this end, both the width (number of channels in a layer) and the depth (number of layers) of the network are also searched. By employing the knowledge distillation algorithm from [17], weights from the fully trained super-network can be transferred to the 'pruned' searched network.

2.3. Curriculum and Coreset Sampling Within NAS

CNAS [18] employs a curriculum learning framework within NAS architecture, in order to slowly introduce new operations to the NAS controller search space, allowing the model to successfully master harder tasks as training progresses. Overall, network topology is the primary focus for contemporary NAS solutions [8, 18, 7]. By contrast, only minimal consideration of the dataset presented within the NAS pipeline is present in the literature.

CLOSE [19] uses curriculum learning to modify the sharing extent. There is no effort to reduce the training dataset size, but image hardness and uncertainty (which can be calculated from a range of different sub-network outputs) is factored into the loss computation.

Peng et al. [20] introduce negative samples within NAS training, drawing from

the benefits of contrastive learning. Core-set Sampling [21] select a small subset of the data space for training the NAS super-network via the greedy k-center algorithm. ADAPTIVE-NAS [22] compares different core-set sampling algorithms for PT-DARTS [23], including adaptive sampling, in which the training set is periodically updated using GLISTER [24]. While their work is most similar to ours, there is no effort to consider image hardness, and is thus unable to utilise any benefits of curriculum learning. Moreover, only one search algorithm is evaluated with core-set selection. However, the core-set selection algorithm depends upon embeddings that are well aligned with the training data, much like DDS-NAS (Table 5).

To our knowledge, this paper represents the first approach to jointly employ online hard example mining and curriculum learning during NAS learning to optimise both model performance and reduce overall NAS computation requirements. With the variety and quickly evolving nature of NAS strategies, it is imperative that our method can be deployed alongside any existing NAS approach. Our work is thus the first to utilise a core-set approach in conjunction with a variety of existing NAS approaches and different architecture search spaces. Our approach is able to accelerate training for even the oldest NAS methods, for which training speed is a known drawback [16].

2.4. Curriculum Learning and Hard Example Mining

Graves et al. [25] posit the need for a surrogate measure of learning progress to inform the curriculum controller, rather than model accuracy. They introduce several different measures, identifying the best as prediction gain (instantaneous loss for a sample) and gradient variational complexity (using the direction of gradient descent to measure model complexity).

Hachoen and Weinshall [4] suggest instead to use a scoring function to generate the curriculum. The scoring function ranks images within the dataset by difficulty through testing either the same model (pre-trained without curriculum learning) or a different model. Harder images are introduced to the model over time. Weinshall et al. [26]

further evolve this process to consider image difficulty in relation to task difficulty (e.g. fine detail differentiation is harder than coarse detail differentiation, which can for instance be trivially approximated with hierarchical datasets). Shrivastava et al. [27], on the other hand, in their hard example mining paper, rank the images in order of difficulty at the time of training to dynamically generate a mini-curriculum at each iteration.

Kumar et al. [28], in their work on self-paced learning, instead monitor image difficulty as either the negative log-likelihood for expectation-maximisation or the upper bound on risk for latent structural support vector machines. Jiang et al. [29] incorporate both self-paced learning and curriculum learning into a single framework. That is, the curriculum is pre-defined by some expert, but takes into account the feedback from the model (the learner) when selecting which images to propose to the network during training.

Finally, Matiisen et al. [30] introduce the concept of *mastery* into the curriculum learning framework. In its simplest form, *mastery* is reaching a performance threshold for the model, identified by prior expert knowledge. The model is presented with images from a global dataset, but with a higher probability of sampling images from the current curriculum subset. As the model masters this subset, the probability of sampling these images decreases, while the probability of sampling the next curriculum subset increases.

If we consider these studies concurrently, it is evident that curriculum learning and hard example mining both greatly benefit the deep learning optimisation process, and the combination of the two does so even more. We therefore uniquely propose to employ such methods within NAS, specifically, levying *mastery* from [30] in tandem with our own hard example mining approach reminiscent of the *'instructor—student collaborative'* learning paradigm [29].

The work of Cazenavette et al. [31] builds upon well-explored dataset distillation

techniques [32]. By optimising the $l2$ loss of the *parameters* of a network trained on only 50 images per class, compared to optimal network parameters (i.e. parameters induced by training with 5000 images per class), they are able to achieve reasonable performance (71.5% on CIFAR-10 [33]). On this basis, we can deduce that training on a fraction of images yields a promising research direction, to which our method pertains without such loss in performance.

3. Proposed Approach

In this section, we detail the process by which our proposed DDS-NAS training strategy dynamically samples the dataset in an online fashion within the NAS cycle (Figure 1). DDS-NAS is subsequently deployed across three leading contemporary NAS frameworks (DARTS [6], P-DARTS [7], and TAS [8]).

Firstly, we define some key terms to which we will refer in our subsequent discussion:

- *hard* or *hard*-ness: a given example within the dataset at the current NAS training cycle iteration is defined as being *hard* if the output of the current model correlates poorly with the ground truth label for this example and hence contributes significantly to the current loss value for the model (i.e. it is either misclassified or classified with a low confidence score in the context of image classification).
- *easy*: the converse of *hard*, where for a given example the output of the current model correlates strongly with the ground truth label for this example and hence contributes less significantly to the current loss value for the model (i.e. correctly classified with a high confidence score in the context of image classification).
- *mastery*: a measure of when a given *a priori* performance threshold is reached on the current data subset such that the number of *easy* examples in the dataset is high with regard to the current model.

3.1. Curriculum Learning Within NAS

To formulate an unbiased subset of the global dataset, we use the hard example mining process detailed in Section 3.2. At every training iteration within the NAS search phase, we present such a subset to the NAS model. Following the success of [30], we in fact present the same subset until it has been *mastered*, according to some *a priori* mastery threshold (see Section 4.1). Only when the NAS model masters a subset do we sample a new set of examples from the global dataset. If the *mastery* threshold is very low, this subset of data will change often. If the mastery threshold is very high, a given subset is presented to the NAS model for several successive iterations, and a smaller portion of the global dataset is sampled throughout the entire training process. Akin to the restriction with P-DARTS [7] whereby only network parameters (i.e. weights) are updated and not architectural parameters within the first 10 training epochs, we similarly restrict DDS-NAS from resampling the dataset in this way for the first 10 epochs of NAS training.

3.2. Dynamic Data Selection

In order to both minimise the data subset used in each NAS iteration without performance degradation and facilitate efficient inter-iteration dataset resampling, we require a low-overhead process by which we can dynamically select new data examples.

From the initial NAS training iteration, and the immediate subsequent iterations thereafter, model performance can be considered near-random.[1] As such, we necessarily depend upon a resampling process independent of model performance, and hence propose the use of dataset example similarity as an alternative measure to relative *hard*-ness between samples. The intuition is that a model will perform poorly on examples with greater dissimilarity to those upon which it has already been trained. By using a resampling process independent of model performance, we do not need to compute the

[1]Noting that Deep Image Priors [34] indicate that untrained model performance in fact correlates to architecture design.

forward-pass of the model on all image samples in the entire dataset per hard example mining iteration, an approach commonplace among existing hard example mining approaches. This significantly reduces the computational complexity of DDS-NAS.

Given the need to perform efficient *one-to-many* feature distance comparisons via an online approach, we construct a series of efficient furthest-neighbour *kd*-tree structures from the chosen N-dimensional feature representations of each example in our global dataset. In order to maintain a balanced data subset in the presence of dynamic reselection, we construct one such *kd*-tree structure per class label in the dataset, resulting in m trees for m dataset classes. In this way we can facilitate *like-for-like* class-aware resampling and hence maintain dataset balance throughout the NAS training cycle. This strategy resembles undersampling, which has been shown to be effective for dealing with biased datasets [35], and is a significant advantage of our approach.

To enable efficient look-up within our *kd*-tree structure, we require a sufficiently low dimension N of our feature representation such that the approximate furthest neighbour algorithm does not collapse [36]. As the dimensionality of image data is high (i.e. $N = 28 \times 28$ in case of MNIST [37], and larger for more complex datasets), we instead propose using an additional autoencoder architecture to construct an image similarity embedding with a much lower dimension ($N = 8$ for easier MNIST and Fashion-MNIST datasets [38], $N = 32$ for CIFAR-10).

In general, we find that contemporary state-of-the-art autoencoder architectures [39] employ skip-connections between the encoder and decoder sub-networks to facilitate improved image reconstruction. However, in this instance, such skip connections are detrimental to the performance of the encoder network in terms of constructing an encoding at the bottleneck of the encoder-decoder architecture (our embedding) that maximally captures the highest level of feature detail within itself. On this basis, we employ the proven autoencoder architecture from GANomaly [40] as it is one of the most successful encoder-decoder architectures employed for encoded image discrimi-

nation, predating the wider move to the use of skip-connections in the field [39].

We require that the use of this encoder architecture results in a compact feature embedding that retains the property of spatial similarity such that similar images have similar embeddings within the latent space and vice versa. This property must not come at the expense of image reconstructability. Otherwise, we cannot be confident that a given embedding represents a given image. In other words, there would be no correlation between embedding space dissimilarity and image space dissimilarity. Given reconstructability without similar images clustering within the embedding space, we cannot guarantee that the correlation is strong.

To enforce these properties, we discovered that contractive loss [41] is sufficient for easier datasets, while harder datasets require a combined triplet margin ranking loss with MSE reconstruction loss, weighted via Kendall Loss [42]. Subsequently, we can thus order images by their dissimilarity within our furthest neighbour *kd*-tree structures. See Table 1 for a lightweight autoencoder training configuration sufficient for each dataset.

During a given NAS training iteration, we measure the *hard*-ness of each example image in the current data subset based on cross-entropy loss, following our earlier definition of *hard* and *easy* examples. To subsequently update our data subset in a dynamic manner, we first retain the images that are *hard* when averaged across the most recent epochs, according to some *a priori hard*-ness threshold (see Section 4.1). Secondly, by selecting the *kd*-tree from our set that is associated to the class label of each image in the current data subset below the *hard*-ness threshold (i.e. the *easy* images), we can then identify the most dissimilar image of the same class in the global training set in $O(log(n))$ time. We can then use this to replace the *easy* image within the data subset. This dynamically updated training data subset will then be used for the next NAS training iteration. A detailed example can be found in Appendix A.

Our overall pipeline is presented as follows: once the previous data subset has been

mastered by iterative NAS training, we dynamically formulate a new balanced subset of the global training dataset based on (a) the retention of images that are considered *hard*, and (b) the replacement of images that are considered *easy* with dissimilar images of the same class to retain dataset balance (Fig. 1). Pseudo-code to illustrate the overall pipeline with time complexity analysis can be found in Appendix B, which highlights the potential search speed efficiency that DDS-NAS affords.

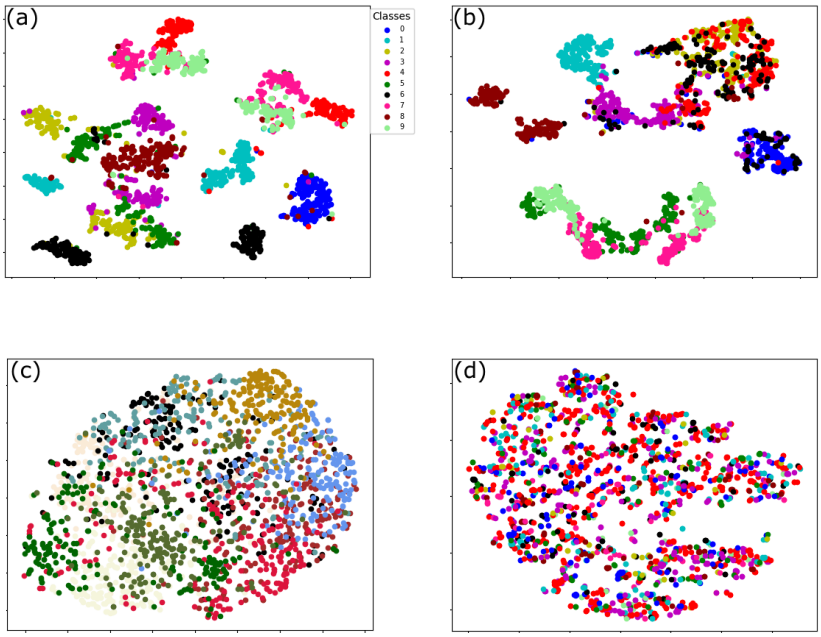

Figure 2: TSNE visualisation of clustering of autoencoded image feature representation within latent space. Our autoencoder preserves the property that similar images have similar encodings for MNIST (a), Fashion-MNIST (b), and CIFAR-10 (c). However, our compact embedding is unsuitable for fine-grained image classification such as FGVC-Aircraft (d), which is a known limitation of autoencoders.

4. Experimental Setup

We detail our experimental setup for DDS-NAS deployment across the Differentiable Architecture Search (DARTS), Progressive DARTS (P-DARTS) and Network Pruning via Transformable Architecture Search (TAS) NAS frameworks. This setup is used to demonstrate the performance of our proposed approach with several image classification datasets.

Dataset	Autoencoder Architecture	Bottleneck Embedding Dimension N	Loss Function
MNIST	GANomaly	8	Contractive Loss
Fashion-MNIST	GANomaly	8	Contractive loss
CIFAR-10	GANomaly	32	Triplet loss and MSE loss with Kendall Loss Weighting

Table 1: Suggested autoencoder training configuration parameters for each dataset to yield a sufficiently lightweight architecture that can generate low-dimensionality embeddings.

4.1. NAS Configuration

Unless otherwise stated, all employed NAS frameworks adopt the same common configuration using Adam optimisation [43] with initial learning rate $lr = 3e^{-4}$, weight decay $wd = 1e^{-3}$, and momentums $\beta_1 = 0.5$ and $\beta_2 = 0.999$ (P-DARTS uses $lr = 6e^{-4}, wd = 1e^{-3}$, TAS uses $lr = 1e^{-4}$). For weight optimisation for the NAS-derived architectures themselves, we use an SGD optimiser with $wd = 3e^{-4}$, and momentum $\beta = 0.9$ (P-DARTS uses $wd = 5e^{-4}$). Additionally, for DARTS we employ a Cyclic Learning Rate Scheduler with base $lr = 0.001$, max $lr = 0.01$, and step size up = step size down = 10. We set $lr = 0.01$ when the previous dynamically selected data subset is mastered, and an updated data subset is introduced. Therefore, the updated data subset is learned quickly and is then 'fine-tuned' as with the previous subset. There is precedence for such an approach in SGDR [44], in which the learning rate is periodically reset to a higher value before the learning rate decay is reapplied. P-DARTS and TAS both adopt Cosine Annealing Learning Rate Scheduler with $lr = 2.5e^{-2}$ and $lr = 0.1e^{-2}$ respectively. We select the ResNet-110 architecture for TAS kd-teacher training. The models are implemented using PyTorch [45] (v1.6.0, Python 3.6.9).

Performance of DDS-NAS deployed across each NAS framework is presented in terms of both Top-1 accuracy and parameter count (complexity) of the optimal NAS-generated architecture, together with the computational effort of the NAS search phase (in GPU days) across all three datasets.

Experimentation indicates that our NAS framework is generally insensitive to *a priori* thresholds that do not need to be exhaustively searched. A subset-size of 100 is sufficient for the easier MNIST [37] and Fashion-MNIST [38] tasks, and 1000 for CIFAR-10 [33]. Adopting a high hardness threshold (*hard*-ness threshold > 0.8) across all datasets and all NAS strategies enables the searched network architecture to formulate a thorough feature representation for image classification. The best network architectures are discovered with a mastery threshold ≈ 0.5. P-DARTS and TAS learn deep representations for images slower than DARTS. This can be attributed to the additional tasks done alongside reducing classification loss, wherein P-DARTS progressively restricts the search space while increasing architecture depth, and TAS minimises for network architecture complexity. Conversely, DARTS can afford a lower mastery threshold ($\approx$ 0.15) for the easier MNIST and Fashion-MNIST tasks, but the performance gain is marginal. All presented results use the same hardness (0.85) and mastery (0.5) thresholds to ensure fairness.

4.2. Hard Example Mining

The GANomaly autoencoder [40] used to encode the images into their latent space representation is trained with Contractive Loss [41] for 30 epochs, with $bs = 8$, and Adam optimiser with momentums $\beta_1 = 0.9$ and $\beta_2 = 0.999$, $wd = 0$, $lr = 1e^{-3}$. For the CIFAR-10 task, the autoencoder is instead trained with combined triplet margin loss [46] and MSE reconstruction loss, weighted under Kendall Loss [42].

5. Evaluation

Having validated the feature representation embedding that underpins our dynamic data selection via hard example mining (see Figure 2), we present out evaluation in terms of DDS-NAS comparison to contemporary state-of-the-art approaches, with supporting ablation studies.

Dataset	NAS Approach	Top-1 Accuracy (%) ↑ DARTS / P-DARTS / TAS	Params (M) ↓ DARTS / P-DARTS / TAS	Search Cost (GPU Days) ↓ DARTS / P-DARTS / TAS
MNIST	Original	99.75 / 99.26 / 99.27†	0.66 / 3.68 / 1.00	0.51 / 1.89 / 0.28
	DDS-NAS	99.78 / 99.17 / 99.30†	0.75 / 3.51 / 0.81	0.030 / 0.070 / 0.021
Fashion MNIST	Original	95.33 / 93.42 / 95.09†	3.27 / 4.04 / 0.94	0.63 / 1.98 / 0.27
	DDS-NAS	95.48 / 93.04 / 95.08†	3.44 / 4.23 / 0.83	0.030 / 0.078 / 0.031
CIFAR-10	Original	97.17 / 96.50 / 93.89	3.16 / 3.43 / 0.85	1.78 / 0.65 / 0.26
	DDS-NAS	96.57 / 95.07 / 93.12	3.72 / 4.13 / 1.06	0.36 / 0.095 / 0.040
	Shapley-NAS [47]	96.96	3.60	0.36
	SNAS [48]	97.15	2.85	1.83
	DenseNet [49]	94.23	7.0	–

Table 2: Accuracy, memory footprint, and (search-phase) training cost of final generated model from DDS-NAS deployed upon DARTS, P-DARTS, and TAS, compared to their original implementations and others. † indicates results without *kd*-teacher training owing to the lack of available teacher models for MNIST and Fashion-MNIST datasets.

5.1. Neural Architecture Search

Table 2 presents the performance obtained by the final model generated by DDS-NAS with respect to each dataset under consideration. Across all cases, the performance of our generated models is competitive with the state of the art, with minimal to no impact on generated model size. Moreover, across all cases, we substantially lower the computational efforts required for NAS (0.07 GPU days compared to 1.89 in the case of P-DARTS for MNIST, 27 times quicker)[2]. Since we can determine a replacement image for our dynamic subset in average case $O(log(n))$ time, we are able to reduce the search phase training cost by one order of magnitude over state-of-the-art results.

Without loss in performance, our hard example mining method yields discriminative architectures that can be transferred to CIFAR-100 [33] and ImageNet [50] (Table 3). However, reproducibility presents a particularly significant problem within the NAS domain [51] and TAS ImageNet performance is considerably lower than the literature

[2]Still an order of magnitude faster even after factoring in the time taken to train the autoencoder

CIFAR-100		
NAS Approach	Top-1 Accuracy (%) ↑ DARTS / P-DARTS / TAS	Params (M) ↓ DARTS / P-DARTS / TAS
Original	81.33 / 80.58 / 71.72	2.75 / 3.49 / 1.15
DDS-NAS	82.64 / 75.45 / 70.47	3.80 / 4.24 / 1.15
Shapley-NAS	83.42	3.66
DenseNet	76.21	7.0
ImageNet		
Original	73.30 / 75.72 / -	4.51 / 4.94 / -
DDS-NAS	76.26 / 75.63 / -	6.14 / 5.68 / -
Shapley-NAS	75.52	5.14
DenseNet	74.98	7.0

Table 3: Accuracy and memory footprint of CIFAR-10 searched models transferred to CIFAR-100 and ImageNet

reports. DDS-NAS-TAS performance is omitted for fairness. Whilst our technique is demonstrated upon commonplace NAS approaches (DARTS, P-DARTS, TAS) it could equally be deployed on top of more recent advancements [47, 15, 16], further minimizing any difference in performance.

5.2. Ablation Studies

To validate our proposed approach, we compare the performance of DDS-NAS to selected NAS frameworks, both: (a) without *dynamic data selection* in order to ablate the contribution of our combined hard example mining and curriculum learning strategy; and (b) with an untrained autoencoder to ablate the contribution of the image-dissimilarity based hard example mining strategy.

Dataset	NAS Approach	Top-1 Accuracy (Search Phase) (%) ↑ DARTS / P-DARTS / TAS	Top-1 Accuracy (Final) (%) ↑ DARTS / P-DARTS / TAS	Params (M) ↓ DARTS / P-DARTS / TAS
MNIST	DDS-NAS	94.00 / 78.89 / 44.89	99.78 / 99.17 / 99.30	0.75 / 3.51 / 0.81
	Original framework with dataset size 100	78.28 / 70.81 / 39.24	94.43 / 98.69 / 99.18	0.70 / 4.54 / 0.53
	DDS-NAS with untrained autoencoder	92.28 / 78.76 / 51.96	95.28 / 98.78 / 99.21	0.75 / 3.11 / 1.05
Fashion-MNIST	DDS-NAS	72.92 / 65.71 / 32.71	95.48 / 93.04 / 95.08	3.44 / 4.23 / 0.83
	Original framework with dataset size 100	56.27 / 58.16 / 35.66	91.69 / 90.03 / 94.61	3.47 / 4.69 / 0.46
	DDS-NAS with untrained autoencoder	69.49 / 64.47 / 39.12	92.04 / 91.52 / 94.87	3.48 / 3.92 / 0.93
CIFAR-10	DDS-NAS	56.00 / 22.14 / 29.88	96.57 / 95.07 / 93.12	3.72 / 4.13 / 1.06
	Original framework with dataset size 1000	51.02 / 41.70 / 23.81	88.58 / 85.74 / 90.83	3.55 / 4.04 / 0.32
	DDS-NAS with untrained autoencoder	51.10 / 46.59 / 28.21	88.90 / 88.96 / 91.72	3.64 / 4.25 / 0.83

Table 4: Ablation studies: accuracy and memory footprint of models generated by DDS-NAS, models generated by the original framework with limited data (equivalent to removing hard example mining and curriculum learning), and models generated by DDS-NAS with an untrained autoencoder (equivalent to removing hard example mining).

5.2.1. Without Dynamic Data Selection

For each dataset, we employ all three original implementations (DARTS [6], P-DARTS [7], TAS [8]), but with a subset of the data at each training iteration. *This is equivalent to omitting both hard example mining and curriculum learning*. We use the same volume of data as adopted by DDS-NAS: 100 randomly selected images for MNIST and Fashion-MNIST, and 1000 for CIFAR-10. Subsequently, we can determine the impact of our curriculum learning and hard example mining pipeline. Comparing the first and second row of the results for each dataset presented in Table 4, it is evident that DDS-NAS achieves substantially improved accuracy while yielding fractionally larger architectures in some cases. This behaviour is exhibited in MNIST, where the original DARTS framework achieves only 78.28% accuracy after the search phase, and 94.43% accuracy after fine-tuning the stacked searched cell (compared to 94.00% and 99.78% respectively for DDS-NAS-DARTS). This performance difference is further

highlighted with both the other datasets and other frameworks. The final performance of the original P-DARTS implementation falls behind DDS-NAS across all datasets (85.74% compared to 95.07% for CIFAR-10, for instance). Interestingly, with hard example mining and curriculum learning omitted in this manner, TAS generates smaller models (0.32M compared to 1.06M for CIFAR-10), but often at the expense of accuracy.

5.2.2. Untrained Autoencoder

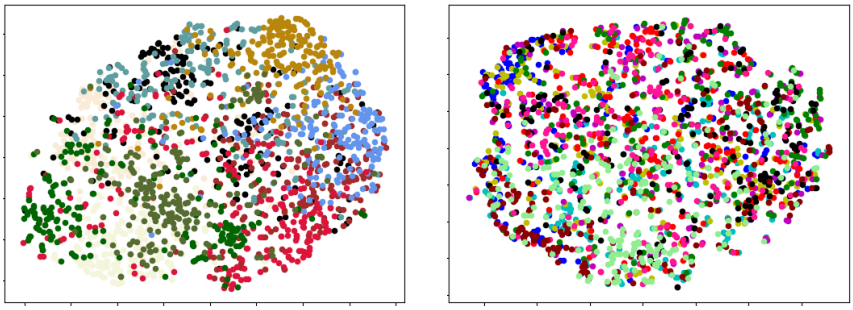

Figure 3: TSNE visualisation of the clustering of autoencoded CIFAR-10 image feature representation within the latent space. Training with triplet margin loss with Kendall loss achieves good clustering (left). Training with contractive loss achieves poor clustering (right).

We ablate the autoencoder-derived feature embedding within our hard example mining method by replacing the DDS-NAS autoencoder with one that is untrained, and thus unable to determine the most dissimilar images from a given training data subset. *This can be considered as a process equivalent to curriculum learning without hard example mining*, as the images are effectively randomly sampled. This time, we compare the first and third row for each dataset in Table 4. Evidently, the models generated by DDS-NAS with an untrained autoencoder are significantly worse (for instance 92.04% compared to 95.48% upon Fashion-MNIST by DDS-NAS-DARTS). On this basis, we can therefore conclude that DDS-NAS necessarily requires a suitable hard example mining approach, for which our image similarity strategy is sufficient.

Furthermore, an autoencoder that achieves good reconstruction but a mediocre

Reconstruction	✓	✓	×	×
Clustering	✓	×	✓	×
Top-1 Accuracy (%) ↑	96.57	95.29	94.94	88.90

Table 5: Accuracy of DDS-NAS-DARTS employing autoencoders with different capabilities on CIFAR-10.

clustering of embedded features is inadequate for DDS-NAS (Fig. 3, Table 5). Bad clustering and thus ineffective hard example mining yields inferior classification accuracy (95.29%) compared to hard example mining with good clustering (96.57%). Similarly, sufficient clustering but poor reconstruction is detrimental to DDS-NAS (94.94%). Lack of both properties yields significantly worse performance (88.90%), wherein there is no correlation between embedding space dissimilarity and image space dissimilarity at all.

By comparing row two (neither hard example mining nor curriculum learning) and row three (curriculum learning but not hard example mining) for each dataset in Table 4, it is clear that our curriculum learning methodology is *somewhat* effective even without incorporating hard example mining. DDS-NAS performance with an untrained autoencoder exceeds that of the original framework with limited data in all cases (88.58% compared to 88.90% for CIFAR-10 with DARTS, 90.03% compared to 91.52% for Fashion-MNIST with P-DARTS).

6. Limitations

The modularity of the proposed DDS-NAS framework provides a significant advantage over existing NAS methods, and allows it to be adopted alongside multiple NAS frameworks. Selecting an off-the-shelf autoencoder or training one from scratch is a reasonable approach provided it can generate a low-dimensionality embedding space that offers reasonable reconstruction and clustering capabilities (see Section 5.2). For fine-grained classification tasks however, this is a challenge (see Figure 2) and remains an open area of research.

In addition, the current DDS-NAS approach requires one *kd*-tree per class so that we can perform class-aware dynamic dataset updates. While this offers reasonable robustness towards biased datasets, long-tailed distributions in datasets may present additional challenges, where there are not enough samples for a given class. We might expect training samples to be memorized in this situation, yielding noisy architecture weight-update steps. One simple solution to resolve this might be to combine samples from classes with few samples into a single *kd*-tree but this is a direction for future research.

7. Conclusion

To conclude, we propose DDS-NAS: a novel NAS framework capable of reducing the time required for the NAS search phase by one order of magnitude. By employing image similarity as a basis for hard example mining, and thus (online) dynamic data sub-selection, DDS-NAS yields models that remain competitive towards accuracy and memory costs upon common image datasets. Further, we demonstrate that DDS-NAS can be deployable upon several NAS approaches and architecture spaces, and is similarly extendable to all existing evolutionary, reinforcement-learning, or gradient-based NAS approaches. DDS-NAS can even incorporate NAS search phase techniques that are deployed alongside rather than in place of existing NAS approaches [15].

Following the success of our approach, we posit that only a fraction of commonly used image datasets contribute to learning. As such, additional analysis of these datasets is necessary. Moreover, a more comprehensive investigation into the autoencoder architecture employed within our hard example mining method may yield better results (and thus further reduce the volume of contributing images within a dataset). Specifically, we require an autoencoder that can generate similar embeddings for similar images even within the fine-grain classification domain. Alternative autoencoder losses, or measures of image similarity such as hashing, may yield similar improvements. Nevertheless, even a glimpse of image similarity as a metric within hard example mining

has proven extremely effective. We thus introduce several new avenues for improvement, particularly alongside NAS frameworks, and demonstrate that network architecture topology design should not necessarily be the sole consideration for future NAS solutions.

8. Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the authors used GPT-4o mini in order to generate the LaTeX code for Algorithm 1 with placeholder steps and time complexity. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

9. Acknowledgements

This work was supported by Durham University, United Kingdom, the European Regional Development Fund Intensive Industrial Innovation Grant No. 25R17P01847 and Petards Joyce-Loebl Ltd .

[1] A. Younesi, M. Ansari, M. Fazli, A. Ejlali, M. Shafique, J. Henkel, A comprehensive survey of convolutions in deep learning: Applications, challenges, and future trends, IEEE Access 12 (2024) 41180–41218.

[2] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vol. 1, 2005, pp. 886–893 vol. 1.

[3] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, Vol. 60, 2009, p. 6.

[4] G. Hacohen, D. Weinshall, On the power of curriculum learning in training deep networks, in: International Conference on Machine Learning, 2019.

[5] M. Poyser, T. P. Breckon, Neural architecture search: A contemporary literature review for computer vision applications, Pattern Recognition 147 (2024) 110052.

[6] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, in: 7th International Conference on Learning Representations, 2019.

[7] X. Chen, L. Xie, J. Wu, Q. Tian, Progressive DARTS: Bridging the Optimization Gap for NAS in the Wild, arXiv e-prints (2019).

[8] X. Dong, Y. Yang, Network pruning via transformable architecture search, in: Advances in Neural Information Processing Systems, 2019, pp. 759–770.

[9] B. Zoph, Q. V. Le, Neural architecture search with reinforcement learning, in: 5th International Conference on Learning Representations, 2017.

[10] B. Baker, O. Gupta, N. Naik, R. Raskar, Designing neural network architectures using reinforcement learning, in: 5th International Conference on Learning Representations, 2017.

[11] A. Cassimon, S. Mercelis, K. Mets, Scalable reinforcement learning-based neural architecture search, Neural Comput. Appl. 37 (1) (2025) 231–261.

[12] T. Gong, Y. Ma, Y. Xu, C. Song, Efficient evolutionary neural architecture search based on hybrid search space, Int. J. Mach. Learn. Cybern. 15 (8) (2024) 3313–3326.

[13] X. Liang, P. Fu, Q. Guo, K. Zheng, Y. Qian, Dc-nas: Divide-and-conquer neural architecture search for multi-modal classification, Proceedings of the AAAI Conference on Artificial Intelligence 38 (12) (2024) 13754–13762.

[14] H. Pham, M. Guan, B. Zoph, Q. Le, J. Dean, Efficient neural architecture search via parameters sharing, in: Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, 2018, pp. 4095–4104.

[15] Q. Guo, X.-J. Wu, J. Kittler, Z. Feng, Differentiable neural architecture learning for efficient neural networks, Pattern Recognition 126 (2022) 108448.

[16] Y. Hu, X. Wang, L. Li, Q. Gu, Improving one-shot nas with shrinking-and-expanding supernet, Pattern Recognition 118 (2021) 108025.

[17] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in: The Conference on Neural Information Processing Systems Workshop, 2014.

[18] Y. Guo, Y. Chen, Y. Zheng, P. Zhao, J. Chen, J. Huang, M. Tan, Breaking the curse of space explosion: Towards efficient nas with curriculum search, in: Proceedings of the 37th International Conference on Machine Learning, 2020.

[19] Z. Zhou, X. Ning, Y. Cai, J. Han, Y. Deng, Y. Dong, H. Yang, Y. Wang, CLOSE: curriculum learning on the sharing extent towards better one-shot NAS, in: Computer Vision - 17th European Conference, Vol. 13680 of Lecture Notes in Computer Science, 2022, pp. 578–594.

[20] J. Peng, J. Zhang, C. Li, G. Wang, X. Liang, L. Lin, Pi-nas: Improving neural architecture search by reducing supernet training consistency shift, arXiv e-prints (2021).

[21] J.-h. Shim, K. Kong, S.-J. Kang, Core-set Sampling for Efficient Neural Architecture Search, arXiv e-prints (Jul. 2021).

[22] V. Prasad C, C. White, P. Jain, S. Nayak, G. Ramakrishnan, Speeding up NAS with Adaptive Subset Selection, arXiv e-prints (Nov. 2022).

[23] R. Wang, M. Cheng, X. Chen, X. Tang, C. Hsieh, Rethinking architecture selection in differentiable NAS, in: 9th International Conference on Learning Representations, OpenReview.net, 2021.

[24] K. Killamsetty, D. Sivasubramanian, G. Ramakrishnan, R. K. Iyer, GLISTER: generalization based data subset selection for efficient and robust learning, in: Thirty-Fifth AAAI Conference on Artificial Intelligence, 2021, pp. 8110–8118.

[25] A. Graves, M. G. Bellemare, J. Menick, R. Munos, K. Kavukcuoglu, Automated curriculum learning for neural networks, arXiv e-prints (2017).

[26] D. Weinshall, G. Cohen, D. Amir, Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks, arXiv e-prints (Feb. 2018).

[27] A. Shrivastava, A. Gupta, R. Girshick, Training region-based object detectors with online hard example mining, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2016, pp. 761–769.

[28] M. Kumar, B. Packer, D. Koller, Self-paced learning for latent variable models, in: Advances in Neural Information Processing Systems, Vol. 23, Curran Associates, Inc., 2010, pp. 1189–1197.

[29] L. Jiang, D. Meng, Q. Zhao, S. Shan, A. G. Hauptmann, Self-paced curriculum learning, in: Proceedings of the 29th AAAI Conference on Artificial Intelligence, AAAI'15, 2015, p. 26942700.

[30] T. Matiisen, A. Oliver, T. Cohen, J. Schulman, Teacherstudent curriculum learning, IEEE Transactions on Neural Networks and Learning Systems 31 (9) (2020) 3732–3740.

[31] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, J.-Y. Zhu, Dataset distillation by matching training trajectories, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10718–10727.

[32] T. Wang, J. Zhu, A. Torralba, A. A. Efros, Dataset distillation, arXiv e-prints (2018).

[33] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech. rep. (2009).

[34] D. Ulyanov, A. Vedaldi, V. Lempitsky, Deep Image Prior, arXiv e-prints (Nov. 2017).

[35] J. M. Johnson, T. M. Khoshgoftaar, Survey on deep learning with class imbalance, J. Big Data 6 (2019) 27.

[36] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is "nearest neighbor" meaningful?, in: Database Theory, 1999, pp. 217–235.

[37] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[38] H. Xiao, K. Rasul, R. Vollgraf, Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, arXiv e-prints (Aug. 2017).

[39] S. Akçay, A. Atapour-Abarghouei, T. P. Breckon, Skip-ganomaly: Skip connected and adversarially trained encoder-decoder anomaly detection, in: International Joint Conference on Neural Networks, IEEE, 2019, pp. 1–8.

[40] S. Akcay, A. Atapour-Abarghouei, T. P. Breckon, Ganomaly: Semi-supervised anomaly detection via adversarial training, in: Asian Conference on Computer Vision, Springer, 2018, pp. 622–637.

[41] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: Explicit invariance during feature extraction, in: Proceedings of the 28th International Conference on Machine Learning, Omnipress, 2011, pp. 833–840.

[42] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 7482–7491.

[43] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, 2015.

[44] I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with warm restarts, in: 5th International Conference on Learning Representations, 2017.

[45] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).

[46] F. Schroff, D. Kalenichenko, J. Philbin, Facenet: A unified embedding for face recognition and clustering, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, 2015, pp. 815–823.

[47] H. Xiao, Z. Wang, Z. Zhu, J. Zhou, J. Lu, Shapley-nas: Discovering operation contribution for neural architecture search, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 11892–11901.

[48] S. Xie, H. Zheng, C. Liu, L. Lin, SNAS: stochastic neural architecture search, in: 7th International Conference on Learning Representations, 2019.

[49] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2017, pp. 2261–2269.

[50] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, Imagenet large scale visual recognition challenge, International Journal of Computer Vision 115 (3) (2015) 211252.

[51] D. Towers, M. Forshaw, A. Atapour-Abarghouei, A. S. McGough, Long-term reproducibility for neural architecture search, arXiv e-prints (2022).

A. Hard Example Mining Search Process

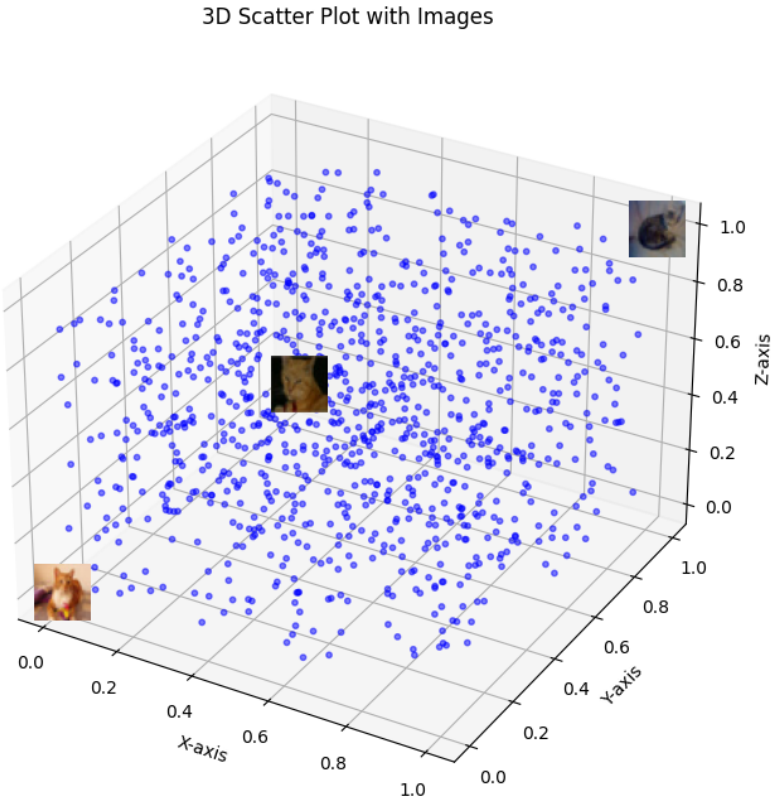

Figure 4: An example embedding space for the cat class.

In the first iterations of training, all samples are considered equally difficult and

images from each class are randomly sampled. The embedding space for a cat class might resemble Figure 4. The ginger cat in daylight (coordinates = $0, 0, 0$) is sampled first, and after a few iterations, the current subset of data is mastered. Performance on the ginger cat sample is good and it is considered *easy*. The most dissimilar image in a given dimension to the current cat sample is the ginger cat in darkness (coordinates = $0, 1, 0$). The model is trained for a few more epochs and eventually the data subset is mastered again, and performance on the cat in darkness is good. This time the most dissimilar image in a new dimension is the black cat (coordinates = $1, 1, 1$), which is now added to the data subset and *replaces* the ginger cat. In this way, different features of the cat embedding space are explored during training. In practice, dimensions are often more abstract based on the learnt embedding space of the autoencoder.

Traditional hard example mining strategies compute hardness for all samples in the entire dataset and re-weight their impact accordingly. By using image dissimilarity as a measure for image hardness, we do not need to evaluate current model performance on images outside of a data subset. We compute similarity embeddings *once* prior to any NAS training iterations and use the embeddings to calculate maximally dissimilar image samples from the current data subset for the next training iteration. We are therefore able to levy the benefits of curriculum learning and hard example mining while *reducing* overhead. This has the benefit of hard example mining and curriculum learning over existing core-set selection NAS approaches [21, 22], which use neither. In this way, DDS-NAS also benefits from data subset selection compared to curriculum strategies [18, 19].

B. Overall Pipeline Pseudo-code with Time Complexity Analysis

Note: In this algorithm, n' represents a subset of size much smaller than n, i.e., $n' \ll n$.

Algorithm 1 Pseudo-code illustrating the overall DDS-NAS pipeline

	Step	**Big-O Time Complexity**
1:	Initialize data subset:	$O(n')$
	Randomly select images from each class	
	to form a balanced subset of size n'	
2:	Construct m class-specific kd-trees	$O(mn \log n)$
3:	Compute feature embeddings with autoencoder	
4:	**for** each NAS iteration **do**	$O(k)$
5:	**if** current subset is mastered **then**	$O(1)$
6:	Retain hard examples based on threshold	$O(n')$
7:	Identify dissimilar images via kd-tree look-up	$O(\log n')$
8:	Replace easy images with dissimilar ones	$O(1)$
9:	Update dataset for next NAS iteration	$O(n')$
10:	**end if**	
11:	Perform a normal NAS training iteration	
12:	**if** network has converged **then**	$O(1)$
13:	**Stop**	$O(1)$
14:	**end if**	
15:	**end for**	

Note: As long as k is sufficiently large, this algorithm significantly reduces convergence time. In modern deep learning, k is typically large enough to ensure this.