Automated neural network generation for industrial datasets, application to laser powder bed fusion

Salmani Pour Avval, Sasan; Abadie, Titouan; Yaghoubi, Vahid

DOI
10.1007/s10845-025-02707-0

Publication date
2025

Document Version
Final published version

Published in
Journal of Intelligent Manufacturing

Journal of Intelligent Manufacturing
https://doi.org/10.1007/s10845-025-02707-0

Automated neural network generation for industrial datasets, application to laser powder bed fusion

Sasan Salmani Pour Avval[1] · **Titouan Abadie**[2] · **Vahid Yaghoubi**[1]

Received: 20 May 2025 / Accepted: 2 October 2025

Abstract New manufacturing techniques like 3D printing are under development, and they need monitoring methods to ensure the quality of the manufactured parts. Artificial Intelligence has outperformed traditional methods in the monitoring process and has shown high potential in recent years. New approaches in Artificial Intelligence, particularly Neural Architecture Search (NAS), have unlocked the potential for automated design of high-performance and resource-efficient deep learning models. In this work, we propose a training-based, low-fidelity NAS framework to systematically discover optimal architectures for regression tasks. Leveraging 8,610 candidate topologies, we trained models on only 0.1% of the data for 10 epochs, enabling faster execution and selection of the architecture using low-fidelity information. The dataset belongs to Laser Powder Bed Fusion (LPBF), which is a manufacturing method that is still not well mastered and requires many trials before obtaining a satisfactory result. To cope with this issue, we developed a NAS algorithm to design a lightweight AI model (an architecture with a low number of parameters) to predict the process parameters from video information to ensure having the same printing parameters in action. The ultimate goal is then to embed the AI model in a low-latency feedback control loop that enables on-the-fly supervision of the printing process. The final designed architecture is based on 3-dimensional convolutional neural networks. The final AI models are 3–30 times lighter than off-the-shelf ones, while maintaining almost the same accuracy. This shows the potential of our methods when dealing with regression tasks in an industrial case study.

✉ Sasan Salmani Pour Avval
S.SalmaniPourAvval@tudelft.nl

Titouan Abadie
titouan.abadie@etu.isae-ensma.fr

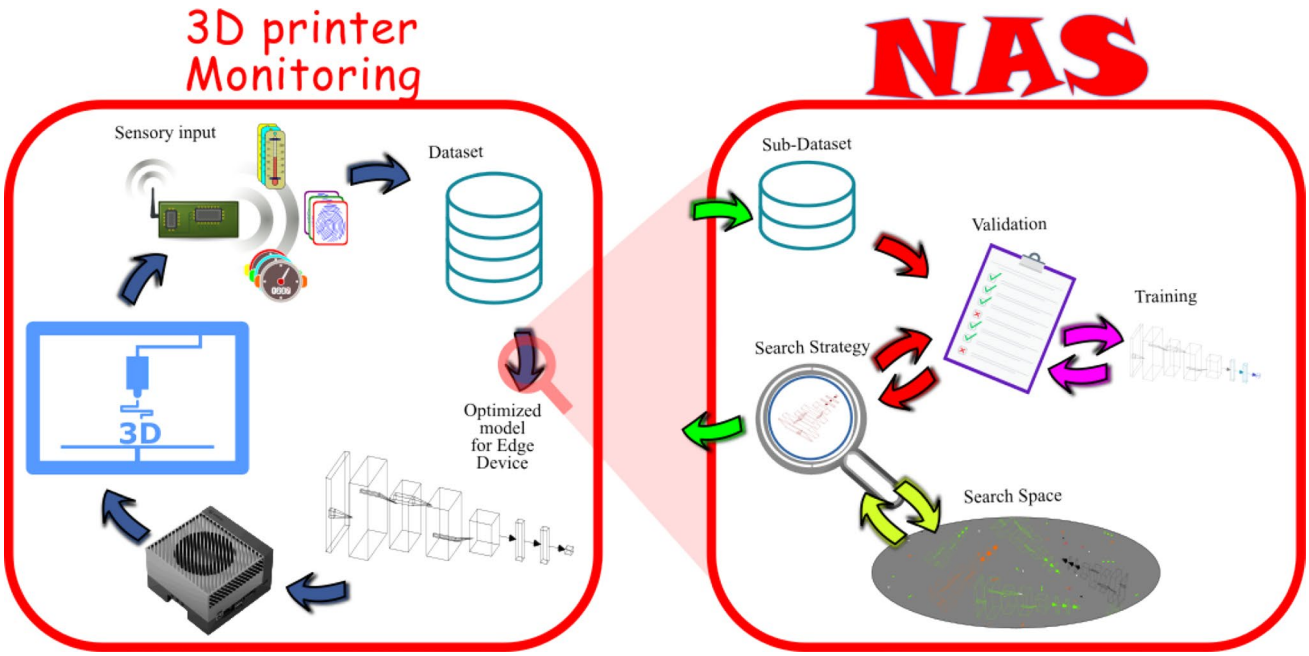
Vahid Yaghoubi
V.Yaghoubi@tudelft.nl

[1] Q-VAIbe Group, Aerospace Structures and Materials, Faculty of Aerospace Engineering, Delft University of Technology, Delft, The Netherlands

[2] École Nationale Supérieure de Mécanique et d'Aérotechnique, Poitiers, France

Published online: 31 October 2025

Graphical abstract



Keywords Neural architecture search · Computer vision · 3D convolutional neural networks · Laser powder bed fusion · Additive manufacturing

Introduction

Artificial Intelligence (AI) (Cheng et al., 2021)has evolved dramatically over the past decades from early rule-based, fully programmed systems to deep learning architectures, transforming the way machines perceive, learn, and interact with complex environments. What was once a niche academic pursuit has now become a core technology foundation for different applications across industries, ranging from natural language processing (Feuerriegel et al., 2025) and computer vision (Voulodimos et al., 2018) to autonomous vehicles (Janai et al., 2020) and predictive maintenance (Hu et al., 2025). This transformative progress has been fueled by advances in computational power (Ravikumar et al., 2022); the exponential growth of data; and innovative methodologies that automate the design and optimization of complex neural network architectures. The increasing complexity of architectures can make them unreliable due to an excessive number of unnecessary parameters, which may lead to overfitting and reduce their generalization capability, which is the problem in the previous Neural Architecture Search methods (Salmani Pour Avval et al., 2025). NAS has emerged as a pivotal tool, harnessing AI's capabilities to discover high-performing and resource-efficient neural networks automatically for devices that have limited power and restricted capacities, so-called edge devices.

Background knowledge on neural architecture search

Since the establishment of AI model and Machine Learning (ML) methods, the architecture has been designed by humans until the introduction of Automatic Machine Learning (AutoML) (Saxena & Verbeek, 2016), which was a way to enhance the AI models using automatic methods by computers and algorithms. Later, researchers realized that the architecture of the AI models is a massive and even infinite space which should be probed and explored without involving other ML parameters in the optimization process (Yaghoubi & Kumru, 2024). This branch of the autoML method is called Neural Architecture Search (NAS). NAS in general can be considered as an automated framework for designing Neural Network architectures that leverages the principles of systematic exploration and optimization. NAS operates on three fundamental pillars (Salmani Pour Avval et al., 2025): the Search Space (SSp), the Search Strategy (SSt), and the Validation Strategy (VSt). In simple words, NAS is an algorithm that finds, designs, or optimizes the topology of a neural network to suit the dataset.

The SSp defines the boundaries and constraints of potential architectures by specifying the types of layers, connectivity patterns, hyperparameters that can be combined, etc. This space includes every possible architecture that could

be generated, from the simplest to the most complex structures, and even sometimes novel architectures that might not be designed by humans.

The SSt is responsible for exploring and probing into the SSp in an efficient and effective manner. It involves an optimization process-using techniques such as reinforcement learning (Zoph & Le, 2016; Mills et al., 2021), evolutionary algorithms (Gottapu & Dagli, 2020; He et al., 2021), gradient-based methods (Wan et al., 2020; Chen et al., 2020; Liu et al., 2018), or even random search methods (Li & Talwalkar, 2020; Wen et al., 2020)-that iteratively refines the candidate architectures based on performance feedback. In essence, SSt explores the trade-offs between exploring new architectural possibilities and exploiting the most promising regions of the Search Space..

Complementing these is the VSt; it plays a crucial role in assessing the performance of each candidate in every iteration. VSt estimates or calculates the AI model's performance by evaluating it on a separate validation dataset, often employing techniques like full or partial training, weight sharing, or even zero-shot evaluation (Salmani Pour Avval et al., 2025).

Together, these three pillars-SSp, SSt, and VSt-form the skeleton of NAS, enabling it to generate architectures that not only achieve high accuracy automatically but can also be efficient, robust, and well-suited for deployment in real-world applications.

Literature review

Industrialized NAS (Benmeziane et al., 2021) represents the transition of NAS from a predominantly research-oriented endeavor into a mature, production-ready technology. In this phase, NAS methodologies have been refined and optimized to meet the stringent requirements of real-world applications by streamlining the search process through techniques such as weight sharing, differentiable search spaces, and multi-objective optimization (Salmani Pour Avval et al., 2025). These enhancements not only boost benchmark performance but also enable architectures to be tailored for deployment on specific hardware platforms with practical constraints. Moreover, industrialized NAS emphasizes robustness and reproducibility, ensuring that search pipelines produce consistent and reliable results across diverse datasets and tasks. This evolution allows companies to leverage automated design processes (Gupta & White, 2021) to accelerate the development of highly efficient machine learning models, significantly reducing the need for manual tuning and speeding up innovation across various sectors-from autonomous driving (Hao et al., 2019) to edge computing (Gupta & White, 2021).

Apart from the mentioned benefits of AI, there are some downsides to it as well. First of all, AI models are normally designed in a way that they are unsuitable for industrial applications because of massive memory and processing needs (Benmeziane et al., 2021). We try to develop a method that optimizes AI models to maintain a high performance across diverse datasets and real-world conditions while minimizing sensitivity to noise and variations. This characteristic is closely tied to generalization, ensuring the training performance without overfitting and achieving high performance. A key aspect is achieving these capabilities with fewer parameters, leading to efficient and resilient AI models. This way, NAS output can use minimal but effective parameterization, which enhances noise resistance and adaptability, making models more reliable for deployment in dynamic environments with varying data distributions. The second problem comes from the need for an expert to design the AI models. For example, introducing an architecture that can solve a problem in a specific industry needs to have an expert in the loop because architectures are far different from each other, and they are designed in a way to analyze a variety of data. Using NAS, we can solve this problem because the mentioned algorithms can understand the performance of the different architectures and functions in the models and guide the topology of the model to the point that fits the need of the problem.

Artificial intelligence in process monitoring

AI methods have enhanced the performance of the monitoring (Ahmadian et al., 2010) approaches for the industrial applications (Bowler et al., 2022; Hu et al., 2025). In addition to that, a new manufacturing method has been introduced as additive manufacturing (Serin et al., 2016). In this technology, the particles of materials are being added over time to each other to make useful parts. Several techniques have been developed in this manufacturing branch, and one of the most important metal manufacturing methods in additive manufacturing is Laser Powder Bed Fusion (LPBF) (Rothfelder et al., 2023). In this method, a layer of powder of a metal is applied on a pool of particles, and a laser beam melts some part of it, and these melted parts fuse to construct the part. This process repeats layer by layer, based on a CAD file, to build the parts. This process is sensitive to various factors like laser power or laser velocity, which can lead to defects such as tiny voids and pores (Dunbar, 2016). These defects can compromise the structural integrity and quality of the build (Kouprianoff et al., 2018). Therefore, the power of the laser beam and the distribution of the energy, which is related to the focal point of the magnifiers and the calibration of the mirrors, can change during the process because of the distortion (Yeung et al., 2020; Mussatto et

al., 2022) or aging of the actuator (Creac'h et al., 2015). As a result, the size of the laser spot and the distribution of the laser beam should be monitored to guarantee the quality of the manufacturing (Ross et al., 1996; Imran & Naeem, 2025). To improve the quality of the monitoring process, we need to implement fast AI models that are working on the edge. Meeting the mentioned requirement needs some special properties like low latency, high performance/ accuracy, etc. These kinds of architectures are hard to design by non-expert humans. Also, the designed AI models should have small memory usage due to the fact that in situ monitoring devices and edge computing machines are low in resources, and they have limited memory and energy. To solve the problem, we used NAS to enhance the architecture design in the monitoring process of the laser beam quality in LPBF 3D printing method.

Problem statement

In industrial applications, an AI model should be fast and lightweight (in terms of the number of parameters and operations). But, AI models that are designed by humans are mainly heavy and less accurate, which makes them not a suitable candidate for the in situ monitoring. Moreover, AI models designed by traditional NAS methods are often complex and computationally heavy, especially if they are designed using proxies for validation. The first reason is that, since the introduction of the NAS (Liu et al., 2018), the majority of the algorithms have been introduced for the classification problems, mainly for image classification such as CIFAR-10 (Krizhevsky et al., 2009) or ImageNet (Russakovsky et al., 2015). But for the regression problems, it is rare to find an algorithm that can optimize the algorithm automatically (Artin et al., 2021). Second, proxies are well defined for the complicated datasets, which are mentioned just before, and there is no possibility to find an architecture in the SSp reaching 100% accuracy. When we face a lack of dataset, like in industrial applications, the performance and reliability of the proxies go down, and actually they calculate the complexity of architectures (Mellor et al., 2021; Zhou et al., 2024).

Therefore, in this article, we introduced a training-based low-fidelity NAS algorithm for the regression problem, and then we applied it to an industrial dataset coming from additive manufacturing industries to optimize the AI models for edge monitoring of the process and even probed into some ideas to make the NAS algorithm quicker. We compared the introduced AI models with the one that is developed in the article of the dataset (Blanc et al., 2023).

Application

As mentioned, one of the common additive manufacturing methods is Laser Powder Bed Fusion (LPBF), which uses a laser beam to melt powder-shaped particles and fuse them. Therefore, we need to be sure that the laser beam is always in the same working conditions to guarantee high-quality parts. In this article, we want to monitor the working condition of the machine and detect the malfunction of the laser. In addition, we want to do the monitoring process in a real-time manner to be able to correct the printing properties. Therefore, having a light and quick AI model is our priority as well as designing high-performing architecture.

Current methods rely on trial-and-error approaches (Cao et al., 2021), involving costly post-production testing and multiple print iterations, resulting in increased scrap rates and higher production costs. A more effective solution would be an automated monitoring system integrated into a low-latency feedback control loop. This system observes the melt pool in real-time, predicting potential changes of the laser and enabling corrective actions to prevent or compensate for them, reducing the likelihood of defective prints.

Such a system could be implemented using computer vision models that analyze video data of the melt pool (Blanc et al., 2023). These architectures can predict the process's actual parameters from the video frames. However, existing AI models are computationally expensive, making them unsuitable for real-time applications.

This study aims to develop a lightweight computer vision model that can predict LPBF process parameters accurately and efficiently, benefiting a NAS algorithm.

Dataset

We are dealing with a large dataset to analyze the effect of laser power and laser dot speed in 316 L stainless steel bulk material (Blanc et al., 2023). It contains videos of the 3D printing process recorded by an on-axis 20,000 frames per second camera. A process is defined by the printing of a line by the laser on the material for a given set of parameters, a specific laser power, and laser dot speed. The dataset is composed of 93,116,645 frames for 678,708 lines, and so 678,708 videos, for a parameters' distribution shown in Fig. 1. These videos contain grayscale frames.

The dataset in Blanc et al. (2023) compared the performance of four off-the-shelf open-source video action classification models. These are the 3D-CNN-based methods 3D ResNet (Blanc et al., 2023), SlowFast (Feichtenhofer et al., 2019), the transformers MViT (Fan et al., 2021) and Swin3D (Yang et al., 2025).

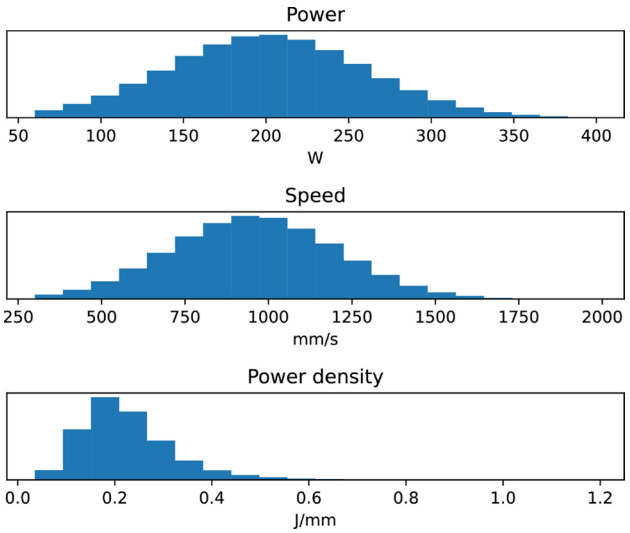


Fig. 1 Parameters' distribution of the dataset. The power density is not a parameter but is the ratio of the power and speed parameters

Pre-processing and data cleaning

The used dataset has a problem that goes into the fundamental camera feature; before the appearance of the laser, the camera frame consists of a large amount of noise in as you can see in the most left image in Fig. 2 due to the fixed ISO sensitivity of the sensor (Levoy, 2014).

This issue is common in the dataset and can occur at any time in the videos when the laser is off. Because these frames contain only noise and no useful information, it has a high potential to affect the AI model's predictions and pull it down. Therefore, removing these frames would help us to have more reliable AI models. It should be mentioned that we did not introduce these frames as a new label to the model because it can increase the complexity dramatically also this problem can be solved on the hardware side by letting the program know that the laser is off and then the monitoring process can stop monitoring which ends up with power saving on the software and power usage. To clean the dataset, we split the videos containing these frames and removed the problematic frames. This process resulted in a dataset of 1,228,921 videos.

There was another problem with the dataset, which made it to be heavy-duty for the training process, and it was its memory size. We solved this by cropping the frames in a way to have 41 × 41 frames. The original size of the frames was 128x128 pixels. In this process, for a given video, we first computed the average location of the laser point and then cropped all the frames around this position. Using this method helped us to keep the laser point at the same position for each video and frame, which makes the dataset more consistent. However, this cropping process did not affect the observation of the movements of the laser point. This process is similar to the pre-processing method, which has been done by Blanc et al. (2023).

In addition, we normalized the frames into a scale of [0, 1]. Input and output (laser's power and speed values) were normalized using the means and standard deviations that have been provided for comparative purposes (Blanc et al., 2023).

Finally, we used AI models with a fixed input size; therefore, we had to clip the videos into time windows. To do so, we implemented a generic dataset object that can handle both variable-length videos and fixed-length ones. The fixed-length videos are obtained by cropping the video every N frames. We did not use too-short videos from the modified dataset. This process ended up with 2383, 101 samples with 32-frame data points. At this stage, we had a clean and normalized dataset that was ready to be used to train both fixed-input-length models and variable-input-length models.

Architecture design

In the following sections, human-designed AI models and machine-designed AI models are introduced, respectively.

Human designed AI model

We first developed a human-designed architecture. Designing the model to achieve good performance and converge to acceptable accuracy took weeks; for some datasets and applications, it can take months. To design a competitive AI model, we designed different architectures and finally obtained one that uses 3D data as input, within which the third dimension is time. For this purpose, data was partitioned with time windows using a 3- Dimensional Convolution Neural Network (3D-CNN). This includes a 3D matrix, which uses the frames as the first 2 dimensions of the data and time as the third one. Then, by processing three dimensions at the same time, the AI model can learn from the spatial and temporal information of the videos, as well as a combination of both. The human-designed architecture– which is a 3D-CNN network– is shown in Fig. 3, so-called 3D-CNN-1.

The human-designed architecture, trained using 70% of the dataset, leads to an Root Mean Square Error (RMSE)

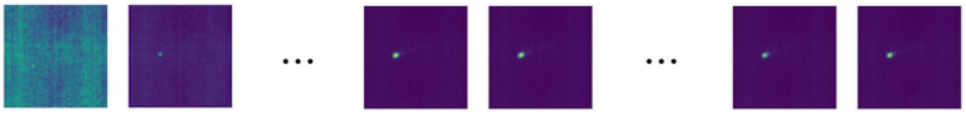


Fig. 2 Example of a video from the RAISE-LPBF dataset. The video is cropped, and the color is a mapping of the pixel's value

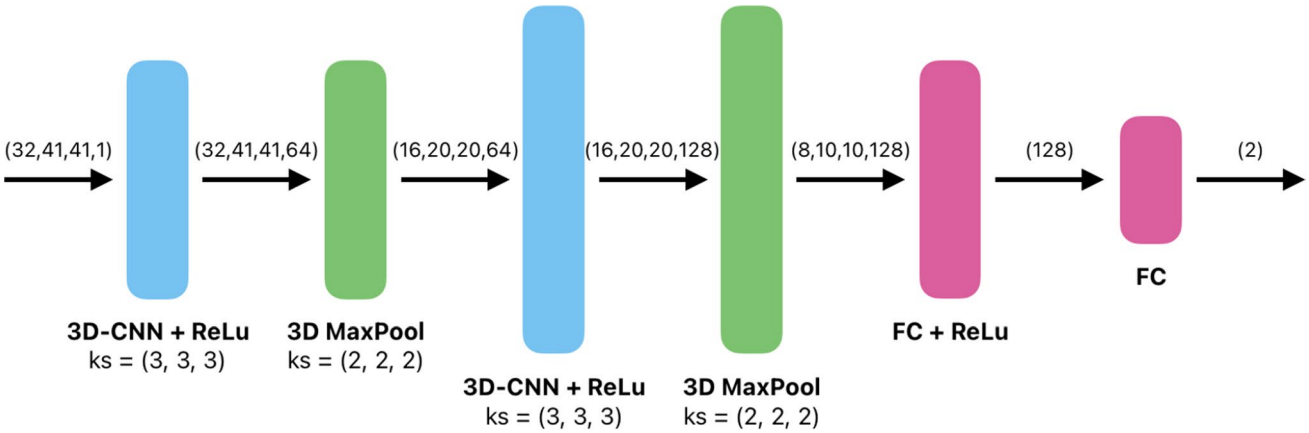


Fig. 3 3D-CNN-1 model's architecture (human-crafted). *ks* stands for kernel size

(Armstrong & Collopy, 1992)—Eq. 1– on the laser power of 94.5W, on the laser dot speed of 16.4 mm s^{-1}, and on the power density of 0.0461 j/mm. The model does not directly predict the power density but does predict the power and speed parameters ratio. The next step is going to describe the NAS algorithm and how it is going to find the optimized architecture in its search space.

$$RMSE\,(x_i - \hat{x_i}) = \sqrt{\frac{\Sigma_{i=1}^{N}\,(x_i - \hat{x_i})^2}{N}} \tag{1}$$

In Eq. 1, N is number of values, P is number of parameters, x_i is i^th value, and $\hat{x_i}$ is i^th predicted value.

Neural architecture search (NAS)

In industrialized NAS, mentioned pillars (Salmani Pour Avval et al., 2025) are further refined through hardware-aware optimizations and resource-efficient approaches. The SSp is often adapted to incorporate domain-specific constraints, ensuring that the candidate architectures meet not only performance metrics but also real-world operational requirements such as energy efficiency and latency. Simultaneously, the SSt is enhanced with advanced optimization algorithms that balance exploration and exploitation, effectively reducing the computational overhead and search time. Meanwhile, the VSt adopts efficient performance estimation methods to rapidly gauge the true potential of each architecture without the need for exhaustive training. This synergistic integration of pillars fosters a NAS framework that is both scalable and robust, paving the way for accelerated AI model development and seamless deployment in industry-scale environments.

To continue, for optimizing the architecture for our industrial application, we used Neural Architecture Search (NAS) algorithms (Salmani Pour Avval et al., 2025) in the following paragraphs.

Defined search space

A SSp is a necessary component of any optimization problem, as it defines the domain over which candidate solutions can be evaluated. Without it, the problem is undefined: there is no structure to guide search, no feasible set to constrain it, and no basis on which optimality can be determined. This applies equally to discrete, combinatorial, and continuous settings. Importantly, the SSp must also be limited. In high-dimensional or combinatorially large spaces, the number of possible solutions increases exponentially, making exhaustive search intractable and degrading the performance of even heuristic or sampling-based methods. This is a direct consequence of the curse of dimensionality (Barkat Ullah et al., 2008). A bounded SSp enables the use of algorithmic strategies such as pruning, surrogate modeling, or adaptive sampling, which rely on structure and prior information to reduce the number of evaluations required. It also mitigates overfitting in data-driven models by constraining complexity. For these reasons, the definition and limitation of the SSp are not auxiliary design choices but necessary conditions for efficient and meaningful optimization (Salmani Pour Avval et al., 2025).

Our defined SSp is constructed of two main parts. The first part is responsible for the first stage of the architecture, which is called *blocks* in this article, and the second part is responsible for the head of the networks, which is the combination of some Fully Connected (FC) layers.

The *block* part has the Rectified Linear Unit (ReLU) activation function after a 3D convolution layer and a pooling layer, which is shown in Fig. 4. each model either has three, two, or one unique *block*, which are connected in series to make the feature extraction section. The convolution layers either have a kernel of size (1, 1, 1), (3, 3, 3), or (5, 5, 5).

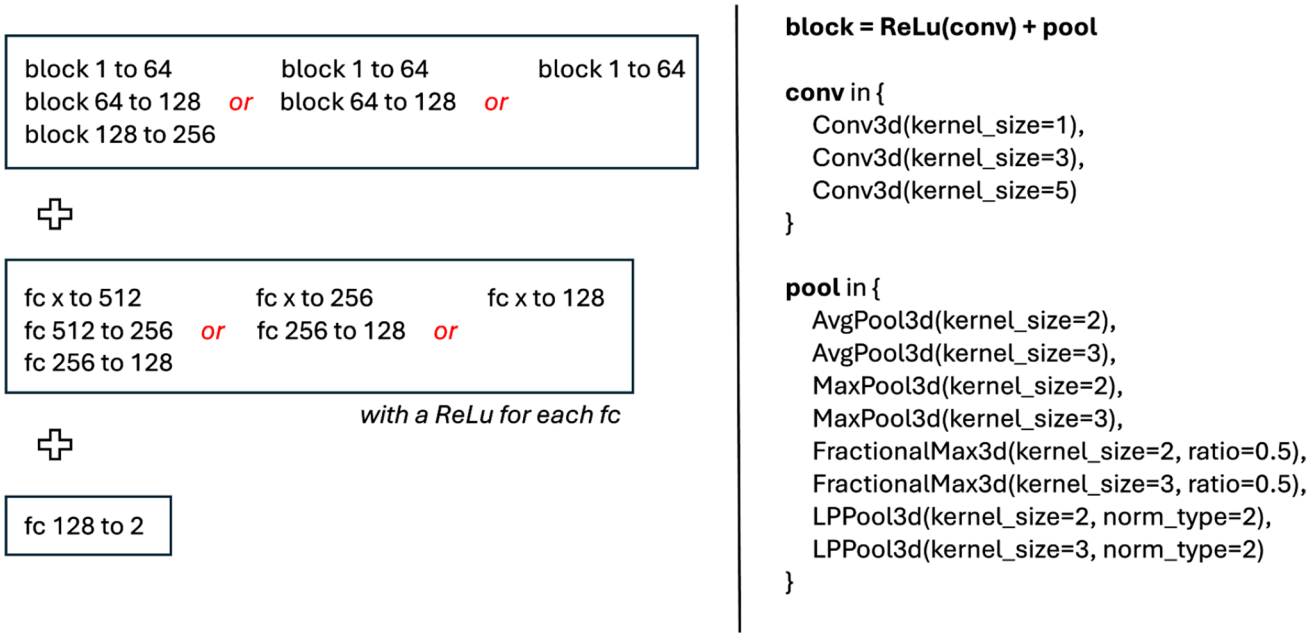


Fig. 4 Search space definition for the neural architecture search. *FC* stands for fully connected

Moreover, the pooling functions have two options: maximum pooling or average pooling. Within which a fractional maximum pooling has the ratio of 0.5, and the average pooling is a squared average pooling layer. Finally, the pooling functions' kernels are either of size (2, 2, 2) or (3, 3, 3). The input and output dimensions of the convolutions are fixed and detailed in Fig. 4.

The second part of the model, which is the head, has three, two, or one fully connected layers. Like the feature extraction part, the input and output dimensions of the head are detailed in Fig. 4. Each fully connected layer is followed by a ReLU activation function to bring the nonlinear complexity to the AI models.

Finally, the last layer of the architecture has a fully connected layer with an output dimension of 2. These two outputs were the laser power and the laser dot speed. There are no classical ML methods because they need intensive processing power and huge memory (Vapnik & Vapnik, 1998), which is never feasible with our dataset.

In general, the SSp consists of 8,610 models. To efficiently explore this vast space to find the most accurate model, several techniques could be considered; we decided to use a surrogate model method to check out if the DNA of the models can give important information about the performance of the owner beforehand of the training.

Surrogate model method as a validation strategy

In our SSp, every model is defined using an array of string variables that have all the hyper-parameter information of the topology of the model, like $conv\_1\_1\_64$, avg_pool_2, $conv\_5\_64\_128$, max_pool_2, *flatten*, fc_256, fc_128, *skipped*. Having all the important information for generating a model brings this challenge: is it possible to make a surrogate model using LSTM structures without using Large Language Models (LLMs) to predict the performance?

Therefore, a surrogate model is designed to predict the performance of different AI models by learning from the performance of a fraction of them in the SSp and predicting the rest. Therefore, the SSp was split into three sets: one thousand architectures were dedicated to training the surrogate model, another thousand were used to validate it, and others were left aside for testing. The architectures that belonged to the last set did not need any training, as the surrogate model would predict their performance. The main interest of this method is to reduce the computational cost of the NAS algorithm while computing its performance in the VSt step. As a result, instead of training all 8,610 architectures, we only need to train 2,000 architectures and record the training information. Then this information (like layer type, layer number, etc.) is used to train a surrogate model, which allows us to predict the performance of different architectures using a lower cost. To prepare the data for training the surrogate model, we used the AI models' architectures (as a string of DNA) as input. The layers were one-hot encoded as a vector of twenty values, as there were twenty different layers in the search space. Hence, the input was a variable-size 2D matrix with one dimension of twenty values and the other equal to the number of layers in the architecture. The surrogate model consisted of an encoder and a Multilayer Perceptron (MLP). The encoder's role was to extract the architecture's features by encoding them into

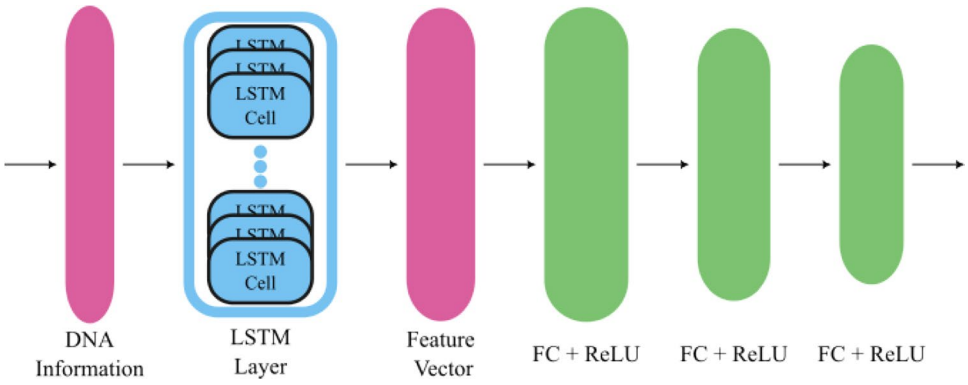


Fig. 5 Surrogate model for predicting the performance of AI models

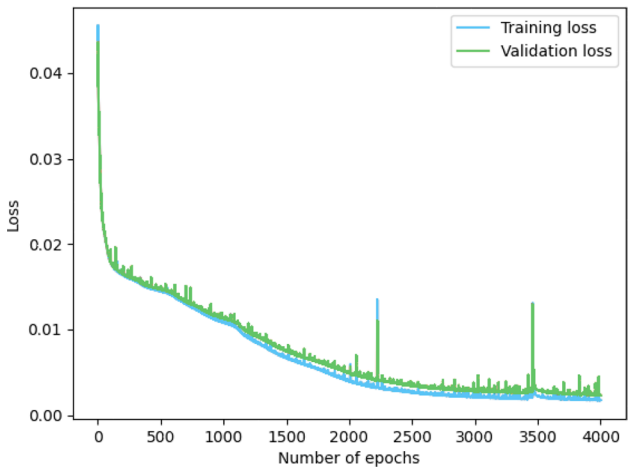


Fig. 6 Autoencoder performance of the surrogate model

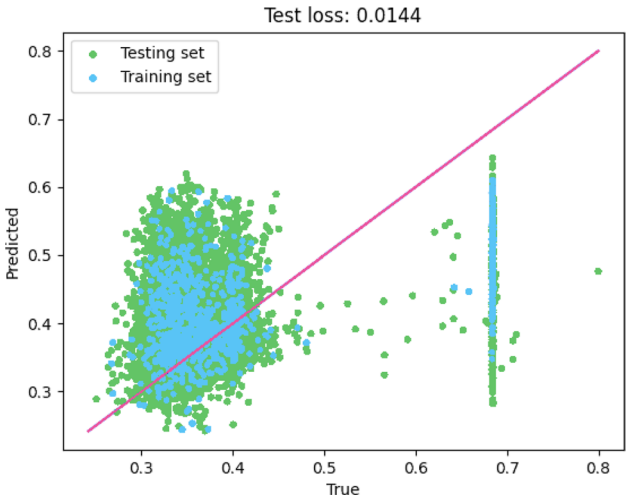


Fig. 7 MultiLayer perceptron evaluation of the surrogate model. The line is the target graph, the best predictions

Table 1 DNA information of the sample AI model that is in Fig. 9 is the values in the table plus DNA string which is: "*3D-CNN → ReLU → MaxPool → FC → ReLU → FC → ReLU → FC*"

Number of Layers	CNN layers	ReLU layers	Pooling layers	FC layers
8	1	3	1	3

a ten-value vector. The MLP section then used this vector to map them into the AI model's performance space (Fig. 5).

The encoder was a Long Short-Term Memory (LSTM) with a single layer with 10 cells in architecture and was trained using an autoencoder. The autoencoder's decoder consisted of two LSTMs with the same number of cells. The first one was used to rescale the encoded architecture's vector, a ten-value vector, to the original size, a twenty-value vector, and the second one was used to decode the architecture by iteratively reconstructing the original matrix. As shown in Fig. 6, the training and testing loss evolution were very good. The encoder was then ready to be used in the surrogate model. The autoencoder is trained using the DNA information, which is shown in Table 1.

In the surrogate model, the MLP section consisted of three fully connected layers with a ReLU activation function (Fig. 5). Unfortunately, the surrogate model was not able to predict the performance of the AI models, as shown in Fig. 7. There is a high possibility of designing a surrogate model using more complex and larger structures that are able to predict the performance, but the aim of this article is not to design them. As a result, we will postpone the surrogate structure design to the future. But, we consider that the information from the string type DNA information cannot lead to the performance at the moment, and we believe that due to the fact that the performance of the AI models is hardly wired to the data, it is even harder duty. Meaning, if we design an AI model that is working for a random image classification dataset, and transfer it to the CIFAR-10 dataset, the performance of it will not stay the same, and the complexity of the dataset can alter the performance. As a result, for this article, we used the low-fidelity NAS method for the architecture selection and then fully trained the selected AI model for measuring the final performance (Salmani Pour Avval et al., 2025).

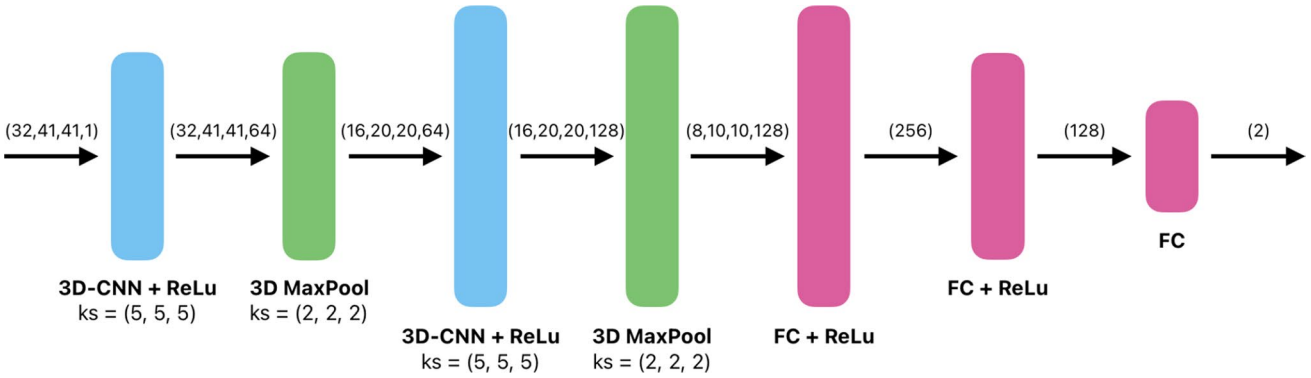
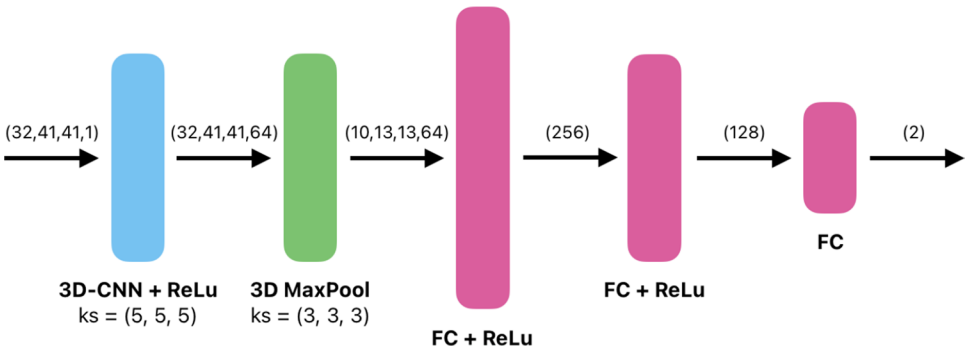


Fig. 8 3D-CNN-2 model's architecture. *ks* stands for kernel size. This model is the best-ranked model in the NAS algorithm

Fig. 9 3D-CNN-3 model's architecture. *ks* stands for kernel size. This model has the fewest number of layers, which is present in the top-ranked architectures in the NAS algorithm



Training-base low-fidelity validation method for neural architecture search

$$Score = \frac{1}{N}\sum_{i=0}^{N-1}\frac{(power_i - \hat{power_i})^2 + (speed_i - \hat{speed_i})^2}{2} \quad (2)$$

There is a guarantee that the best-performing AI model in the SSp can be found by training-based NAS algorithms despite their low efficiency because it has to train all the architectures in the SSp, which makes the process a time and energy-demanding one. These algorithms are sometimes not feasible due to the high number of architectures in the SSp. To make the expensive computational problems cheaper, we aimed to make the algorithm faster while keeping the performance of the training methods. Therefore, we selected the low-fidelity approach and then trained the selected AI model with the whole dataset. Therefore, this new direction made us train and evaluate the architectures on a tiny subset of the dataset and for a few epochs. To make this subset, we made some trial and error at the beginning to find the right amount of data that is needed, which can help the algorithm in the model selection and validation. This subset consisted of 0.1% of the dataset, with 70% dedicated to training the AI models and 30% to evaluating them. For more information about the architectures, you can check the results in Table 3 in Appendix 1. In this figure, the scores represent the mean square error of the architectures on the testing set: the lower it is, the better it is. For calculating the score, we used the error of two outputs of the models, which are the *power* and *speed* that are presented in Eq. 2

To train and test the top architectures, the whole dataset is divided into two parts: 70% for training and 30% for evaluation. It should be mentioned that models are trained just for 10 epochs to calculate the score. After applying the NAS algorithm to the dataset using the defined SSp, the algorithm returned the order of the models in a way that we can choose the model according to our criteria. The reason that we did not define the criteria at the beginning is that the selected model can be changed from one edge device to another because of their hardware structure and their ability to process different variables. We selected three different models called: the best architecture, the shortest architecture, and the simplest architecture. Later, these models are trained. We named them *3D-CNN-2*, *3D-CNN-3*, and *3D-CNN-3*, respectively. Their architectures are shown in Figs. 8, 9 and 10, respectively. The best performing model is the one that appeared at the top of the list by our NAS algorithm. The shortest model is the one that has the least number of layers in the top list. The simplest model is the model that has the fewest number of parameters. In some edge devices, we have some processors that are tending to have faster in series calculations which are good for the models that has least number of parameters, but some other type of devices like Jetson series (Mazzia et al., 2020; Smolyanskiy et al.,

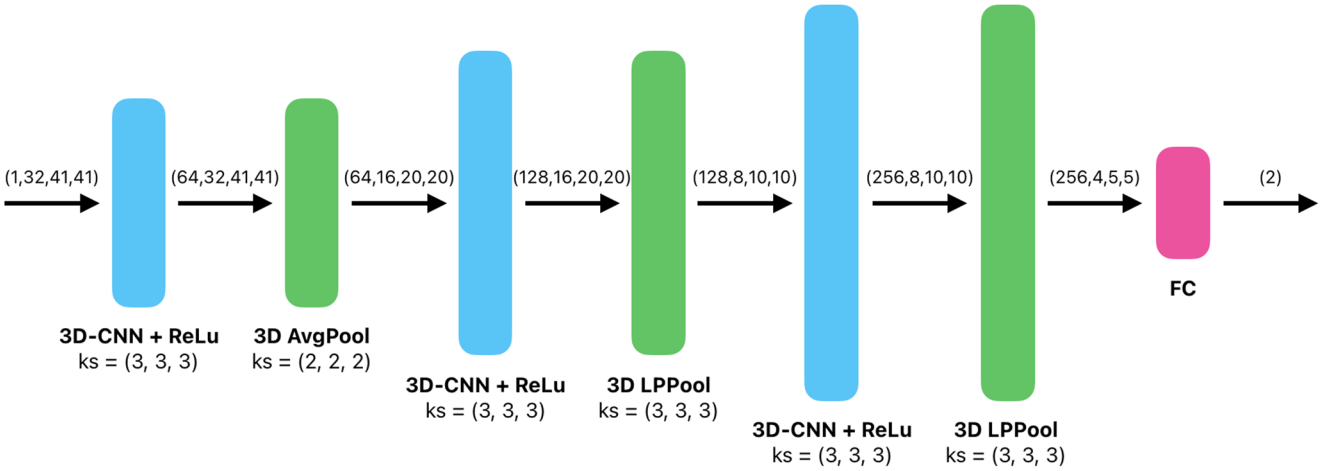


Fig. 10 3D-CNN-4 model's architecture. *ks* stands for kernel size. This model has the fewest number of parameters, which is a characteristic of the top-ranked architectures in the NAS algorithm

Table 2 Benchmark results of all the models

NAS ranking	Model	Speed RMSE	Power RMSE	No. parameters
N/A	SlowFast	67.6	11.12	34848090
1	*3D-CNN-2*	93.2	16.6	27728002
702	*3D-CNN-1*	94.5	16.4	13330690
16	*3D-CNN-4*	103.1	17.2	**1239554**
4	*3D-CNN-3*	105.8	16.8	27730434
N/A	3D ResNet	157.8	19.50	32735698
N/A	MViT	191.9	24.88	36892274
N/A	Swin3D	205.0	25.34	28238480

Our models are highlighted in italic. N/A means "not available". The model named 3D-CNN-1 is the human-crafted one. In the No. parameters, the bold value shows the model with fewest number of parameters

2017) tend to be good in in parallel processes which makes the fewer number of layers a benefit for the edge processing unit.

In this table, we compared the RMSE values with each other because this metric is the only metric that is provided by the introduction article of the dataset (Blanc et al., 2023). As shown in Table 2, the 3D-CNN-2's RMSE on laser power was 93.2, on laser dot speed was 16.6, and on power density was 0.0452. The 3D-CNN-3's RMSE on the same parameter predictions were 105.8, 16.8, and 0.0512, respectively.

In this article, we used the approach to consider the number of parameters in a model as complexity (Huang & Gao, 2022; Canziani et al., 2016), which is one of the methods that is used in literature because there is a positive correlation between the number of parameters and complexity. In addition, the number of parameters shows the number of mathematical operations that should be done indirectly (Bian et al., 2025; Tobiasz et al., 2023), and that is why we consider the number of parameters as a metric to show how fast the model works on edge devices. However, some edge devices that contain graphics processing units (GPUs) or tensor processing units (TPUs) are quicker with models that have a larger number of parameters but less number of layers (Mazzia et al., 2020; Smolyanskiy et al., 2017). Here is why we announced two models with the fewest number of parameters (3D-CNN-4) and the fewest layers (3D-CNN-3). Speaking of robustness, reducing the number of parameters in AI models can enhance their robustness: simpler models are less prone to overfitting and demonstrate improved resistance to adversarial perturbations. For instance, in the domain of medical image analysis, networks with significantly fewer parameters exhibited superior performance under adversarial attacks while maintaining comparable accuracy on clean data. Moreover, theoretical and empirical work has emphasized that models with low effective complexity tend to generalize better and show greater robustness to input variation (Budnikov et al., 2025; Rodriguez et al., 2022).

Results and discussion

After training and testing the defined architectures by humans and NAS algorithm, we put the results in Table 2. Our architectures outperformed three of the proposed previous models; they all had quite similar performances despite the massive difference in the parameter numbers- the fewer, the better. This research aimed to find a competitive architecture according to its performance while increasing the

potential of using it on edge devices with low latency and higher efficiency, which can be concluded from the difference between the number of parameters. As shown in Table 2, introduced AI models were created with fewer parameters than those proposed in the previous paper, which makes them suitable for the Edge-Devices and industrialized AI models due to the less complexity, which also can increase the potential of generalization and decrease the chance of overfitting.

Training the architectures in a subset of the dataset and on a few epochs tends to highlight the architectures that converge faster and with acceptable and competitive accuracy. We were looking for an AI model that is trainable in a few epochs; however, we managed to find architectures with good and compatible accuracy. Analyzing the results, with more time and a larger dataset partition, it is possible to optimize the topology of the AI models with higher accuracy. In addition, it could have been interesting to work on the surrogate model using different methods as well, which makes the NAS algorithm quicker and more accurate.

The architectures outperformed the previous ones in terms of accuracy over parameter numbers. Also, we showed that against the main mindset trend of the researchers, industrial applications often do not require complex architectures; rather, simpler ones, as built in this paper, are sufficient, generalized, and reliable. This is an important conclusion, as it is always better to use a AI model with fewer parameters, as it has less chance to overfit, is faster to train, consumes less, and is faster to run.

Conclusion and future direction

In this work, a training-based low-fidelity NAS algorithm for regression problems was developed to design better-performing and less complex architectures. It was also applied to a dataset coming from the metal additive manufacturing industry. It should be mentioned that NAS designed architectures have higher potential than traditional and off-the-shelf models to work in real-time, and they become more energy-efficient algorithms, which means the next step in the automated AI model design field, the main focus will be on three different branches:

- Within the scoring methodology, there is a substantial need to develop algorithms that explicitly account for complexity.
- Within industrial applications, robustness constitutes a critical parameter, and its incorporation into the validation process enhances the accuracy and reliability of model optimization.
- Model training is computationally intensive in terms of both time and energy, creating a strong demand for algorithms capable of predicting model performance without full training, or with substantially reduced training epochs.
- A random partitioning strategy was employed to allocate a subset of the dataset for low-fidelity validation. Developing principled methods for mini-batch selection has the potential to enhance the scoring methodology and improve predictive accuracy.

Appendix 1: Neural architecture search (NAS) results

See Table 3.

Table 3 Neural architecture search's top models when training all of them on few epochs

Layers									Model
0	1	2	3	4	5	6	7	8	Score
conv_5_1_64	max_pool_2	conv_5_64_128	max_pool_2	fc_256	fc_128				0.25053099
conv_5_1_64	avg_pool_2	conv_5_64_128	max_pool_2	fc_256	fc_128				0.26365426
conv_5_1_64	avg_pool_2	conv_5_64_128	fracmax_pool_2	fc_256	fc_128				0.26365426
conv_5_1_64	lp_pool_2	conv_5_64_128	max_pool_3	fc_256	fc_128				0.26476849
conv_5_1_64	avg_pool_2	conv_5_64_128	max_pool_3	fc_256	fc_128				0.26692758
conv_5_1_64	max_pool_3	conv_5_64_128	max_pool_3	fc_128					0.26692982
conv_5_1_64	max_pool_2	conv_5_64_128	max_pool_3	fc_256	fc_128				0.26796104
conv_5_1_64	max_pool_2	conv_5_64_128	max_pool_3	conv_5_128_256	max_pool_3	fc_128			0.26839190
conv_5_1_64	lp_pool_2	conv_5_64_128	max_pool_3	fc_128					0.26849535
conv_5_1_64	max_pool_3	conv_5_64_128	max_pool_3	conv_5_128_256	max_pool_3	fc_128			0.26875695
conv_3_1_64	max_pool_2	conv_3_64_128	lp_pool_3	conv_5_128_256	max_pool_2	fc_256	fc_128		0.27148522
conv_3_1_64	lp_pool_2	conv_3_64_128	lp_pool_3	conv_5_128_256	lp_pool_2	fc_512	fc_256	fc_128	0.26839190

The scores are the mean square error of the models on the testing set: the lower it is, the better it is. The ReLU activation functions are not shown

References

Ahmadian, M. T., Nasrabadi, V. Y., & Mohammadi, H. (2010). Nonlinear transversal vibration of an axially moving viscoelastic string on a viscoelastic guide subjected to mono-frequency excitation. *Acta Mechanica, 214*(3), 357–373.

Armstrong, J. S., & Collopy, F. (1992). Error measures for generalizing about forecasting methods: Empirical comparisons. *International Journal of Forecasting, 8*(1), 69–80.

Artin, J., Valizadeh, A., Ahmadi, M., Kumar, S. A., & Sharifi, A. (2021). Presentation of a novel method for prediction of traffic with climate condition based on ensemble learning of neural architecture search (nas) and linear regression. *Complexity, 2021*(1), 8500572.

Barkat Ullah, A.S., Sarker, R., & Cornforth, D. (2008) Search space reduction technique for constrained optimization with tiny feasible space. In proceedings of the 10th annual conference on genetic and evolutionary computation (pp. 881–888).

Benmeziane, H., El Maghraoui, K., Ouarnoughi, H., Niar, S., Wistuba, M., & Wang, N. (2021) Hardware-aware neural architecture search: Survey and taxonomy. In IJCAI (pp. 4322–4329).

Bian, S., Yan, M., & Venkataraman, S. (2025). Scaling inference-efficient language models, arXiv preprint arXiv:2501.18107

Blanc, C., Ahar, A., & De Grave, K. (2023). Reference dataset and benchmark for reconstructing laser parameters from on-axis video in powder bed fusion of bulk stainless steel. *Additive Manufacturing Letters, 7*, Article 100161.

Bowler, A. L., Pound, M. P., & Watson, N. J. (2022). A review of ultrasonic sensing and machine learning methods to monitor industrial processes. *Ultrasonics, 124*, Article 106776.

Budnikov, M., Bykova, A., & Yamshchikov, I. P. (2025). Generalization potential of large language models. *Neural Computing and Applications, 37*(4), 1973–1997.

Canziani, A., Paszke, A., & Culurciello, E. (2016). An analysis of deep neural network models for practical applications, arXiv preprint arXiv:1605.07678

Cao, L., Li, J., Hu, J., Liu, H., Wu, Y., & Zhou, Q. (2021). Optimization of surface roughness and dimensional accuracy in lpbf additive manufacturing. *Optics & Laser Technology, 142*, Article 107246.

Cheng, L., Yaghoubi, V., Van Paepegem, W., & Kersemans, M. (2021). Quality inspection of complex-shaped metal parts by vibrations and an integrated mahalanobis classification system. *Structural Health Monitoring, 20*(6), 3075–3091.

Chen, H., Zhang, B., Zheng, X., Liu, J., Doermann, D., & Ji, R. (2020). Binarized neural architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*, 10526–10533.

Creac'h, C., Bertholon, A., Convers, P., Garcia-Larrea, L., & Peyron, R. (2015). Effects of aging on laser evoked potentials. *Muscle & Nerve, 51*(5), 736–742.

Dunbar, A. J. (2016). Analysis of the laser powder bed fusion additive manufacturing process through experimental measurement and finite element modeling. The Pennsylvania State University.

Fan, H., Xiong, B., Mangalam, K., Li, Y., Yan, Z., Malik, J., & Feichtenhofer, C. (2021). Multiscale vision transformers. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6824–6835).

Feichtenhofer, C., Fan, H., Malik, J., & He, K. (2019). Slowfast networks for video recognition. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6202–6211).

Feuerriegel, S., Maarouf, A., Bär, D., Geissler, D., Schweisthal, J., Pröllochs, N., Robertson, C. E., Rathje, S., Hartmann, J., Mohammad, S. M., et al. (2025). Using natural language processing to analyse text data in behavioural science. *Nature Reviews Psychology, 4*(2), 1–16.

Gottapu, R. D., & Dagli, C. H. (2020). Efficient architecture search for deep neural networks. *Procedia Computer Science, 168*, 19–25.

Gupta, S., & White, M. (2021). Improved on-device ml on pixel 6, with neural architecture search, Google Research Blog [https://blog. research. google/2021/11/improved-on-device-ml-on-pixel-6-with. html] (Accessed 8 November 2021)

Hao, C., Chen, Y., Liu, X., Sarwari, A., Sew, D., Dhar, A., Wu, B., Fu, D., Xiong, J., Hwu, W.-m., et al. (2019). Nais: Neural architecture and implementation search and its applications in autonomous driving. In 2019 IEEE/ACM international conference on computer-aided design (ICCAD) (pp. 1–8). IEEE.

He, C., Tan, H., Huang, S., & Cheng, R. (2021). Efficient evolutionary neural architecture search by modular inheritable crossover. *Swarm and Evolutionary Computation, 64*, Article 100894.

Huang, K., & Gao, W. (2022). Real-time neural network inference on extremely weak devices: agile offloading with explainable AI. In Proceedings of the 28th annual international conference on mobile computing and networking (pp. 200–213).

Hu, M., Avval, S. S. P., He, J., Yue, N., & Groves, R. M. (2025). Explainable artificial intelligence study on bolt loosening detection using lamb waves. *Mechanical Systems and Signal Processing, 225*, Article 112285.

Imran, T., & Naeem, M. (2025). *Image-based laser-beam diagnostics using statistical analysis and machine learning regression, 12*(5), 504.

Janai, J., Güney, F., Behl, A., Geiger, A., et al. (2020). Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and trends ® in computer graphics and vision, 12*(1–3), 1–308.

Kouprianoff, D., Luwes, N., Yadroitsava, I., & Yadroitsev, I. (2018). Acoustic emission technique for online detection of fusion defects for single tracks during metal laser powder bed fusion.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images

Levoy, M. (2014). Noise and iso

Li, L., & Talwalkar, A. (2020). Random search and reproducibility for neural architecture search. In Uncertainty in artificial intelligence, PMLR (pp. 367–377).

Liu, H., Simonyan, K., & Yang, Y. (2018). DARTS: Differentiable architecture search. arXiv preprint arXiv:1806.09055.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., & Murphy, K. (2018). Progressive neural architecture search. In Proceedings of the European conference on computer vision (ECCV) (pp. 19–34).

Mazzia, V., Khaliq, A., Salvetti, F., & Chiaberge, M. (2020). Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application. *IEEE Access, 8*, 9102–9114. https://doi.org/10.1109/ACCESS.2020.2964608

Mellor, J., Turner, J., Storkey, A., & Crowley, E. J. (2021). Neural architecture search without training.

Mills, K. G., Salameh, M., Niu, D., Han, F. X., Rezaei, S. S. C., Yao, H., Lu, W., Lian, S., & Jui, S. (2021). Exploring neural architecture search space via deep deterministic sampling. *IEEE Access, 9*, 110962–110974.

Mussatto, A., Groarke, R., Vijayaraghavan, R. K., Hughes, C., Obeidi, M. A., Doğu, M. N., Yalçin, M. A., McNally, P. J., Delaure, Y., & Brabazon, D. (2022). Assessing dependency of part properties on the printing location in laser-powder bed fusion metal additive manufacturing. *Materials Today Communications, 30*, Article 103209.

Ravikumar, A., Sriraman, H., Saketh, P. M. S., Lokesh, S., & Karanam, A. (2022). Effect of neural network structure in accelerating performance and accuracy of a convolutional neural network with gpu/tpu for image analytics. *PeerJ Computer Science, 8*, Article e909.

Rodriguez, D., Nayak, T., Chen, Y., Krishnan, R., & Huang, Y. (2022). On the role of deep learning model complexity in adversarial

robustness for medical images. *BMC Medical Informatics and Decision Making, 22*(Suppl 2), 160.

Ross, M., McCormick, D., Bong, E., Arnett, D., Inmam, W., Horton-Smith, S., Alley, R., Jobe, R. K., Kotseroglou, T., Scheeff, M., et al. (1996). A high performance spot size monitor.

Rothfelder, R., Nahr, F., Chechik, L., Bartels, D., & Schmidt, M. (2023). A brief history of the progress of laser powder bed fusion of metals in Europe. *Journal of Manufacturing Science and Engineering, 145*(10), Article 100801.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV), 115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Salmani Pour Avval, S., Eskue, N. D., Groves, R. M., & Yaghoubi, V. (2025). Systematic review on neural architecture search. *Artificial Intelligence Review, 58*(3), 73.

Saxena, S., & Verbeek, J. (2016). Convolutional Neural Fabrics, Tech. rep.

Serin, G., Kahya, M., Unver, H., Gulec, Y., Durlu, N., & Erogul, O. (2016). A review of additive manufacturing technologies. in: 17th International conference on machine design and production. Bursa.

Smolyanskiy, N., Kamenev, A., Smith, J., & Birchfield, S. (2017). Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness. *IEEE International Conference on Intelligent Robots and Systems*. https://doi.org/10.1109/IROS.2017.8206285

Tobiasz, R., Wilczyński, G., Graszka, P., Czechowski, N., & Łuczak, S. (2023). Edge devices inference performance comparison. arXiv preprint arXiv:2306.12093.

Vapnik, V., & Vapnik, V. (1998). Statistical learning theory Wiley. *New York, 1*(624), 2.

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience, 2018*(1), 7068349.

Wan, A., Dai, X., Zhang, P., He, Z., Tian, Y., Xie, S., Wu, B., Yu, M., Xu, T., & Chen, K. (2020). Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 12965–12974).

Wen, W., Liu, H., Chen, Y., Li, H., Bender, G., & Kindermans, P.-J. (2020). Neural predictor for neural architecture search. In European conference on computer vision (pp. 660–676). Springer.

Yaghoubi, V., & Kumru, B. (2024). Retrosynthetic life cycle assessment: A short perspective on the sustainability of integrating thermoplastics and artificial intelligence into composite systems. *Advanced Sustainable Systems, 8*(5), 2300543.

Yang, Y.-Q., Guo, Y.-X., Xiong, J.-Y., Liu, Y., Pan, H., Wang, P.-S., Tong, X., & Guo, B. (2025). Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *Computational Visual Media, 11*(1), 83–101.

Yeung, H., Lane, B. M., Donmez, M., & Moylan, S. (2020). In-situ calibration of laser/galvo scanning system using dimensional reference artefacts. *CIRP Annals, 69*(1), 441–444.

Zhou, F., Kilickaya, M., Vanschoren, J., & Piao, R. (2024). Hytas: A hyperspectral image transformer architecture search benchmark and analysis. Tech. rep.

Zoph, B., & Le, Q. V. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.