applied sciences

MDPI

Article

Deep Learning and Bayesian Hyperparameter Optimization: A Data-Driven Approach for Diamond Grit Segmentation toward Grinding Wheel Characterization

Damien Sicard [1,2,3,*], Pascal Briois [2,*], Alain Billard [2], Jérôme Thevenot [3], Eric Boichut [3], Julien Chapellier [3] and Frédéric Bernard [1,*]

1 Laboratoire Interdisciplinaire Carnot de Bourgogne, PMDM, ICB-UMR6303, CNRS, Université de Bourgogne Franche-Comté, 9 Avenue Alain Savary, BP47870, CEDEX, 21078 Dijon, France
2 FEMTO-ST UMR 6174, CNRS, Université de Bourgogne Franche-Comté, UTBM, F-90010 Belfort, France
3 DIAMATEC, Route de Grachaux, 70700 Oiselay-et-Grachaux, France
* Correspondence: damien.sicard@utbm.fr (D.S.); pascal.briois@utbm.fr (P.B.); frederic.bernard@u-bourgogne.fr (F.B.)

Abstract: Diamond grinding wheels (DGWs) have a central role in cutting-edge industries such as aeronautics or defense and spatial applications. Characterizations of DGWs are essential to optimize the design and machining performance of such cutting tools. Thus, the critical issue of DGW characterization lies in the detection of diamond grits. However, the traditional diamond detection methods rely on manual operations on DGW images. These methods are time-consuming, error-prone and inaccurate. In addition, the manual detection of diamond grits remains challenging even for a subject expert. To overcome these shortcomings, we introduce a deep learning approach for automatic diamond grit segmentation. Due to our small dataset of 153 images, the proposed approach leverages transfer learning techniques with pre-trained ResNet34 as an encoder of U-Net CNN architecture. Moreover, with more than 8600 hyperparameter combinations in our model, manually finding the best configuration is impossible. That is why we use a Bayesian optimization algorithm using Hyperband early stopping mechanisms to automatically explore the search space and find the best hyperparameter values. Moreover, considering our small dataset, we obtain overall satisfactory performance with over 53% IoU and 69% F1-score. Finally, this work provides a first step toward diamond grinding wheel characterization by using a data-driven approach for automatic semantic segmentation of diamond grits.

Keywords: deep learning; Bayesian hyperparameter optimization; computer vision; semantic segmentation; U-Net; diamond abrasive grits; diamond grinding wheels

check for updates

Citation: Sicard, D.; Briois, P.; Billard, A.; Thevenot, J.; Boichut, E.; Chapellier, J.; Bernard, F. Deep Learning and Bayesian Hyperparameter Optimization: A Data-Driven Approach for Diamond Grit Segmentation toward Grinding Wheel Characterization. *Appl. Sci.* **2022**, *12*, 12606. https://doi.org/10.3390/app122412606

Academic Editors: Zhenglei He, Yi Man and Kim Phuc Tran

Received: 7 November 2022
Accepted: 5 December 2022
Published: 8 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Diamond grinding wheels (DGWs) have a central role in cutting-edge industries such as aeronautics, defense and spatial applications. DGWs are mainly produced by powder metallurgy (PM) technologies such as the sintering process [1]. As illustrated in Figure 1, a DGW is composed of two parts: a matrix and diamond abrasive grits. The matrix, generally a metallic alloy, ensures the retention of diamond abrasive grits and acts as a binder for the cohesion of the wheel. Characterizations of DGWs are essential in order to optimize the design and machining performance of such cutting tools. One way is to quantify the abrasive power of DGWs. This quantity could be seen as a complex relationship between matrix wear and loosening diamond grits. Non-exhaustively, we could distinguish several approaches for the evaluation of DGW abrasive power: 3D topographic analysis of active surfaces [2,3], micro-geometric approaches [4,5] or in situ acoustic emission monitoring [6].

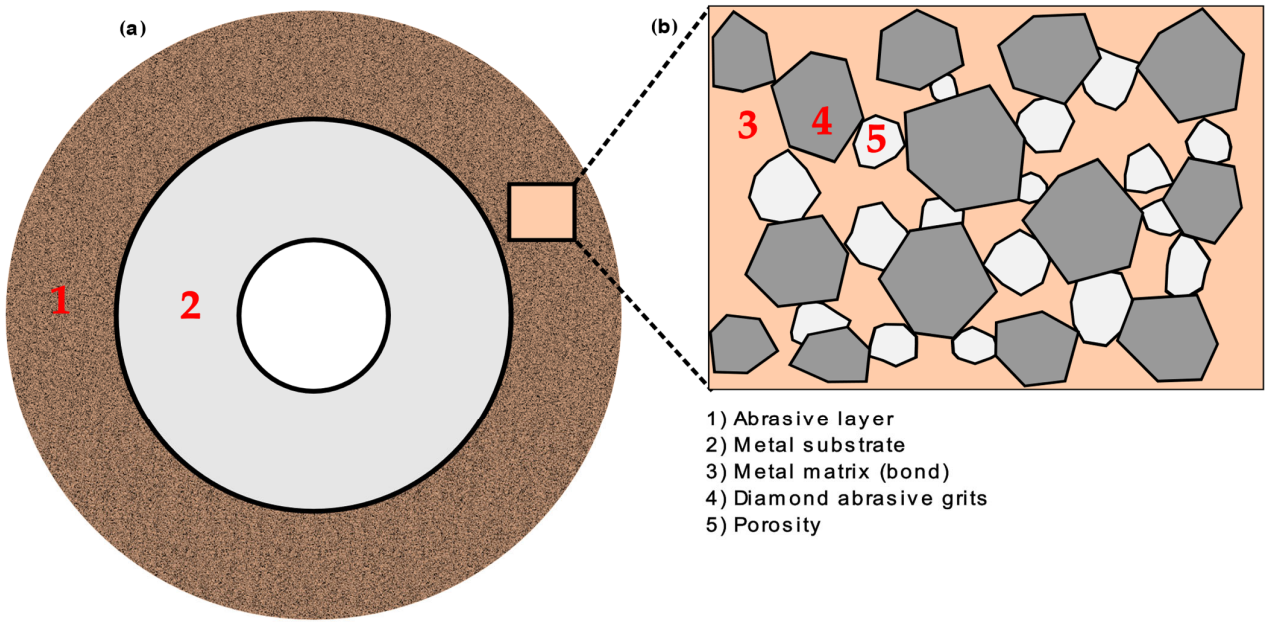

Figure 1. Constitution of a diamond grinding wheel (DGW) (**a**) and its abrasive layer (**b**). Phenomena involved during abrasive machining: the quantification of abrasive power remains a challenging, opening research questions.

As the abrasive power evaluation systems are difficult to set up and use in an industrial environment, more promising approaches are computer vision-based methods. The aim of such methods is to leverage the knowledge of subject experts in order to characterize DGWs from images. In these approaches, one of the major challenges is the detection of diamond grits and their interfaces. Generally, diamond detection methods rely on manual operations on DGW images. However, these methods are time-consuming, error-prone and inaccurate. In addition, due to the diversity of DGW image acquisition parameters, it is challenging to generalize traditional image processing algorithms for an accurate semantic segmentation of diamond grits.

In recent years, in the data-driven era, with the rise of deep learning algorithms, semantic segmentation has become an extensively studied research area in the field of computer vision [7,8]. The aim of semantic segmentation algorithms [9] is to assign a separate class label to each pixel of an image for object detection purposes. Applications of such algorithms can be found in autonomous vehicles [10], where the pixel labels could be trees, cars, humans or roads, and in medical image analysis [11,12], where they could be cells, tumors or even brain activity analysis [13,14]. For DGW image segmentation, it is about diamond abrasive grits.

The semantic understanding of a scene by a computer is a challenging problem mainly solved today thanks to the development of convolutional neural networks (CNNs) by LeCun et al. [15] and the computational resources of modern computers. CNNs use successive convolution and pooling operations on an input image. The convolution product between the input image and a kernel extracts features from the input [16]. Then, the result, the feature map, is operated by a pooling layer to capture the essential features which describe the object of interest. This operation produces spatial reduction and a loss of object position information. Moreover, a fully connected network with non-linear activation functions is connected at the end to make predictions, and the whole network is trained by a backpropagation algorithm [17]. CNN constitutes the "fundamental blocks" of more advanced CNN-based architectures for computer vision applications [18,19].

It is well known that training a CNN from scratch is energy-intensive and requires huge labeled datasets [20,21]. That is why, to overcome this problem, it is established in the computer vision community to use transfer learning techniques [22–24]. Transfer learning

consists of using a pre-trained CNN model [25] (often called "backbones"), trained over a large standardized dataset model by shifting its weight over a new training dataset of interest (i.e., DGW images). For example, in computer vision (especially segmentation tasks), ImageNet [26] is a widely used dataset for a pre-trained model of over 14 million labeled images. Based on this observation, several new CNN architectures have been developed over the years, such as AlexNet [27], VGG [28], ResNet [29], GoogLeNet [30], U-Net [31], Inception [32], DenseNet [33] and EfficientNet [34].

In this work, we implement a U-Net architecture for the semantic segmentation of diamond grits, using ResNet34 backbones trained on ImageNet as an encoder. However, although transfer learning is efficient, with more than 8600 hyperparameter (HP) combinations, it is impossible to manually find the optimal configuration of HPs. Some examples of hyperparameters include the model architecture and training parameters such as epoch number, batch size or learning rates. For that reason, finding the best combination of hyperparameters remains challenging even for experts in the field. In this respect, we use an Automated Machine Learning (AutoML) [35] methodology by leveraging Bayesian Hyperparameter Optimization (BO) [36–40] and Hyperband [41] algorithms. Finally, our work is part of an ongoing effort to make deep learning models more accurate, reliable and easier to use for researchers, engineers and programmers.

The key contributions of this article include:

- A practical application and implementation of state-of-the-art (SOTA) deep learning computer vision algorithms for semantic segmentation of objects in images in the industrial field of DGW manufacturing.
- An assessment of segmentation performances of the corresponding models over state-of-the-art metrics.
- To the best of our knowledge, this work is the first published attempt of using a U-Net deep learning model for automatic segmentation of diamond grits from DGW images.
- The foundations of upcoming data-driven methods for DGW characterizations.
- A unified methodology for automatic configuration/calibration of deep learning models by leveraging Bayesian optimization and Hyperband algorithms, an AutoML framework for making deep learning approaches easier to use.

The remainder of this paper is organized as follows. Section 2 introduces the implemented U-Net deep learning architecture and the Bayesian and Hyperband algorithms, including the overall hyperparameter optimization strategy. Section 3 presents the data acquisition, software and hardware specifications, with an emphasis on manual hyperparameter complexity reduction. Section 4 provides the experimental results in two parts: Bayesian optimization experiments and diamond grit segmentation results. Finally, Section 5 summarizes the article, draws conclusions about this work and outlines future works.

2. Methods

The aim of this section is to progressively introduce our AutoML framework. The U-Net deep learning architecture and associated segmentation metrics are introduced. Then, the Bayesian optimization and Hyperband algorithms are presented in detail. Finally, a presentation of the overall Bayesian hyperparameter optimization strategy is given.

2.1. Model Architecture and Metrics

In the previous section, CNN basic architectures were introduced. These kinds of networks are well suited to capture the context of an image but with a loss of spatial information due to the down-sampling effect. This drawback, due to the alternation of convolution and max pooling layers, makes CNN basic architecture not well suited for semantic segmentation tasks. For that reason, we use the U-Net encoder–decoder model first introduced by Ronneberger et al. [31], as shown in Figure 2. The U-Net architecture is composed of two paths: the encoder and decoder. The encoder is a contracting path with 4 successive convolution blocks associated with max pooling steps. The convolution blocks

extract raw features from DGW images ("what is a diamond grits?"), and max pooling operations grab the most dominant ones. This implies a spatial reduction in the (XY) image axis by increasing the depth of the feature channels. A direct consequence is a loss of information about the position of the object (diamond) in the image. To address this issue, Ronneberger et al. introduced skip connections, an image concatenation mechanism that allows high-resolution spatial features to be cropped from the encoder and concatenated with the corresponding up-convolution of the expansion path. Indeed, the decoder is composed of an expanding path of 4 transposed convolution blocks and respective skip connections in order to learn spatial information (i.e., "where is the diamond?"). Furthermore, a fully connected layer is connected at the end of the decoder for classification purposes. In addition, we use the pre-trained weight of ResNet34 [29] as a feature extraction for the contracting path. This transfer learning technique allows us to save a large amount of training time, computational resources and leverage the U-Net architecture to perform semantic segmentation on a small DGW dataset of 153 images.

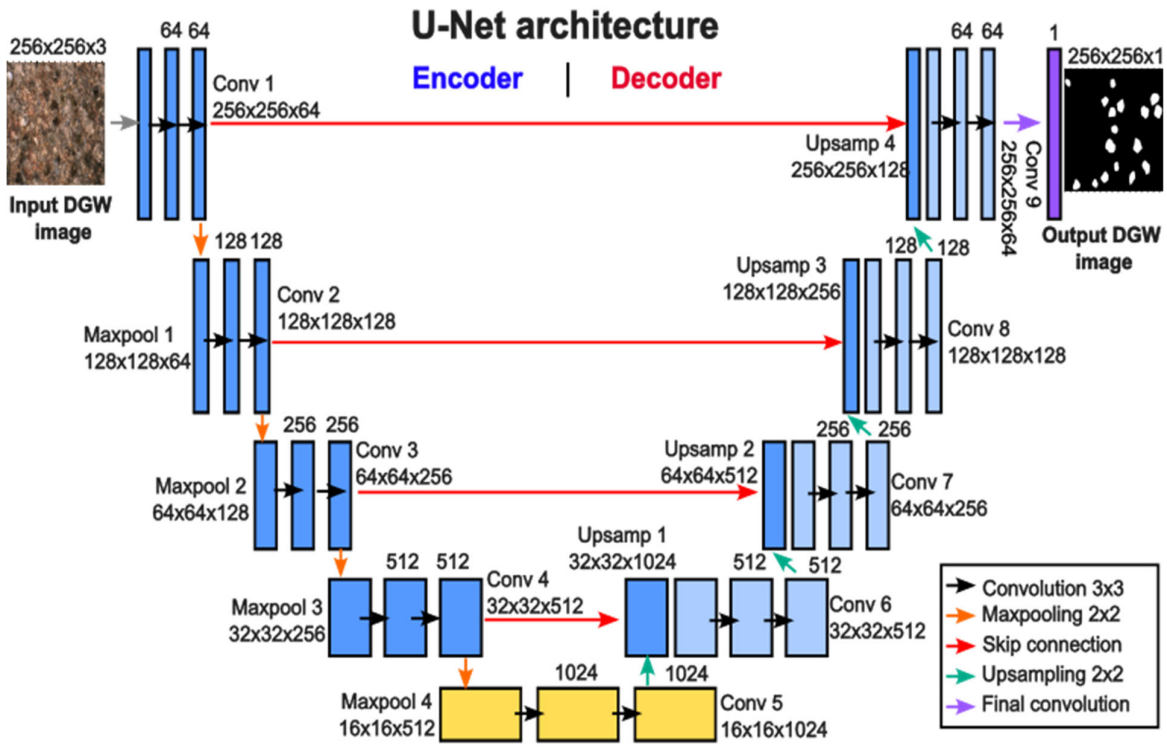

Figure 2. U-Net architecture for DGW image semantic segmentation.

Evaluating the performance of semantic segmentation tasks is essential in order to assess the prediction accuracy of the developed model. One way to access such performance prediction is to use the confusion matrix as a starting point for the definition of segmentation metrics [42,43]. Such a matrix is a tabular summary of correct and incorrect predictions.

From the confusion matrix, it is possible to extract the respective coefficients—True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN)—and use them to define two fundamental metrics used in our work: the Intersection Over Union (IoU) and the F1-score. These metrics are defined as follows:

$$\mathrm{IoU} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{1}$$

$$\mathrm{F}_1 = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}} \tag{2}$$

Although the IoU metrics are popular to assess the performance of the semantic segmentation model, they tend to penalize over and under segmentation prediction. Therefore, prediction performance tends to be underestimated with this index. That is why it is convenient to also use the F1-score defined in Equation (2).

2.2. Bayesian Optimization Methodology

Deep learning and more generally machine learning models are driven by a set of input parameters called hyperparameters. In the case of deep learning, this can be, e.g., network architecture, such as the number of convolution blocks, activation functions and training parameters such as epochs number or learning rate. Fine-tuning hyperparameters is essential to leverage the potential of the developed deep learning model. However, manually tuning hyperparameters is time-consuming and nearly impossible. For example, in the simple version of our model, there are more than 8600 hyperparameter combinations. Deep learning hyperparameter optimization (HPO) is an active research area [44–49]. In such problems, the goal is to maximize/minimize an expensive objective function whose closed form and its gradients are unknown. In this paper, we use a Bayesian optimization (BO)-based approach. BO is a black box (i.e., model-based) optimization algorithm searching for a global optimum of an unknown objective function from which observations can be made. The BO algorithm is composed of two components: a surrogate model for objective function approximation and an activation function for the integration of the regression history. In addition, it is worth noting that one of the main advantages of BO is its ability to learn from past experiments. Thus, the global optimization problem could be formulated as follows:

$$x^* = \underset{x \in X}{\operatorname{argmax}} f(x) \tag{3}$$

where x^* designs the hyperparameter input in the search space X, i.e., $f(x)$ is optimal. However, in order to use the BO for hyperparameter optimization, we assume the following constraints:

- We have a few input control parameters x (less than 20).
- The objective function is relatively "smooth" (i.e., not so discontinuous in the discrete meaning).

As a black box optimization approach, BO uses a surrogate model for the regression task (i.e., fit the unknown objective function). In this work, we used a statistical Gaussian Process (GP) as a surrogate model. The key ideas behind BO are illustrated in Figure 3.

From Figure 3, we can describe the underlying ideas behind the BO algorithm as follows:

(a) Based on the initial observation points (i.e., sample points measured) from the objective function $f(x)$, a first regression attempt is made using an initial version of a Gaussian Process model. In addition, a confidence interval is built due to the probabilistic nature of the GP. Then, an acquisition function $\alpha(x)$, derived from the GP model, is used to estimate the next computing step.
(b) The GP model has been updated from past observation points. Thus, the regression improves and the uncertainty area decreases toward the objective function. The next observation point x_n is computed by taking the global maximum of the associated acquisition function.
(c) Finally, the fitting quality is iteratively improved. When the regression becomes acceptable, the best hyperparameter configuration x^* is saved and the optimization problem can be solved by finding a global optimum of the GP model.

Several surrogate models have been developed in the field of BO [40,50]. The most common one is the Gaussian Process because of its acceptable performance and its simplicity of implementation. In addition, using GP as a regression model to fit the unknown objective function is appealing because it considers the uncertainty associated with each prediction.

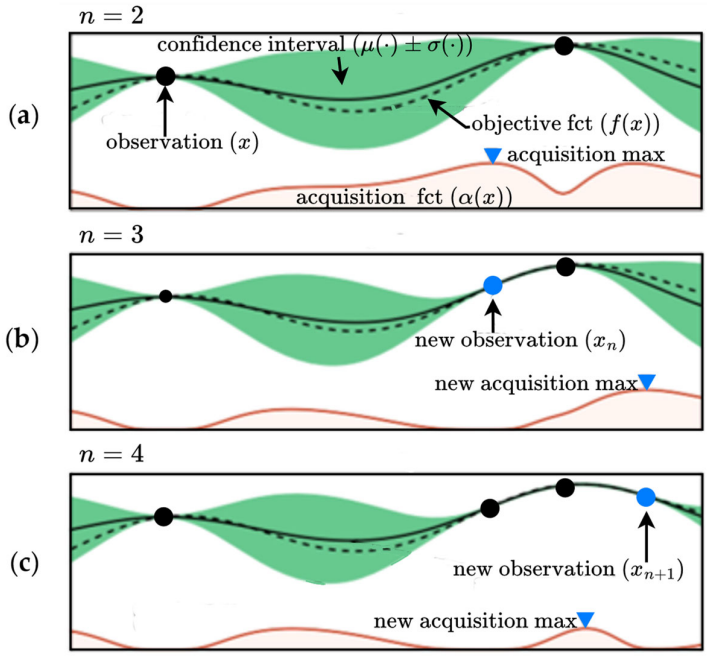

Figure 3. Bayesian optimization methodology: (**a**) Initialization of Gaussian Process model; (**b**) Update of Gaussian Process model; (**c**) Improve Gaussian Process model iteratively. Figure adapted from [36].

Given a set of observations $\mathbf{D_{1:t}} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x_t}, \mathbf{y_t})\}$, a Gaussian Process is a stochastic model defined by Equation (4) as the multivariate normal distribution **N**:

$$P(y|\mathbf{D_{1:t}}, x) = \mathbf{N}\left(\mu_t, \sigma_t^2\right) \tag{4}$$

where model predictions are represented by the mean term μ_t and the uncertainty by the variance σ_t^2, defined in Equations (5) and (6), respectively.

$$\mu_t = \mathbf{k}^{\mathbf{T}}\mathbf{K}^{-1}\mathbf{D_{1:t}} \tag{5}$$

$$\sigma_t^2 = k(x, x) - \mathbf{k}^{\mathbf{T}}\mathbf{K}^{-1}\mathbf{k} \tag{6}$$

$$\mathbf{K} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_t) \\ \vdots & \ddots & \vdots \\ k(x_t, x_1) & \cdots & k(x_t, x_t) \end{bmatrix} + \sigma_r^2\mathbf{I} \tag{7}$$

However, since the observation of real systems could be noisy, we introduced a sampling noise σ_r^2 in Equation (7). In Equations (5)–(7), **K** represents the covariance matrix. It describes how the observations are related to each other. In other words, the covariance matrix measures the correlation strength between observations by using a kernel function $k(\chi_1, \chi_1)$ as coefficients. The kernel function has a strong impact on the prediction quality of the GP surrogate model. In fact, it drives the output responses of the GP and the shape of the objective function that we can fit. That is why choosing it carefully according to the optimization problem is of major importance. In this paper, since deep learning metrics such as loss functions or validation IoU are continuous without periodicity, we used the Matérn kernel function that is a generalization of radial basis kernel (RBF) [51], defined by Equation (8):

$$k(x_i, x_j) = \frac{1}{\Gamma(\nu)2^{\nu-1}}\left[\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right]^{\nu} K_\nu\left[\frac{\sqrt{2\nu}}{l}d(x_i, x_j)\right] \tag{8}$$

where $\Gamma(\nu)$ is the gamma function, $d(x_i, x_j)$ the Euclidian distance between two observation points and K_ν is the modified Bessel function. The parameter ν controls the smoothness of the output function. In fact, a smaller ν leads to a sharper approximated function.

Thus, the first part of the BO algorithm used a Gaussian Process as a surrogate model of the objective function. The remaining part of the algorithm concerns the choice of an activation function. Such a function is used to decide which observation point x_t should be inferred next. In this work, we used the common Expected Improvement (EI) acquisition function defined by Equation (9):

$$\alpha(x) = \mathbf{EI}(x, D_t) = \int_{-\infty}^{+\infty} \max(y^* - y, 0) P(y \,|D_{1:t}, x) dy \text{ with } y^* = \min_{1<i<n} f(x_i) \tag{9}$$

Based on the updated GP model $P(y \,|D_{1:t}, x)$ and the observation database D_t, the activation function considers the region of high uncertainty with a large confidence interval and those with the minimum observed true objective function score y^* in order to compute the next sampling point x_t. Finally, the entire BO algorithm is summarized in Algorithm 1.

Algorithm 1: Bayesian optimization.

input: $(T, f(x), \alpha(x))$
initialization: $D_0 \leftarrow \{x_0, y_0\}$
random initialize the Gaussian Process
for $t \leftarrow 1$ **to** T **do**
    $x' \leftarrow \underset{x \in X}{\operatorname{argmax}} \alpha(x, D_t)$
    $y' \leftarrow f(x')$
    $D_t \leftarrow D_{t-1} \cup \{x', y'\}$
    Update GP model by computing $P(y' \,|D_t, x')$
end for
return best hyperparameters D^* such as $x^* = \underset{x \in X}{\operatorname{argmax}} f(x)$

2.3. Hyperband Early Stopping Algorithm

Although the BO algorithm allows us to automatically find the optimal hyperparameters in order to maximize IoU and F1-score segmentation metrics, it does not prevent the full test of unpromising experiments. For that reason and to save computational time resources, we included an early stopping mechanism in our BO loop for preventing unpromising training experiments. Hyperband is a bandit optimization algorithm developed by Li et al. [41]. It is based on the Successive Halving (SH) algorithm from Jamieson et al. [52]. The SH algorithm is resource-based, which means that it allocates a budget B, for example, training time to a set of hyperparameter configurations n. It evaluates the performance of all configurations by uniformly allocating the $\frac{B}{n}$ average resource. Then, it throws away the worst half of the performances and repeats the procedure until one configuration remains. However, a fundamental issue is how to choose the tradeoff between considering a large budget (e.g., long training time) with a small number of configurations, or the reverse. Hyperband attempts to solve this tradeoff by fixing the budget and exploring several configurations using a grid search method (i.e., design of experiment of all possible configurations).

The Hyperband algorithm, as described in Algorithm 2, requires two inputs: R, the maximum number of allocated resources to one configuration, and η, the proportion of throwing out configurations generated by SH algorithms at each iteration. A bracket designed a run of the SH algorithm. The input parameters control the number of brackets and allocating budgets.

Algorithm 2: Hyperband early stopping.

input: (R, η)
1 **initialization:** $s_{max} = \left\lfloor \log_\eta R \right\rfloor$, $B = (s_{max} + 1)R$
2 **for** $s \in \{s_{max}, s_{max} - 1, \ldots, 0\}$ **do**
3 $\quad n = \left\lceil \frac{B}{R} \frac{\eta^S}{S+1} \right\rceil$, $r = R\eta^{-S}$
4 $\quad$ // Start Successive Halving inner loop
5 $\quad \Omega = \text{get_hyperparameter_configuration}(n)$
6 $\quad$ **for** $i \in \{0, s\}$ **do**
7 $\quad\quad n_i = \lceil n\eta^{-1} \rceil$, $r_i = n\eta^i$
8 $\quad\quad L(\Omega) = \{\text{run_then_return_val_loss}(t, r_i) \mid t \in \Omega\}$
9 $\quad\quad \Omega = \text{top_k}(\Omega, L(\Omega), \left\lfloor \frac{n_i}{\eta} \right\rfloor)$
10 $\quad$ **end for**
11 **end for**
return configuration with the best validation loss seen so far

In lines 1–2, the algorithm starts with the more aggressive bracket with the maximum allocated resources for maximizing exploration (i.e., give some minimal time to tracking experiment and prevent premature experiment early stopping). The function "get_hyperparameter_configuration(n)" performs a hypercube sampling of the input hyperparameter and obtains an initial guess configuration. Then, each bracket (i.e., SH iterations, lines 6–9) decreases the number of remaining configurations n by the dropout η until the last bracket (s = 0). For each bracket, the function "run_then_return_val_loss(t, r_i)" (line 8) takes an input hyperparameter configuration t with the corresponding resources budget r_i, performs a training of the network and returns the associated validation loss. Finally, at line 9, the function "top_k" returns the top k training configurations.

To summarize, Hyperband is an adaptative algorithm that tracks iterations as a budget resource and associated training/validation losses. In this way, by adaptatively optimizing the average budget ratio, it could automatically stop unpromising experiments early.

2.4. Bayesian Hyperparameter Optimization Strategy (AutoML Framework)

In this paper, we suggested a combined hyperparameter strategy using Bayesian optimization and Hyperband algorithms, following an AutoML approach, as illustrated in Figure 4. Our implemented strategy follows three main steps:

1. **Configuration and model training:** This step is about automatically changing the hyperparameters of a given U-Net architecture, which we call a U-Net configuration. Then, we perform a training of the selected configuration and monitor related training metrics such as the loss function that we record and store for further use.
2. **Evaluate U-Net configuration performances:** In this step, an assessment of prediction performances of the current U-Net configuration is carried out through the computation of segmentation metrics such as IoU.
3. **Bayesian Optimization Loop (BOL):** Once a U-Net configuration is trained and evaluated, the hyperparameter optimization loop begins. In fact, when a U-Net configuration is trained, a surrogate Gaussian Process model is fitted in real time by observing the influence of input variation (U-Net hyperparameters) on the outputs, the training loss functions and the corresponding IoU score. In addition, a "watchguard" mechanism is implemented through the Hyperband algorithm in order to automatically address unpromising training experiments to save computing resources. Thus, the optimization loop performs several iterations on the overall process (steps 1–3) by varying the U-Net configurations until the target IoU score is reached (IoU ≥ 0.5).

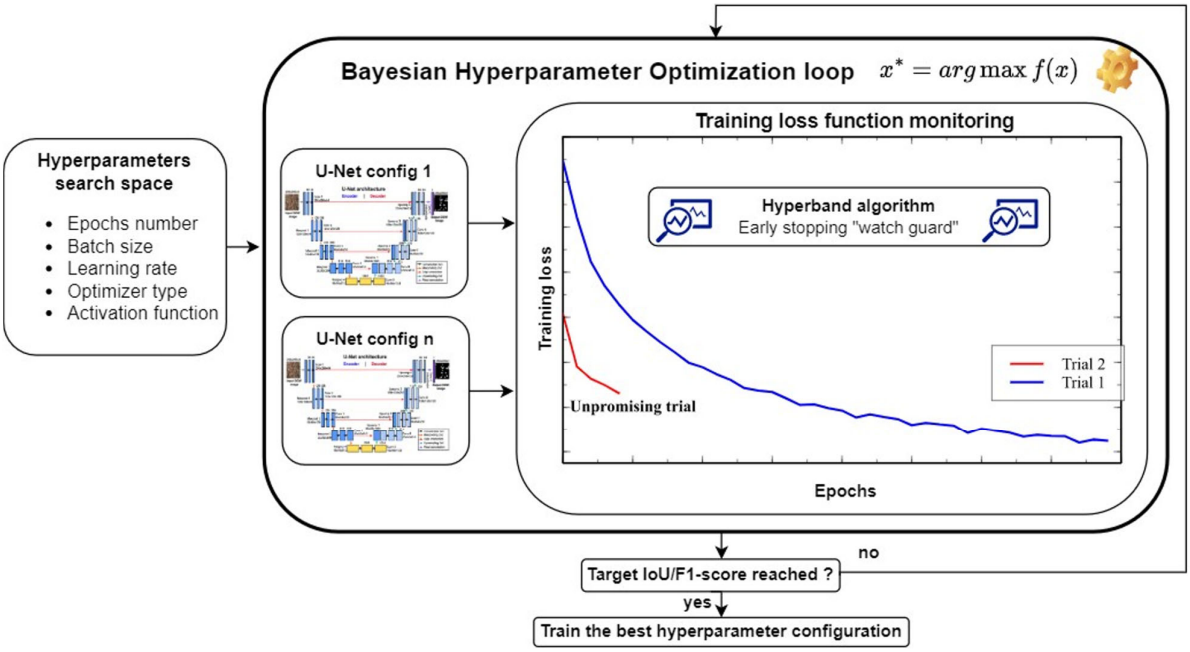

Figure 4. Bayesian hyperparameter optimization strategy.

3. Implementation Details

3.1. Data Acquisition, Hardware and Software Specifications

DGW image acquisition is realized using a numerical microscope, Keyence VHX-5000. Our training image stack is built on 153 ".tiff "images of size 256 × 256 × 3. We manually labeled each DGW image, as illustrated in Figure 5, using the open source software QuPath and Fiji [53,54]. For performing Bayesian optimization experiments and model training, the Google Colab Pro+ environment with an Nvidia Tesla P100 GPU 54 GB RAM was used. We built our deep leaning model with TensorFlow v2 [55] through the high-level API framework Keras [56]. In addition, we used the Weight and Bias (W&B) [57] python API, with its implementation of Bayesian optimization (from scikit-learn [58]) and the Hyperband algorithm. Finally, our experiments were monitored with Keras callbacks and the W&B MLOps [59,60] platform.

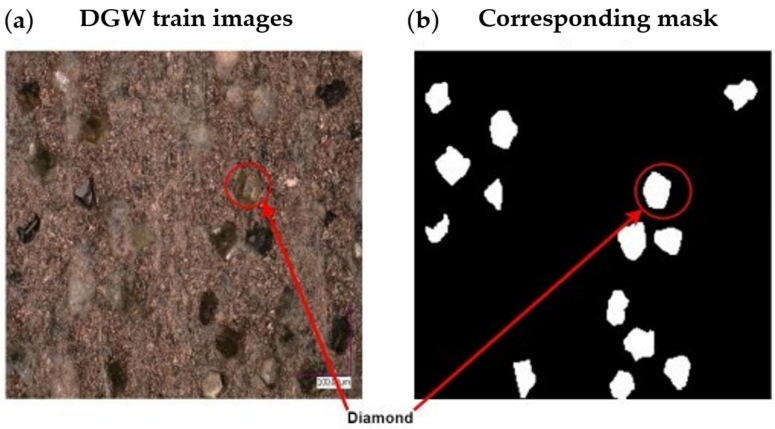

Figure 5. (**a**) DGW training images. (**b**) Corresponding binary mask.

3.2. Hyperparameter Search Space Reduction

Although our hyperparameter optimization strategy is fully automated, we must reduce the hyperparameter search space for easier convergence toward the optimization solution and to speed up computations and save resources. To overcome these shortcom-

ings, the hyperparameter search space was reduced, thanks to several assumptions that we discussed later. Tables 1 and 2 summarize the hyperparameter search space values and the combination numbers per category, respectively.

Table 1. Hyperparameter search space values (raw configuration A).

Hyperparameters	Range
Optimizer	SGD, Ftrl, Nadam, Adam, Adamax, Adagrad, Adadelta, RMSprop
Activation function	ReLU, SeLU, ELU, sigmoid, softplus, softmax, softsign, tanh, exponential
$\boldsymbol{\alpha}$	$[1 \times 10^{-1}; 1 \times 10^{-2}; 1 \times 10^{-3}; 1 \times 10^{-4}; 1 \times 10^{-5}]$
Batch size	[2; 4; 8; 16; 32; 64]
Epochs	[30; 40; 50; 60]

Table 2. Hyperparameter search space combination numbers.

Configuration	$\boldsymbol{\Lambda}$	$\boldsymbol{\xi}$ (%)	Optimizer	Activation Function	$\boldsymbol{\alpha}$	Batch Size	Epochs
A	8640	0	8	9	5	6	4
B	1296	85	3	9	2	6	4
C	144	98.3	3	1	2	6	4

The total number of hyperparameters combinations $\boldsymbol{\Pi}$ is computed by the general combinatorics formula described by Equation (10):

$$\boldsymbol{\Lambda} = \prod_{\mathbf{i}=1}^{\mathbf{r}} \mathrm{n_i} \tag{10}$$

where $\boldsymbol{\Lambda}$ is the search space total configurations number, r represents the hyperparameter variables, $\mathrm{n_i}$ represents their associated value numbers and $\boldsymbol{\xi}$ is the total average reduction factor.

In deep learning, optimizer algorithms are key systems which strongly impact the training and prediction performance of the model. Such an optimization system modifies the weights of the network through the adaptation of learning rate and the minimization of a loss function by a gradient descent algorithm [61]. This optimization procedure confers to the deep learning model the ability to learn from these errors. In addition, optimizer choice is a huge research area [62–68] and a complex question strongly correlated with the type of data to analyze. We can distinguish two main optimizer families:

- Stochastic gradient descent (SGD) with momentum (i.e., inertia added for reducing variance and increasing the convergence rate).
- Adaptative stochastic gradient descent optimizer, which automatically tunes learning rate parameter $\boldsymbol{\alpha}$ during training.

Although the SGD family could be efficient, fine-tuning the learning rate, momentum and implementation of a learning rate scheduler is mandatory. Taking all of that into account, adaptative optimizers such as Adam, Adamax and RMSprop were selected. Therefore, we could reduce the range of learning parameter $\boldsymbol{\alpha}$ between 1×10^{-3} and 1×10^{-4}. These assumptions led to reducing the hyperparameter search space combinations by 85% from 8640 in configuration A to 1296 in B. Although this is a huge search reduction improvement, it is not sufficient and should be improved. Thus, our last degree of freedom is to reduce the activation function search space. Activation functions play a key role in deep learning networks by applying nonlinear transformations to their input weights. By doing so, they trigger/activate the learning process of the network. They are used as hidden and output layers and strongly influence the prediction quality. They are generally chosen in functions of the deep learning model architecture and application field [69,70]. In this work, we use sigmoid for the output prediction layer. The sigmoid function was used in the early days of deep learning, and it remains well suited for binary

classification tasks as in our problem, the semantic segmentation of two classes: diamond grits and the background. Based on these additional assumptions, the hyperparameter search space was reduced by 13.3%, leading to a total reduction factor of 98% with 144 remaining configurations, becoming more affordable for our optimization strategy.

4. Results and Discussion

In this section, the Bayesian optimization experiments and diamond grit segmentation results are presented in detail. First, a sensibility analysis of hyperparameter optimization experiments is carried out in regard to segmentation metrics such as IoU and F1-score. Finally, a practical application of the best U-Net configurations is performed for the segmentation of diamond grits on three DGW types.

4.1. Bayesian Optimization Experiments Results

Our search space reduction effort based on the assumptions described previously drastically reduced the search space to 144 configurations. However, although the search space becomes computationally more affordable, the aim of BO algorithms is not to explore all the hyperparameter configurations. In fact, unlike brute force algorithms such as grid search, we are trying to find the best configurations in a limited number of experiments, as shown in Table 3. We performed 64 BO experiments for a total uninterrupted computing time of 22 h, as illustrated in Figure 6. For the Hyperband early stopping mechanism, we used a maximum allocated resource value R = 27, such as one epoch is equal to one allocated resource. This parameter is set according to our previous experience, where an increase in IoU was observed in the range of the 25th epochs. In addition, we set s = 2, which corresponds to two brackets of the Successive Halving (SH) algorithm. Thus, the SH algorithm will discard one-third (η = 3) of unpromising experiments in the following bands $\left[\frac{27}{3}, \frac{27}{9}\right]$ (i.e., ninth and third epochs).

Table 3. Hyperparameter reduced search space.

Optimizer	Epochs	Batch Size	α	s	R	η
Adamax, Adam, RMSprop	[30, 40, 50, 60]	[2; 4; 8; 16; 32; 64]	$[1 \times 10^{-3}; 1 \times 10^{-4}]$	2	27	3

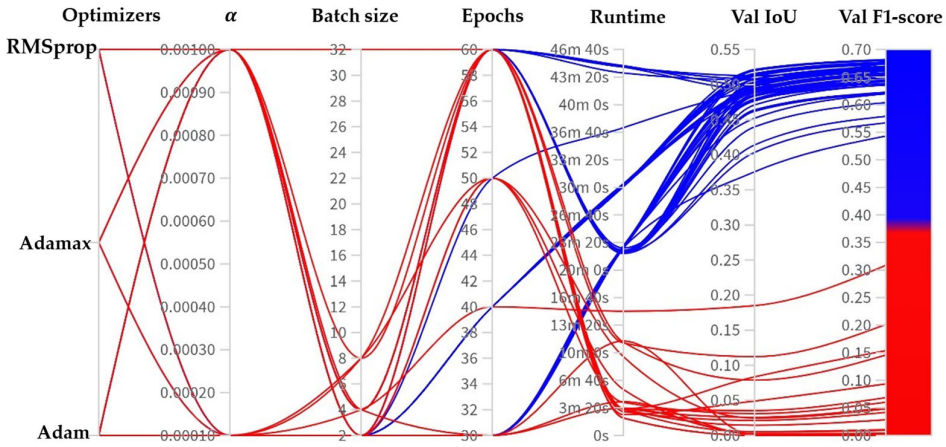

Figure 6. Parallel coordinate plots of hyperparameter optimization experiments.

Thanks to our hyperparameter optimization strategy, 24.6% of configurations with a validation IoU above 0.50 and a validation F1-score over 0.66 were identified. Furthermore, the Hyperband early stopping mechanisms identify 18% of unpromising training experiments by tracking, in a resource-based manner, the validation IoU in regard to runtime for the best previous experiments. In that way, unpromising trials are generally identified early, in the range of 2 min to 4 min after the experiments start.

The best hyperparameter configuration led to a validation IoU of 0.52 and a validation F1-score of 0.68 for a training time of 22 min 38 s. Although the BO algorithm quickly finds the best hyperparameter configurations, understanding the effects of training parameters and optimizer choices is essential from an engineering perspective. That is why we investigated several training configurations. The best/worst experimental results for each hyperparameter configuration explored by the algorithm are shown in Table 4.

Table 4. Summary of hyperparameter BO experiments.

ID	Total Exp	Total Failed	Total IoU $\geq$ 0.5	Optimizer	Epochs	Batch Size	α	Runtime	Early Stops	Val IoU	Val F1	Loss	Val Loss
A1	5	1	20%	Adamax	60	8	1×10^{-3}	11 min 8 s	False	0.11	0.20	0.61	1.37
A2				Adamax	60	4	1×10^{-3}	22 min 16 s	False	0.50	0.66	0.44	0.73
A3				Adamax	60	4	1×10^{-3}	3 min 10 s	True	0.03	0.06	0.76	1.31
A4				Adamax	40	4	1×10^{-4}	14 min 59 s	False	0.18	0.30	0.77	1.10
A5				Adamax	40	2	1×10^{-4}	30 min 22 s	False	0.48	0.64	0.56	0.75
B1	34	3	26%	Adam	60	8	1×10^{-4}	11 min 14 s	False	0.07	0.14	0.87	1.24
B2				Adam	60	4	1×10^{-3}	22 min 20 s	False	0.48	0.64	0.48	0.73
B3				Adam	60	4	1×10^{-3}	2 min 23 s	True	0.00002	0.00005	0.73	190.68
B4				Adam	60	2	1×10^{-4}	44 min 29 s	False	0.50	0.66	0.37	0.77
B5				Adam	60	2	1×10^{-3}	3 min 38 s	True	0.01	0.02	0.66	30.7
B6				Adam	50	8	1×10^{-4}	9 min 28 s	False	0.02	0.04	0.75	1.35
B7				Adam	40	2	1×10^{-4}	30 min 13 s	False	0.52	0.68	0.42	0.74
B8				Adam	30	2	1×10^{-4}	22 min 35 s	False	0.50	0.66	0.47	0.75
B9				Adam	30	2	1×10^{-4}	22 min 51 s	False	0.45	0.61	0.55	0.77
C1	25	6	24%	RMSprop	60	32	1×10^{-3}	2 min 47 s	False	0.021	0.04	0.81	1.36
C2				RMSprop	60	4	1×10^{-3}	22 min 17 s	False	0.49	0.66	0.47	0.74
C3				RMSprop	60	4	1×10^{-3}	2 min 29 s	True	0.04	0.09	0.76	1.41
C4				RMSprop	60	2	1×10^{-3}	44 min 27 s	False	0.51	0.67	0.39	0.76
C5				RMSprop	60	2	1×10^{-3}	3 min 48 s	True	0.02	0.05	0.71	1.34
C6				RMSprop	50	2	1×10^{-4}	37 min 10 s	False	0.49	0.65	0.38	0.80
C7				RMSprop	50	2	1×10^{-3}	5 min 56 s	True	0.004	0.009	0.65	1.66
C8				RMSprop	30	4	1×10^{-3}	11 min 20 s	False	0.0004	0.0008	0.55	1.796
C9				RMSprop	30	2	1×10^{-3}	22 min 38 s	False	0.52	0.68	0.47	0.73
C10				RMSprop	30	2	1×10^{-4}	22 min 37 s	False	0.50	0.66	0.48	0.77
C11				RMSprop	30	2	1×10^{-3}	4 min 11 s	True	0.001	0.003	0.66	1.9

As the selection of hyperparameter configurations is automatically computed by the BO algorithm, some experiments are missing a rigorous quantitative comparison. Thus, our study should be seen as a first qualitative attempt of hyperparameter configuration analysis of DGW image segmentations.

The Adamax optimizer configuration represents 7.8% of the experiments which have been the least explored one by the algorithm: it counts a total of five trials and one failed by Hyperband early stopping. This small experimental group led to acceptable segmentation performances. In fact, 66% of this group led to a validation IoU between 0.48 and 0.50. In addition, the batch size tends to have a linear effect on the runtime. Indeed, a decrease of 50% of the batch size led to an average half augmentation of validation IoU and F1-score. Furthermore, decreasing the learning rate coefficient alpha with epoch number seems to confer acceptable performances.

Opposite to Adamax, the Adam optimizer category has been the most explored by the algorithm, representing around 53.1% of the experiments. It counts around 26% of hyperparameter configurations with a validation IoU above 0.5 and 8.8% of failed experiments. However, with this optimizer, increasing the epoch number and batch size seems to reduce the segmentation performance and training stability. In fact, an augmentation of the epoch number tended to generate some overfitting phenomena observed with huge validation losses for experiments B3 and B5, as shown in Table 4. In addition, the best segmentation performances seem to be reached with an overall learning rate coefficient alpha of 1×10^{-4}.

The RMSprop optimizer configuration represents 39% of the BO trials. It counts around 24% of hyperparameter configurations with a validation IoU above 0.5. However, it records 24% of discarded experiments due to the largest exploration of the hyperparameter range. We clearly observed a drop in segmentation performance with batch size in this group. The batch size and epoch number seem to follow a linear trend with the runtime and confirm the same observed effect on the two other groups. Thus, the RMSprop optimizer

tends to be more stable than Adam in regard to overfitting when the training time increases. Finally, the RMSprop optimizer achieved the overall best performance, with an IoU over 0.52 and an F1-score of 0.68 with a quick training time (experiment C9, Table 4).

Thanks to this qualitative study, we could formulate the following rules about the observed effects of hyperparameter range and optimizer choice for the semantic segmentation of DGW images:

- The batch size and epoch number have a linear effect on the runtime. For the three studied optimizers, increasing the batch size and epoch number by 50% led to an augmentation of 50% of the training time.
- Increasing the batch size over a value of 4 with a high epoch number led to a significative reduction in segmentation performance.
- The Adam optimizer tends to be more sensitive to overfitting with large epoch numbers.
- Decreasing the learning rate coefficient alpha in conjunction with the epoch number exhibits the best overall performances for Adamax and Adam optimizers.
- With RMSprop optimizer, no evident patterns are seen for fine-tuning the learning rate alpha.
- Adam and RMSprop optimizers exhibit better segmentation performances with minimal training time compared to Adamax.
- The overall best configuration is reached by RMSprop optimizer with a reduced training time of 25% compared to Adam for equivalent segmentation performances.

Finally, the best hypermeter configurations found by our BO optimization strategy are summarized in Table 5.

Table 5. Best hyperparameter configurations found by our BO optimization strategy.

Optimizer	Epochs	Batch Size	α	Runtime	Val IoU	Val F1-Score
Adamax	60	4	1×10^{-3}	22 min 16 s	0.50	0.66
Adam	40	2	1×10^{-4}	30 min 13 s	0.52	0.68
RMSprop	30	2	1×10^{-3}	22 min 38 s	0.52	0.68

Finally, it is of importance to highlight some limitations of using pre-trained ResNet34 backbone weights on the encoder part of the network. In fact, although transfer learning techniques allow us to train U-Net deep learning architecture on a small dataset, it prevents us from fully using our AutoML framework. Indeed, input images must be $256 \times 256 \times 3$ due to transfer learning usage, which leads to a lack of flexibility in regard to input images size. In addition, we are not able to fully leverage the power of Bayesian optimization algorithms for neural architecture search (number of CNN layers, size of convolution kernel and so on) due to the inherent constraints of using the ResNet34 weights on the encoder part.

4.2. Diamond Grits Segmentation Results

Thanks to our optimization strategy and the previous experimental study, we found the best hyperparameter configurations, shown in Table 5. Based on these, we trained three diamond grit segmentation models, as shown in Figures 7–10. We trained our models on a heterogeneous stack of 153 DGW images. We deliberately chose to test our model on three types of DGW images:

- Type I: Standard metallic DGW images taken at high magnification.
- Type II: Metallic DGW images taken at low magnification.
- Type III: Electroplated DGW images.

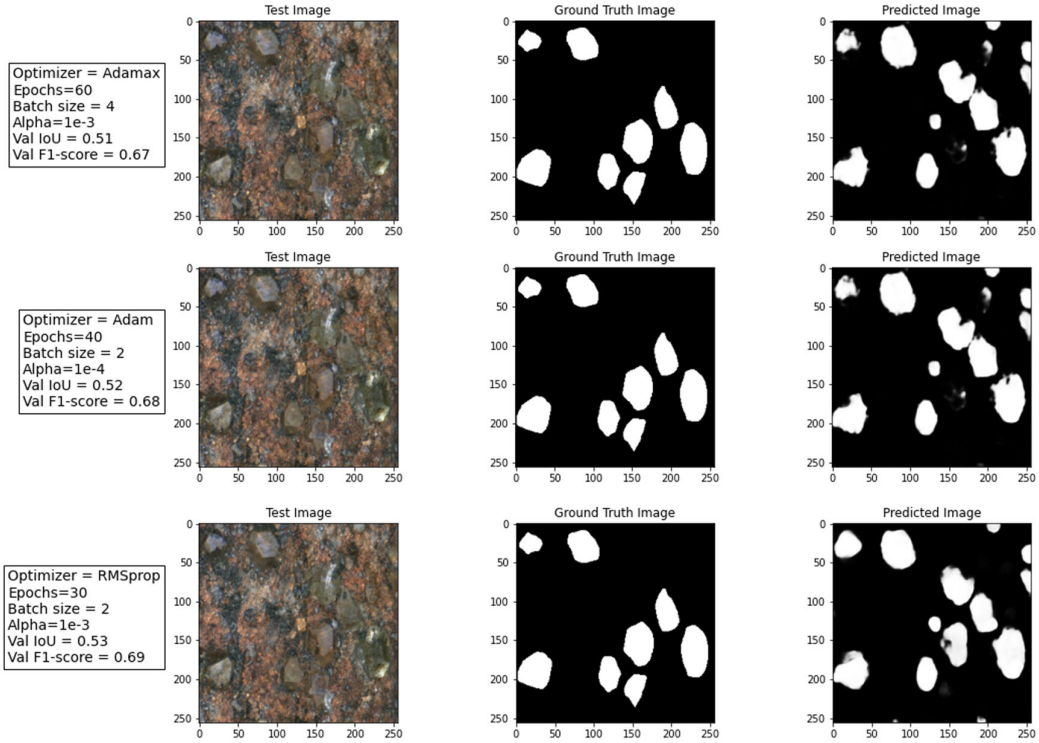

Figure 7. DGW test images of type I, associated mask and prediction for each optimizer.

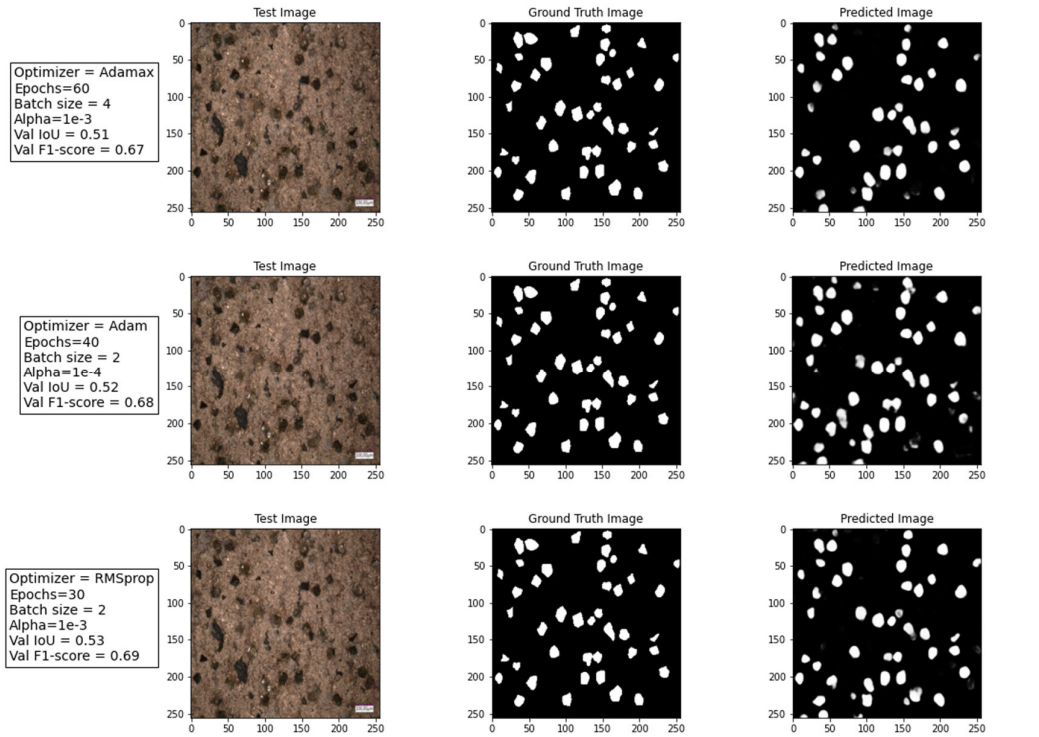

Figure 8. DGW test images of type II, associated mask and prediction for each optimizer.

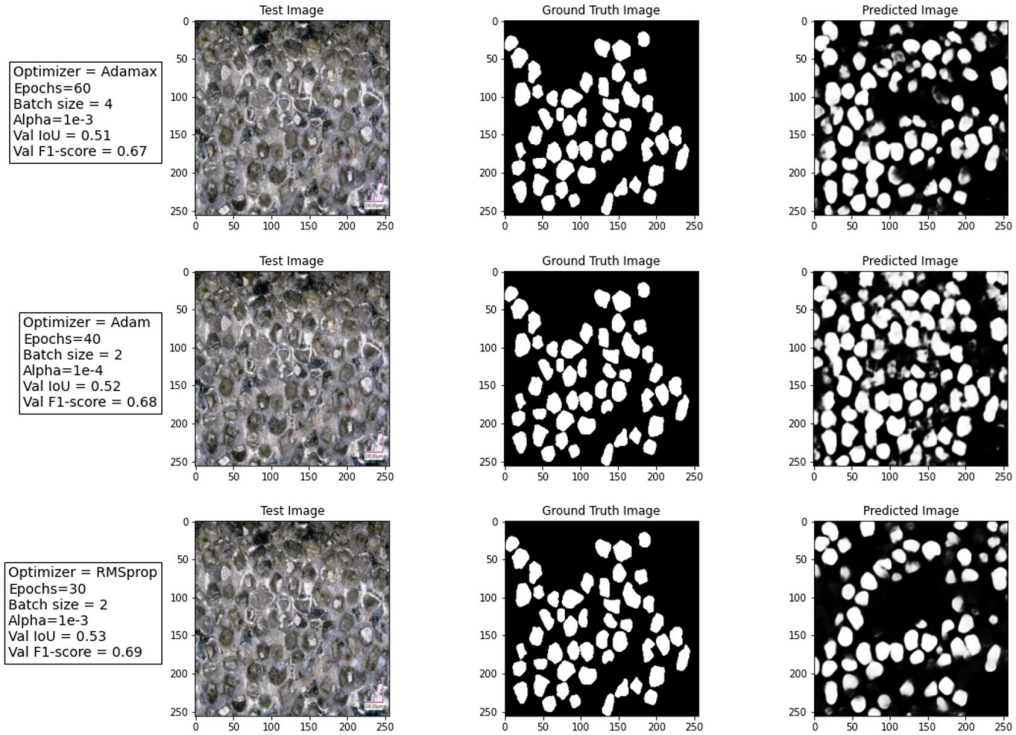

Figure 9. DGW test images of type III, associated mask and prediction for each optimizer.

In addition, it is worth noting that deep learning experiment reproducibility is a complex problem [71–74]. Indeed, previous segmentation results should not be seen as a rule of thumb but as automatic methodology and guidelines toward the best segmentation performance for our dataset. Furthermore, it is important to notice that we have not labeled all the diamond grits on the ground truth. Each model was tested on different types of DGW images for performance estimation purposes, and the results are summarized in Figure 10.

Although the reproducibility of deep learning experiments is a complex problem, the results conform to the best hyperparameter configurations found previously with our BO experiments. In fact, they are even better, with a validation IoU in the range of [0.51–0.53], and of [0.67–0.69] for the corresponding F1-score. On type I DGW images, the models performed relatively well, with an overall validation IOU of 0.52 and F1-score of 0.68, as shown in Figure 8. The RMSprop optimizer exhibits the best performance, detecting a labeled diamond grit not seen by the others. In addition, smoother segmentation with better edge detection was achieved by the RMSprop optimizer. On the type II images shown in Figure 9, some subtle differences began to appear in the segmentation quality, although the IoU/F1 score denotes the RMSprop configuration as the best. In fact, Adam detects more diamond grits than the two others, whereas RMSprop tends to better preserve the diamond grits' shape. Although our models reached some overall acceptable segmentation scores, the type III images shown in Figure 10 highlight some limitations of only using IoU/F1-score for partially annotated ground truth. In fact, the RMSprop configuration is the worst one in regard to diamond grit detection. The best configuration in this case is Adam, which segments practically all the diamond grits. This highlights an important point about the selected segmentation metrics for evaluating model performances. For type III images, we found that IoU/F1-score are not well suited for evaluating the model performance, as the labels/real diamond numbers are strongly unbalanced. In such extreme cases of partially annotated labels, the IoU/F1-score should be used in conjunction with other metrics such as Dice, Hausdorff distance, partially annotated algorithms [75–77] and visual evaluation.

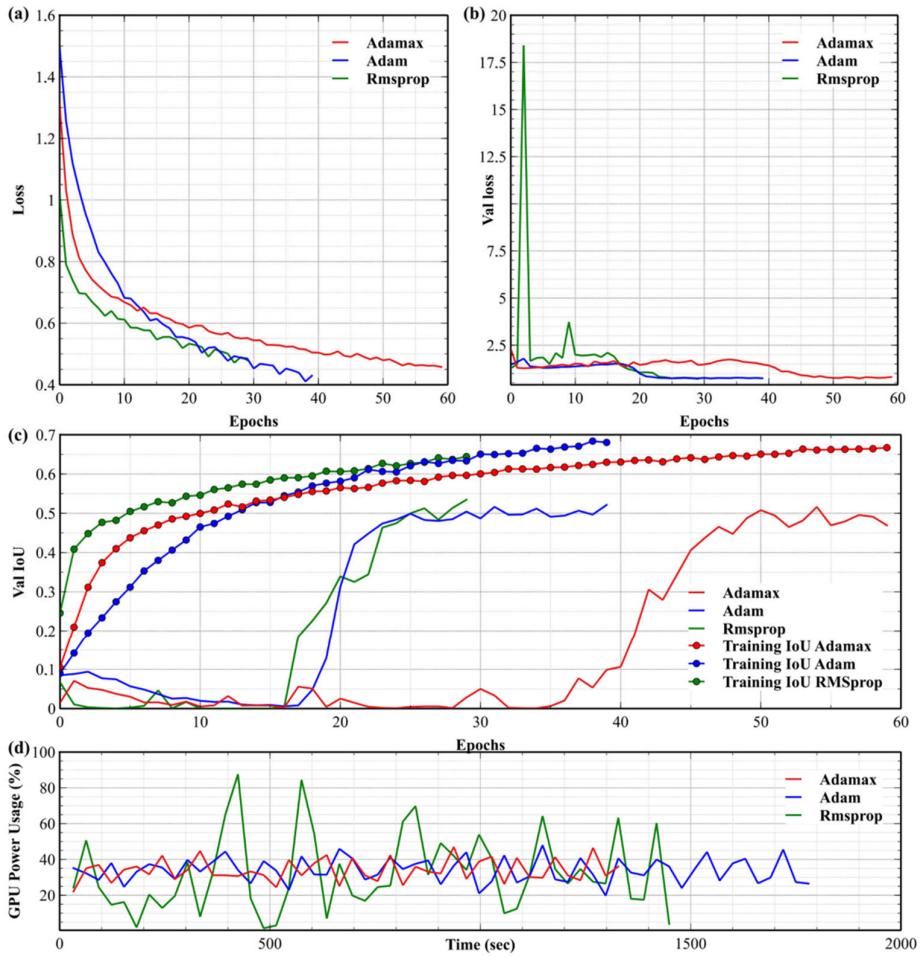

Figure 10. Evolution of training metrics for the three best optimizer configurations: (**a**) training loss function evolution; (**b**) validation loss during training; (**c**) training and validation IoU segmentation metrics; (**d**) GPU power consumption during training.

Regarding the loss evolution, RMSprop exhibits some overfitting behavior until the 20th epoch, as shown in Figure 10b. Adamax also has some predisposition to overfitting in the range of [20–50] epochs. Thus, the Adam optimizer tends to quickly converge toward a loss minimum and is more stable than the two others. In fact, Adam and Adamax converge 50% quicker than RMSprop by following a growing trend before 20 epochs, as shown in Figure 10c. On the other hand, RMSprop's validation IoU tends to decrease from the 55th epoch.

On the computing resources side, although Adamax has a bigger training epoch number, it is the cheapest to train in regard to computing time and GPU power, as shown in Figure 10d. It takes around 22 min 30 s to be trained for an average GPU power consumption of 35%. Adam has a bit longer training time than the two others, with around 30 min. In the end, RMSprop is the most GPU-demanding, with GPU peak usage in the range of [80–60%]. This could be explained by its tendency for model overfitting in the early training stages.

Although the best U-Net configurations led to overall satisfactory diamond grit segmentation performances, some limitations of the presented methods cannot be ignored. Using a small training dataset amounts to losing the ability of the network to generalize, which leads to segmentation issues for DGW images that are scarcely represented, such as type III.

5. Conclusions

The configuration and calibration of deep learning computer vision remains a challenging task, even for subject experts. In this respect, the Bayesian optimization results demonstrate the ability to automatically find the best configuration of such models by leveraging an AutoML approach. Indeed, using the Bayesian optimization loop in conjunction with the Hyperband early stopping mechanism allows an effective scanning of complex search spaces in a cost-saving way. By using the proposed hyperparameter optimization methodology, it becomes possible to leverage U-Net within the industrial environment of DGW manufacturing.

Thus, to the best of our knowledge, this article is the first published work of using U-Net deep learning architecture for semantic segmentation of diamond grits of DGW materials. Considering the small size of the training dataset, acceptable segmentation performances were reached, which largely outperformed time-consuming and error-prone manual methods.

The availability of the methods concerning the lack of flexibility of the fixed image input format and U-Net architecture due to the ResNet34 backbone would be improved in future works. A potential improvement lies in removing the pre-trained weights on the encoder part by using a Generative Adversarial Network (GAN) [78] to generate simulated DGW surface images. In that respect, it should be possible to substantially increase the training dataset size and apply the presented AutoML methodology on the neural architecture for better segmentation results.

Finally, this work is the first step of a broader research effort of making deep learning computer vision algorithms easier to use for scientists, engineers and programmers.

Author Contributions: Conceptualization, D.S.; methodology, D.S.; software, D.S.; validation, D.S., P.B. and F.B.; formal analysis, D.S.; investigation, D.S.; resources, D.S., E.B. and J.C.; data curation, D.S. and J.T.; writing—original draft preparation, D.S.; writing—review and editing, D.S., P.B., A.B. and F.B; visualization, D.S.; supervision, D.S., P.B. and F.B.; project administration, D.S., P.B. and F.B.; funding acquisition, F.B., D.S., J.T. and P.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the French CIFRE fellowship, ANRT support, Université de Bourgogne Franche-Comté and DIAMATEC company.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Denkena, B.; Bergmann, B.; Lang, R. Influence of the Powder Metallurgy Route on the Mechanical Properties of Cu–Cr–Diamond Composites. *SN Appl. Sci.* **2022**, *4*, 161. [CrossRef]
2. Nguyen, A.; Butler, D. Correlation of Grinding Wheel Topography and Grinding Performance: A Study from a Viewpoint of Three-Dimensional Surface Characterisation. *J. Mater. Process. Technol.* **2008**, *208*, 14–23. [CrossRef]
3. Choudhary, A.; Babu, N.R. Influence of 3D Topography on Tribological Behavior of Grinding Wheel. *Procedia Manuf.* **2020**, *48*, 533–540. [CrossRef]
4. Bazan, A.; Kawalec, A.; Rydzak, T.; Kubik, P.; Olko, A. Determination of Selected Texture Features on a Single-Layer Grinding Wheel Active Surface for Tracking Their Changes as a Result of Wear. *Materials* **2020**, *14*, 6. [CrossRef]
5. Ye, R.; Jiang, X.; Blunt, L.; Cui, C.; Yu, Q. The Application of 3D-Motif Analysis to Characterize Diamond Grinding Wheel Topography. *Measurement* **2016**, *77*, 73–79. [CrossRef]
6. Caraguay, S.J.; Boaron, A.; Weingaertner, W.L.; Bordin, F.M.; Xavier, F.A. Wear Assessment of Microcrystalline and Electrofused Aluminum Oxide Grinding Wheels by Multi-Sensor Monitoring Technique. *J. Manuf. Process.* **2022**, *80*, 141–151. [CrossRef]
7. Garcia-Garcia, A.; Orts-Escolano, S.; Oprea, S.; Villena-Martinez, V.; Garcia-Rodriguez, J. A Review on Deep Learning Techniques Applied to Semantic Segmentation. *arXiv* **2017**, arXiv:1704.06857.
8. Guo, Y.; Liu, Y.; Georgiou, T.; Lew, M.S. A Review of Semantic Segmentation Using Deep Neural Networks. *Int. J. Multimed. Inf. Retr.* **2018**, *7*, 87–93. [CrossRef]

9. Thoma, M. A Survey of Semantic Segmentation. *arXiv* **2016**, arXiv:1602.06541v2.
10. Siam, M.; Elkerdawy, S.; Jagersand, M.; Yogamani, S. Deep Semantic Segmentation for Automated Driving: Taxonomy, Roadmap and Challenges. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 1–8.
11. Fu, Y.; Lei, Y.; Wang, T.; Curran, W.J.; Liu, T.; Yang, X. A Review of Deep Learning Based Methods for Medical Image Multi-Organ Segmentation. *Phys. Med.* **2021**, *85*, 107–122. [CrossRef]
12. Navab, N.; Hornegger, J.; Wells, W.M.; Frangi, A.F. (Eds.) *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Proceedings of the 18th International Conference, Munich, Germany, 5–9 October 2015*; Springer: Cham, Switzerland, 2015.
13. Khaleghi, N.; Rezaii, T.Y.; Beheshti, S.; Meshgini, S.; Sheykhivand, S.; Danishvar, S. Visual Saliency and Image Reconstruction from EEG Signals via an Effective Geometric Deep Network-Based Generative Adversarial Network. *Electronics* **2022**, *11*, 3637. [CrossRef]
14. Sheykhivand, S.; Yousefi Rezaii, T.; Naderi, A.; Romooz, N. Comparison between Different Methods of Feature Extraction in BCI Systems Based on SSVEP. *Int. J. Ind. Math.* **2017**, *9*, 341–347.
15. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
16. Jogin, M.; Mohana; Madhulika, M.S.; Divya, G.D.; Meghana, R.K.; Apoorva, S. Feature Extraction Using Convolution Neural Networks (CNN) and Deep Learning. In Proceedings of the 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), Bangalore, India, 18–19 May 2018; pp. 2319–2323.
17. LeCun, Y.; Bengio, Y.; Hinton, G. Deep Learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
18. Gu, J.; Wang, Z.; Kuen, J.; Ma, L.; Shahroudy, A.; Shuai, B.; Liu, T.; Wang, X.; Wang, L.; Wang, G.; et al. Recent Advances in Convolutional Neural Networks. *arXiv* **2017**, arXiv:1512.07108. [CrossRef]
19. Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* **2017**, *29*, 2352–2449. [CrossRef]
20. Selvan, R.; Bhagwat, N.; Anthony, L.F.W.; Kanding, B.; Dam, E.B. Carbon Footprint of Selecting and Training Deep Learning Models for Medical Image Analysis. In *Medical Image Computing and Computer Assisted Intervention—MICCAI 2022*; Springer: Cham, Switzerland, 2022; Volume 13435, pp. 506–516.
21. Xu, J.; Zhou, W.; Fu, Z.; Zhou, H.; Li, L. A Survey on Green Deep Learning. *arXiv* **2021**, arXiv:2111.05193.
22. Oquab, M.; Bottou, L.; Laptev, I.; Sivic, J. Learning and Transferring Mid-Level Image Representations Using Convolutional Neural Networks. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1717–1724.
23. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A Survey of Transfer Learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
24. Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; He, Q. A Comprehensive Survey on Transfer Learning. *arXiv* **2020**, arXiv:1911.02685. [CrossRef]
25. Elharrouss, O.; Akbari, Y.; Almaadeed, N.; Al-Maadeed, S. Backbones-Review: Feature Extraction Networks for Deep Learning and Deep Reinforcement Learning Approaches. *arXiv* **2022**, arXiv:2206.08016.
26. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
27. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2009; Curran Associates, Inc.: Red Hook, NY, USA, 2012; Volume 25.
28. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556.
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *arXiv* **2015**, arXiv:1512.03385.
30. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going Deeper with Convolutions. *arXiv* **2014**, arXiv:1409.4842.
31. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015, Munich, Germany, 5–9 October 2015; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Springer International Publishing: Cham, Switzerland, 2015; pp. 234–241.
32. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA; pp. 2818–2826.
33. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
34. Tan, M.; Le, Q.V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv* **2020**, arXiv:1905.11946.
35. He, X.; Zhao, K.; Chu, X. AutoML: A Survey of the State-of-the-Art. *Knowl. Based Syst.* **2021**, *212*, 106622. [CrossRef]
36. Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R.P.; de Freitas, N. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proc. IEEE* **2016**, *104*, 148–175. [CrossRef]
37. Mockus, J. Application of Bayesian Approach to Numerical Methods of Global and Stochastic Optimization. *J. Glob. Optim.* **1994**, *4*, 347–365. [CrossRef]

38. Malu, M.; Dasarathy, G.; Spanias, A. Bayesian Optimization in High-Dimensional Spaces: A Brief Survey. In Proceedings of the 2021 12th International Conference on Information, Intelligence, Systems & Applications (IISA), Chania Crete, Greece, 12–14 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8.
39. Turner, R.; Eriksson, D.; McCourt, M.; Kiili, J.; Laaksonen, E.; Xu, Z.; Guyon, I. Bayesian Optimization Is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. *arXiv* **2021**, arXiv:2104.10201.
40. Lei, B.; Kirk, T.Q.; Bhattacharya, A.; Pati, D.; Qian, X.; Arroyave, R.; Mallick, B.K. Bayesian Optimization with Adaptive Surrogate Models for Automated Experimental Design. *npj Comput. Mater.* **2021**, *7*, 194. [CrossRef]
41. Li, L.; Jamieson, K.; DeSalvo, G.; Rostamizadeh, A.; Talwalkar, A. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *arXiv* **2018**, arXiv:1603.06560.
42. Harouni, M.; Baghmaleki, H.Y. Color Image Segmentation Metrics. *arXiv* **2020**, arXiv:2010.09907.
43. Müller, D.; Soto-Rey, I.; Kramer, F. Towards a Guideline for Evaluation Metrics in Medical Image Segmentation. *arXiv* **2022**, arXiv:2202.05273. [CrossRef] [PubMed]
44. Bergstra, J.; Yamins, D.; Cox, D.D. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *arXiv* **2012**, arXiv:1209.5111.
45. Bartz, E.; Zaefferer, M.; Mersmann, O.; Bartz-Beielstein, T. Experimental Investigation and Evaluation of Model-Based Hyperparameter Optimization. *arXiv* **2021**, arXiv:2107.08761.
46. Feurer, M.; Hutter, F. Hyperparameter Optimization. In *Automated Machine Learning: Methods, Systems, Challenges*; Hutter, F., Kotthoff, L., Vanschoren, J., Eds.; The Springer Series on Challenges in Machine Learning; Springer International Publishing: Cham, Switzerland, 2019; pp. 3–33. ISBN 978-3-030-05318-5.
47. Li, L.; Jamieson, K.; Rostamizadeh, A.; Gonina, E.; Hardt, M.; Recht, B.; Talwalkar, A. A System for Massively Parallel Hyperparameter Tuning. *arXiv* **2020**, arXiv:1810.05934.
48. Morales-Hernández, A.; Nieuwenhuyse, I.; Rojas Gonzalez, S. A Survey on Multi-Objective Hyperparameter Optimization Algorithms for Machine Learning. *arXiv* **2021**, arXiv:2111.13755.
49. Yang, L.; Shami, A. On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice. *arXiv* **2020**, arXiv:2007.15745. [CrossRef]
50. Lu, Q.; Polyzos, K.D.; Li, B.; Giannakis, G. Surrogate Modeling for Bayesian Optimization beyond a Single Gaussian Process. *arXiv* **2022**, arXiv:2205.14090.
51. Rasmussen, C.E.; Williams, C.K.I. *Gaussian Processes for Machine Learning*; Adaptive Computation and Machine Learning; MIT Press: Cambridge, MA, USA, 2006; ISBN 978-0-262-18253-9.
52. Jamieson, K.; Talwalkar, A. Non-Stochastic Best Arm Identification and Hyperparameter Optimization. *arXiv* **2015**, arXiv:1502.07943.
53. Bankhead, P.; Loughrey, M.B.; Fernández, J.A.; Dombrowski, Y.; McArt, D.G.; Dunne, P.D.; McQuaid, S.; Gray, R.T.; Murray, L.J.; Coleman, H.G.; et al. QuPath: Open Source Software for Digital Pathology Image Analysis. *Sci. Rep.* **2017**, *7*, 16878. [CrossRef]
54. Schindelin, J.; Arganda-Carreras, I.; Frise, E.; Kaynig, V.; Longair, M.; Pietzsch, T.; Preibisch, S.; Rueden, C.; Saalfeld, S.; Schmid, B.; et al. Fiji: An Open-Source Platform for Biological-Image Analysis. *Nat. Methods* **2012**, *9*, 676–682. [CrossRef] [PubMed]
55. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv* **2016**, arXiv:1603.04467.
56. Keras: Deep Learning for Humans 2022. Available online: https://keras.io/ (accessed on 3 December 2022).
57. Biewald, L. Experiment Tracking with Weights and Biases 2020. Available online: https://wandb.ai/site (accessed on 3 December 2022).
58. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
59. Zhao, Y. Machine Learning in Production: A Literature Review. Available online: https://staff.fnwi.uva.nl/a.s.z.belloum/LiteratureStudies/Reports/2021-LiteratureStudy-report-Yizhen.pdf (accessed on 3 December 2022).
60. Hewage, N.; Meedeniya, D. Machine Learning Operations: A Survey on MLOps Tool Support. *arXiv* **2022**, arXiv:2202.10169.
61. Ruder, S. An Overview of Gradient Descent Optimization Algorithms. *arXiv* **2016**, arXiv:1609.04747. [CrossRef]
62. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2017**, arXiv:1412.6980.
63. Tieleman, T.; Hinton, G. Lecture 6.5-Rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. *COURSERA Neural Netw. Mach. Learn.* **2012**, *4*, 26–31.
64. Zeiler, M. ADADELTA: An Adaptive Learning Rate Method. *arXiv* **2012**, arXiv:1212.5701.
65. Gupta, A.; Ramanath, R.; Shi, J.; Keerthi, S.S. Adam vs. SGD: Closing the Generalization Gap on Image Classification. Available online: https://opt-ml.org/papers/2021/paper53.pdf (accessed on 3 December 2022).
66. Sun, S.; Cao, Z.; Zhu, H.; Zhao, J. A Survey of Optimization Methods from a Machine Learning Perspective. *IEEE Trans. Cybern.* **2020**, *50*, 3668–3681. [CrossRef]
67. Reddi, S.J.; Kale, S.; Kumar, S. On the Convergence of Adam and Beyond. *arXiv* **2018**, arXiv:1904.09237.
68. Sun, R. Optimization for Deep Learning: Theory and Algorithms. *arXiv* **2019**, arXiv:1912.08957.
69. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. *arXiv* **2022**, arXiv:2109.14545. [CrossRef]

70. Szandała, T. Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. In *Bio-Inspired Neurocomputing*; Bhoi, A.K., Mallick, P.K., Liu, C.-M., Balas, V.E., Eds.; Studies in Computational Intelligence; Springer: Singapore, 2021; Volume 903, pp. 203–224. ISBN 9789811554940.
71. Lynnerup, N.A.; Nolling, L.; Hasle, R.; Hallam, J. A Survey on Reproducibility by Evaluating Deep Reinforcement Learning Algorithms on Real-World Robots. *arXiv* **2019**, arXiv:1909.03772.
72. Isdahl, R.; Gundersen, O.E. Out-of-the-Box Reproducibility: A Survey of Machine Learning Platforms. In Proceedings of the 2019 15th International Conference on eScience (eScience), San Diego, CA, USA, 24–27 September 2019.
73. Liu, C.; Gao, C.; Xia, X.; Lo, D.; Grundy, J.; Yang, X. On the Replicability and Reproducibility of Deep Learning in Software Engineering. *ACM Trans. Softw. Eng. Methodol.* **2022**, *31*, 1–46. [CrossRef]
74. Chen, B.; Wen, M.; Shi, Y.; Lin, D.; Rajbahadur, G.K.; Jiang, Z.M. Towards Training Reproducible Deep Learning Models. In Proceedings of the 44th International Conference on Software Engineering, Pittsburgh, PA, USA, 21 May 2022; pp. 2202–2214.
75. Koch, L.M.; Rajchl, M.; Bai, W.; Baumgartner, C.F.; Tong, T.; Passerat-Palmbach, J.; Aljabar, P.; Rueckert, D. Multi-Atlas Segmentation Using Partially Annotated Data: Methods and Annotation Strategies. *arXiv* **2016**, arXiv:1605.00029. [CrossRef] [PubMed]
76. Semantic Segmentation with Incomplete Annotations. Available online: https://uoguelph-mlrg.github.io/CFIW/slides/SMILE_DeepVision.pdf (accessed on 3 December 2022).
77. Martinez, N.; Sapiro, G.; Tannenbaum, A.; Hollmann, T.J.; Nadeem, S. ImPartial: Partial Annotations for Cell Instance Segmentation. *bioRxiv* **2021**. [CrossRef]
78. Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; Bengio, Y. Generative Adversarial Networks. *Commun. ACM* **2020**, *63*, 139–144. [CrossRef]