Multi-task shape optimization using a 3D point cloud autoencoder as unified representation

de Jesus de Araujo Rios, T.; Stein, B. van; Bäck, T.H.W.; Sendhoff, B.; Menzel, S.

Multitask Shape Optimization Using a 3-D Point Cloud Autoencoder as Unified Representation

Thiago Rios, *Member, IEEE*, Bas van Stein, Thomas Bäck, *Senior Member, IEEE*, Bernhard Sendhoff, *Senior Member, IEEE*, and Stefan Menzel

***Abstract*—The choice of design representations, as of search operators, is central to the performance of evolutionary optimization algorithms, in particular, for multitask problems. The multitask approach pushes further the parallelization aspect of these algorithms by solving simultaneously multiple optimization tasks using a single population. During the search, the operators implicitly transfer knowledge between solutions to the offspring, taking advantage of potential synergies between problems to drive the solutions to optimality. Nevertheless, in order to operate on the individuals, the design space of each task has to be mapped to a common search space, which is challenging in engineering cases without clear semantic overlap between parameters. Here, we apply a 3-D point cloud autoencoder to map the representations from the Cartesian to a unified design representation: the latent space of the autoencoder. The transfer of latent space features between design representations allows the reconstruction of shapes with interpolated characteristics and maintenance of common parts, which potentially improves the performance of the designs in one or more tasks during the optimization. Compared to traditional representations for shape optimization, such as free-form deformation, the latent representation enables more representative design modifications, while keeping the baseline characteristics of the learned classes of objects. We demonstrate the efficiency of our approach in an optimization scenario where we minimize the aerodynamic drag of two different car shapes with common underbodies for cost-efficient vehicle platform design.**

***Index Terms*—Automotive engineering, commonality, evolutionary multitask optimization, point cloud autoencoder.**

I. Introduction

CHALLENGING engineering optimization problems often comprise computationally expensive objective functions, for which the derivatives are mostly unavailable. Evolutionary algorithms (EAs) tackle some of these challenges by searching for solutions in a nature-inspired and parallelized way without the need of mathematically formulated objective functions or even derivatives. Thereby, they improve the feasibility of design exploration in complex and multimodal optimization landscapes.

Multifactorial EAs (MFEAs) exploit the intrinsic parallelism of EAs further by tackling multiple tasks with a single population of designs. The potential benefits of MFEAs are twofold. First, the algorithms reduce the computational effort by evaluating each solution only on one of the several tasks, which yields a task-dependent fitness metric. Second, the multitask approach leverages synergies between tasks by randomly assigning tasks in the population. As a result, solutions that perform well on several tasks in different generations have a selective advantage, and the information about these solutions is transferred to other individuals. This is different from multiobjective optimization, where every solution is evaluated on all tasks in every generation [1]. However, in order to perform different tasks sequentially and to allow information transfer, it is important for MFEAs to represent the solutions in a way that is suitable for all tasks in the set.

In principle, defining a unified representation for the designs requires neither prior information on the similarity between tasks nor a clear semantic overlap between the different task-specific design spaces. Nevertheless, it is intuitive that only specific matches between task-specific parameters might favor the synergies between tasks. In particular for geometric design optimization in complex, real-world applications, identifying the representations that enable task synergies when mapped to the unified search space is challenging [2]. Hence, a more generic, yet systematic, approach to automatically identify geometric design features and map task-specific representations to a unified search space is currently not available. However, it has significant potential to improve the performance of MFEAs.

Recently, geometric deep learning algorithms have been applied to non-Euclidean (unstructured) data representations, such as voxels, graphs, and 3-D point clouds [3]–[5], showing impressive performance in shape classification, reconstruction, and segmentation tasks. Particularly, for engineering shape design, autoencoder networks can learn compact design representations of computer-aided engineering (CAE) models by compressing the data through a bottleneck layer. The obtained low-dimensional feature space, the so-called latent representation, allows a designer to perform shape operations and generate a diverse set of shapes, not necessarily observed in

Manuscript received December 15, 2020; revised April 1, 2021; accepted May 20, 2021. Date of publication June 3, 2021; date of current version March 31, 2022. This work was supported by the European Union's Horizon 2020 Research and Innovation Programme under Grant 766186 (ECOLE). *(Corresponding author: Thiago Rios.)*

Thiago Rios, Bernhard Sendhoff, and Stefan Menzel are with the Honda Research Institute Europe, 63073 Offenbach, Germany (e-mail: thiago.rios@honda-ri.de; bernhard.sendhoff@honda-ri.de; stefan.menzel@honda-ri.de).

Bas van Stein and Thomas Bäck are with the Department of Computer Science, Leiden Institute of Advanced Computer Science, 2333 CA Leiden, The Netherlands (e-mail: b.van.stein@liacs.leidenuniv.nl; t.h.w.baeck@liacs.leidenuniv.nl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TEVC.2021.3086308.

Digital Object Identifier 10.1109/TEVC.2021.3086308

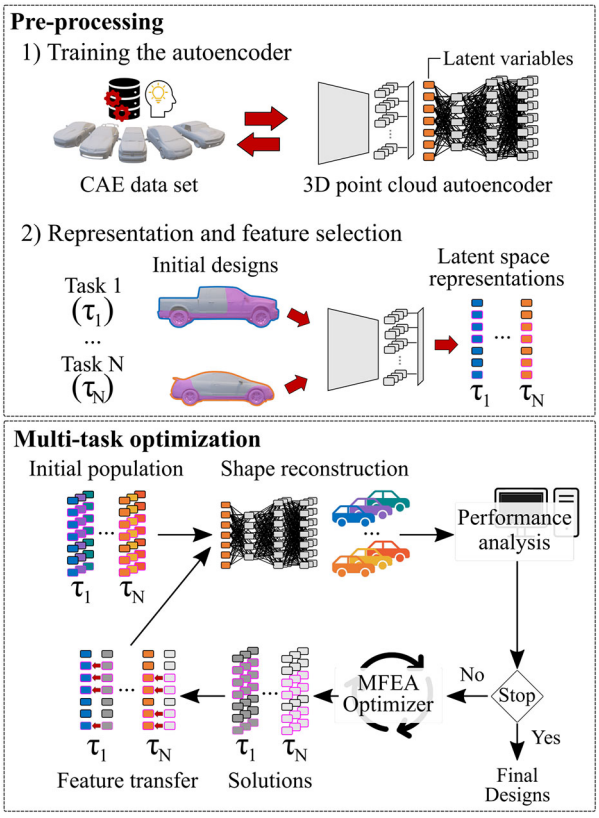

Fig. 1. Proposed workflow of a multitask optimization using the latent space of an autoencoder as unified search space.

the training data [6]–[8]. Furthermore, since these autoencoder networks typically learn exclusively from geometric data, the latent representations are domain independent and only carry the bias intrinsic to the geometric data generation. Therefore, the design space learned by the autoencoder network unifies the representation of a set of shapes, which might belong to different semantic classes, and potentially enhances the transfer of knowledge between problems in different domains.

In this article, we use and adapt the latent space representation of CAE models learned with a 3-D point cloud autoenoder to define a unified search space for MFEAs in shape optimization problems (Fig. 1). Our approach comprises a preprocessing phase and the multitask optimization searching in the latent space. In the preprocessing phase, the autoencoder is trained on a data set of CAE models generated randomly or taken from previous design optimizations. The learned latent space is considered as basis for representing the initial designs for each task. The engineer decides which variables of each representation (in the latent space) should be modified during the optimization based on the visualization of the latent features, which reveals the mapping between latent variables and geometric characteristics of the designs. In the optimization step, the MFEA generates solutions in the latent space, from which the subspace selected by the designer is transferred to the representations of the initial design of the corresponding tasks. Since the optimizer shuffles some of the task assignments, it allows to share geometric characteristics that evolved to solve particular tasks with other individuals in the population. By mapping the search space to the CAE representation, which was originally generated by the designer, the trained decoder recovers the 3-D point clouds from the representations in the latent space. The recovered models are postprocessed and forwarded to downstream tasks, such as computer simulations, to calculate the performance of the designs.

The main contribution of our work is the learning-based (autoencoder) and domain-independent unified search space for multitask evolutionary optimization of 3-D shapes and designs. Instead of using a superset of all task-specific representations or choosing a task-agnostic representation (e.g., binary coding), we propose to learn the features that optimally describe the available geometric data in order to increase the synergy between optimization tasks. Furthermore, the transfer of features enables the algorithm to exploit commonalities between the designs, which can be used to enforce modularity, e.g., to address design constraints from maintenance and manufacturing. Finally, by learning the representation with the autoencoder network, it is also possible to include "out of the box" shapes, i.e., shapes that defy to a certain extent engineering standards like nature-inspired shapes often do. This opens up search space opportunities for the MFEA.

The remainder of this article is organized as follows. First, we review both the work related to multitask optimization algorithms and to the architecture of the proposed 3-D point cloud autoencoder in Section II. In Section III, we present the experimental settings for our analyses, which include the hyperparameters for training the 3-D point cloud autoencoder and the optimization algorithm. Here, we also verify our implementation using benchmark functions and we compare our results to the literature. In Section IV, we formulate a vehicle aerodynamic optimization problem as a multitask optimization with different car shapes. In our experiments, we enabled the modification of the latent features that mapped the underbody of the car shapes targeting optimal geometries with similar underbody structure. Finally, we conclude this article in Section V by summarizing our main findings and providing an outlook on future research potentials.

II. Related Work

In this section, we present the work related to multitask evolutionary optimization algorithms and geometric deep learning.

A. Multitask Evolutionary Optimization

Multitask evolutionary optimization is a paradigm that aims at solving simultaneously multiple self-contained optimization problems. The concept was introduced in [1] as multifactorial optimization (MFO), named after the assumption that each task influences a particular factor in the evolution of the population. Gupta *et al.* [1] also defined the following concepts for MFO, on which they based a proposal of a genetic algorithm (GA) to tackle multifactorial problems.

Definition 1: The factorial cost Ψ_j^i of an individual x_i on a task T_j corresponds to the value of the objective function f_j

evaluated at x_i, penalized by the constraint violation terms, if applicable.

Definition 2: The factorial rank r_j^i of x_i on a task T_j corresponds to the rank of Ψ_j^i with respect to the remainder solutions of task T_j. In case multiple individuals achieve the same factorial cost, the parity is determined by random tie breaking.

Definition 3: The scalar fitness of x_i is proportional to the inverse of the best factorial rank of the individual, determined by $\varphi_i = (\min_{j\in\{1,\dots,K\}}\{r_j^i\})^{-1}$, where K is the number of tasks.

Definition 4: The skill factor τ_i of an individual is the task, among all considered in the MFO, on which the individual achieves the highest rank. Hence, $\tau_i = \text{argmin}_j\{r_j^i\}$, where $j \in \{1, \dots, K\}$.

Based on the fitness and skill factor of the individuals, the GA proposed in [1] generates the offspring population using assortative mating through simulated binary crossover (SBX) [9] and vertical cultural transmission. Both methods enable knowledge transfer between individuals assigned to different tasks, however, at different levels. When individuals with different skill factors are selected to generate the offspring, a crossover operator in the assortative mating combines these individuals with a given *random mating probability* (rmp), defined by the user; otherwise, the individuals are mutated and each offspring has a single parent. In the vertical cultural transmission, the skill factor of one of two parents (chosen randomly) is assigned to the offspring. Hence, if the fittest individuals of each task carry information that can improve the performance of the individuals in other tasks, these mechanisms allow to propagate the respective genes across the population and improve the convergence behavior. Gupta *et al.* [1] evaluated the algorithm on a set of continuous and discrete benchmark functions, and concluded that their proposal improves the performance in two-task MFOs not only by spreading the solutions on the landscape but also by evading obstacles and local optima. These mechanisms for knowledge transfer were also implemented in other MFEAs [10]–[12].

These techniques for knowledge transfer partially explore the benefits of sharing information across optimization problems, since the information is propagated randomly and is only based on a pair of individuals [13]. Thus, approaches that reduce the randomness in the transfer of knowledge, as in [14], and particle swarm optimization algorithms, which have learning-based mechanisms for sharing information, yield higher efficiency in knowledge transfer and achieve better performance in MFOs [15]–[17]. Alternatively, learning from historical optimization data, if possible, also improves the quality of knowledge transfer in MFOs either by injection of preexisting solutions, or by creating surrogate models or by using structured knowledge learned from past optimization data [18]–[20].

Regardless of the algorithm, the representation is central to knowledge transfer for design optimization. Gupta *et al.* [1] have proposed a unified random-key representation, where each design variable is encoded as a number $z \in [0, 1]$ with upper and lower bounds of variation. Although the representation is simple and enables mathematical operations between designs, the ordering of the variables still requires experience of the user to detect a semantic overlap between the design domains. This is often unintuitive in real-world problems. Consequently, the performance of MFEAs can decrease if the tasks in the MFO differ in dimensionality and if the optimal designs are located in different regions of the design space. Ding *et al.* [21] have overcome these challenges by proposing a generalized MFEA (G-MFEA), which shifts and shuffles the variables of the unified representation. While shifting the variables allows to place the optimal designs in similar regions of the design space, which fosters the positive knowledge transfer, the permutation of the variables allows the algorithm to transfer the information between different combinations of variables, which addresses the compatibility between the features of each task. The approach in [21] focuses on the interaction between design variables and not on deriving a representation that leverages synergies between the design spaces.

If data from previous optimizations are available, it is possible to learn a representation that is feasible for multiple tasks and domains. Artificial neural networks (ANNs), in particular, autoencoders, learn and compress the input data into a low-dimensional representation that is suitable for optimization tasks [6], [8], [22]. Feng *et al.* [23] proposed a one-layer denoising autoencoder for learning the relation between the design representations of heterogeneous problems from historical optimization data. Given a new optimization problem T, after a predefined number of generations, the autoencoder was trained to reconstruct the population of T from the population of a past optimization problem S. If solutions in S had better fitness than in T, the individuals were included into the population of T by utilizing the mapping learned by the autoencoder. The authors assessed their approach with benchmark functions and a real-world optimization for polymer composites manufacturing with the result that the proposed algorithm reduced the number of evaluations on average by 61%.

B. 3-D Point Cloud Autoencoders

The representation of 3-D data is not canonical and different methods have been explored for geometric deep learning applications [4], [5]. 3-D point clouds are simpler and more efficient for representing 3-D data than voxels and meshes, since point clouds are sampled directly from CAE models requiring low (if any) preprocessing effort and preserve enough geometric details [24]. Furthermore, the architectures available for learning on point cloud data do not require topological similarities between geometries, as for meshes [25], [26], which increase the range of potential applications for geometric deep learning in engineering scenarios [27].

Nevertheless, learning on 3-D point cloud data is challenging, as the representation is invariant against the permutation of the points and lacks high-level information about the geometry [28]. According to the experiments and taxonomy of the survey in [24], point-based networks address the permutation invariance by handling the input data with pointwise operators and by processing larger-scale features at deeper layers of the network through global operators, e.g., *max-pooling*. The autoencoders in this class comprise a bottleneck layer placed

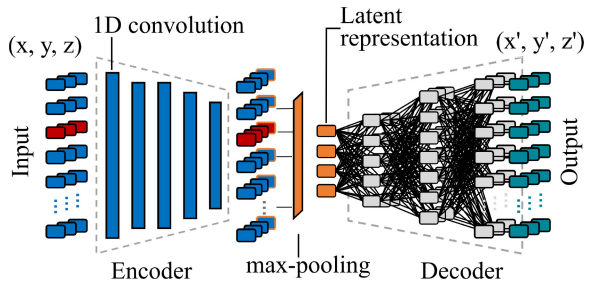

Fig. 2. Architecture of the 3-D point cloud autoencoder proposed in [7].

after the global operator to learn a compact representation of the input data, the so-called latent vector. This layer also divides the network into two parts: 1) an encoder, which maps the input data to the latent features and 2) a decoder, which recovers the Cartesian coordinates of the points to reconstruct the point cloud shape. Among the point-based networks available in [7] and [28]–[31], we based our approach on the autoencoder proposed in [7] (Fig. 2), which achieved high-quality results in shape generative tasks.

The architecture proposed in [7] comprises five 1-D convolutional layers, followed by a max-pooling operator that extracts the latent representation, and three fully connected layers to reconstruct the point clouds from the parameters in the latent space. All the network layers, apart from the max-pooling and output layers, were activated with rectified linear units (ReLUs). To assess the performance of the network as a shape-generative and classification model, the authors trained the proposed autoencoder and derivations on different shape classes of ShapeNetCore [32], considering two point-permutation invariant loss functions: 1) the Chamfer distance (CD) [33] and 2) Earth mover's distance (EMD) [34]. Hence, while the 1-D convolutions in the encoder ensure that latent features are invariant against the ordering of the points, the loss functions enable the autoencoder to learn the point clouds with unknown point correspondence between different shapes.

For the present work, we opted for the architecture utilized in [27], where the authors modified the activation functions of the network proposed in [7] to bind the values of the latent and output spaces. Furthermore, besides reporting similar shape reconstruction capabilities (Fig. 3), the authors applied the autoencoder as a shape-generative model in an evolutionary target shape matching optimization framework in [22]. The autoencoder achieved comparable performance to a representation based on a free-form deformation (FFD) lattice [35] for target shapes, whose geometric features were included in the training set of the autoencoder. Novel features (i.e., features not included in the training set) were difficult to be represented by the autoencoder and to be found by the optimization algorithm.

Albeit the extrapolation of knowledge is a known challenge for machine learning algorithms, Rios *et al.* [27] proposed a novel feature visualization technique to analyze the features learned by the point cloud autoencoder. After training the network on the car class of ShapeNetCore, the authors projected the activations of the last convolutional layer onto the corresponding input 3-D point clouds and showed that the

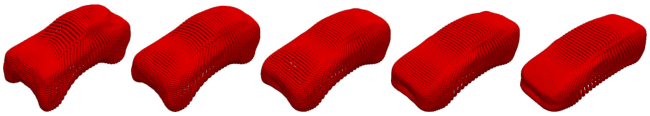

Fig. 3. Reconstruction of shapes obtained by interpolating the latent representations of two car shapes (extreme left and right) obtained with the autoencoder implemented in [22].

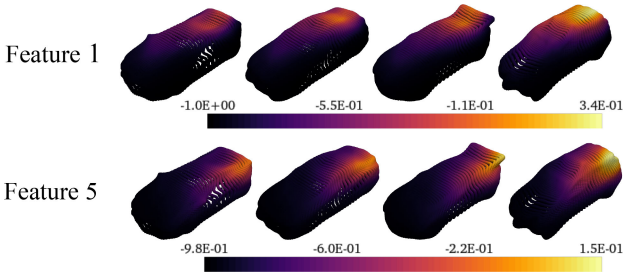

Fig. 4. Visualization of two network features of the last convolutional layer of the autoencoder for four different shapes. Brighter colors indicate higher activation values.

latent variables—or the maximum activation values of each feature—indicate the occupancy of distinct regions in the input space (Fig. 4). Furthermore, by reconstructing point clouds after transferring variables between shape representations in the latent space, the authors showed that the decoder changed the distribution of points in space according to the peaks of activations that were visualized on the point clouds. If the set of latent variables after the transfer of features represented a combination of occupied regions in the input space that was frequently observed in the data set, the decoder could reconstruct feasible 3-D shapes with mixed geometric characteristics of the original shapes. Hence, the transfer of latent features is a potential method to exploit common geometric characteristics between a set of shapes. Furthermore, it can transfer shape characteristics that potentially increase the fitness of the designs in an MFO problem.

Finally, the surface reconstruction on 3-D point clouds remains a challenge for embedding the autoencoder in automated optimization pipelines as a shape-generative model. The available remeshing techniques require manual tuning and postprocessing [36], which is prohibitive for automatic optimization pipelines. Novel ANN-based methods still can only be applied to topologically simple meshes and the computational cost is also still high [37], [38]. Rios *et al.* [39] proposed to morph optimal FFD-based prototypical meshes to resemble the point clouds based on a search in the latent space. The approach circumvents the limitations of mesh topology, however, morphing the prototypes requires a second optimization loop, which increases the optimization costs. The shrink wrapping technique [40], [41] generates only watertight genus-0[1] meshes, but requires lower computational costs and automation effort, and thus, we implemented the method for generating meshes on the point clouds. Furthermore, as

[1]The *genus* of a shape is the maximum number of cuts along a nonintersecting closing curve defined on the manifold. For example, a sphere has genus-0 while a torus has genus-1.

TABLE I
ARCHITECTURE OF OUR 3-D POINT CLOUD AUTOENCODER

Layer	Type	Activation	Features	Output dimensions
1	1D-C	ReLU	64	[N×64]
2	1D-C	ReLU	128	[N×128]
3	1D-C	ReLU	128	[N×128]
4	1D-C	ReLU	256	[N×256]
5	1D-C	tanh	L	[N×L]
6	max-pool.	-	L	[1×L]
7	FC	ReLU	256×3	[256×3]
8	FC	ReLU	256×3	[256×3]
9	FC	sigmoid	N×3	[N×3]

1D-C: 1D-convolution L: Number of latent variables
N: Size of the point cloud FC: *Fully connected*

in [6], the shrink wrapping organizes the point clouds according to the allocation of the vertices in the mesh. This allows us to modify the loss function for the decoder to learn the point ordering, easing the mesh reconstruction.

III. METHODS

In this section, we present the architecture, training, and application of the 3-D point cloud autoencoder utilized in our experiments, as well as our implementation of the MFEA adapted from [1]. We also present the experiments that verified our implementation using benchmark functions. Finally, we perform a preliminary MFO with the proposed geometric representation in order to assess the feasibility of our approach.

A. Learning on 3-D Point Cloud Data

The architecture of the 3-D point cloud autoencoder used in the experiments follows the implementation in [27] (Table I). The encoder comprises five 1-D convolutional layers, where the first four layers are activated with ReLUs and the last layer with a hyperbolic tangent function. Since 1-D convolutions handle the points individually, the size of the point cloud is preserved through the convolutional layers and each activated output depends exclusively on the coordinates of the corresponding input point. Following the convolutional layers, a max-pooling operator over the obtained features yields a latent representation $\mathbf{Z} \in [-1, 1]^L$, where L is the number of latent features. To recover the Cartesian coordinates of the points from the latent space, the decoder comprises three fully connected layers: the first two activated with ReLU and the last with a sigmoid function.

We verified the architecture by comparing the reconstruction losses with the ones published in [7]. The data set comprises 3-D point clouds with 2048 points sampled from the car class of ShapeNetCore [32] and split into 90%/10% for training and testing the networks, respectively. We defined $L = 128$ and optimized the network's weights using the Adam optimizer [42], which only requires the first-order gradients of the loss function and adapts individual learning rates for the parameters based on estimates of the moments of the gradient. Hence, while being memory efficient, the algorithm handles high-dimensional models with sparse gradients and nonstationary settings better than conventional algorithms such as gradient descent [42]. We trained the networks with the learning rate $\eta = 5.00\text{E}{-}04$ and the momenta $\beta_1 = 0.9$ and $\beta_2 = 0.99$. Finally, we used the CD [33] between the input and output point clouds $S_i, S_o \subseteq \mathbf{R}$ as the loss function

$$\text{CD} = \sum_{p_i \in S_i} \min_{p_o \in S_o} \|p_i - p_o\|_2^2 + \sum_{p_o \in S_o} \min_{p_i \in S_i} \|p_i - p_o\|_2^2. \quad (1)$$

Here, p_i and p_o are points in the input and output point clouds, respectively. For these settings and considering a confidence interval of 95%, the obtained losses on the test data were $\text{CD} = (3.03 \pm 0.07)\text{E}{-}04$ and $\text{CD}_R = (4.00 \pm 0.06)\text{E}{-}04$ for our autoencoder and the architecture implemented in [7], respectively. Hence, we considered our model verified with performance comparable to the state of the art.

In the following experiments, we replace the random sampling of point clouds with a uniform probability by a shrink wrapping meshing algorithm based on [43]. Our motivation was to simplify the surface reconstruction on the point clouds generated by the autoencoder, so that the shapes could be directly used in engineering simulations, e.g., computational fluid dynamics (CFDs). Typically, a shrink wrapping algorithm has three steps: 1) initial coarse mesh generation; 2) shrinking; and 3) surface smoothing. We used as initial mesh a rectangular box, which was meshed with 6146 vertices and triangular elements, with faces tangential to the extremes of the target shape. In the shrinking phase, the vertices of the initial mesh are iteratively displaced toward the nearest points in the target shape. Hence, for the iteration $t + 1$, a vertex $\mathbf{p}_i$ of the initial mesh is updated according to the following equation:

$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \alpha\left(\mathbf{p}_n - \mathbf{p}_i^t\right) \quad (2)$$

where $\mathbf{p}_n$ is the nearest point in the shape of interest and $\alpha \in (0, 1)$ is the step size. In our implementation, we considered $\alpha = 0.5$ as recommended in [43].

In the last step, the meshes are smoothed for relaxing the shrink-wrapped surface to obtain a more uniform distribution of points. For our experiments, we opted for a Laplacian smoothing algorithm as proposed in [44]. We generated the data set for our experiments sampling 3500 shapes from the car class of ShapeNetCore, considering ten and six iterations, which we determined experimentally, as termination criteria (Fig. 5).

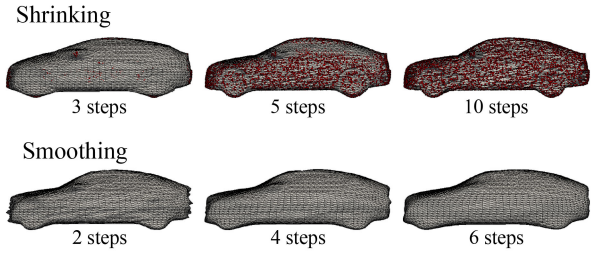

Fig. 5. Intermediate meshes obtained with the implemented shrink wrapping algorithm applied to a car shape sampled from the ShapeNetCore data set. The target point cloud is represented with red spheres in the first row.

B. Visualization and Transfer of Autoencoder Features

We analyzed the features learned by the autoencoder in the latent layer following the proposal in [27]. This technique allows us to select which latent features to optimize

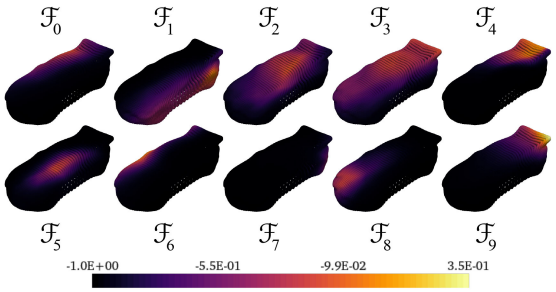

Fig. 6. Features learned in the last convolutional layer projected onto the corresponding 3-D point cloud, sampled from the data set.

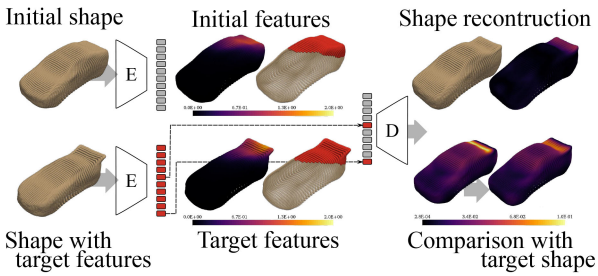

Fig. 7. Workflow for identifying and transferring latent features between shape representations. In the mid column, plots of $\mathcal{F}_4$ and $\mathcal{F}_7$ and regions with values above average are highlighted in red. On the right column, the distance between points is with respect to the initial shape (top) and target shape, before and after the transfer (bottom).

for each task, thereby constraining the regions to be modified and fostering common design characteristics in the population. To demonstrate the application of the visualization technique, we trained the 3-D point cloud autoencoder with $L = 10$, using the same algorithm and hyperparameters as presented in Section III-A (data set sampled with the shrink wrapping algorithm), to ease the analysis and comparison of latent features. Since the point clouds were ordered, we changed the loss function from CD to the mean-squared distance (MSD) between corresponding points in the input and output point clouds, defined as

$$\text{MSD} = \frac{1}{N}\sum_{i=1}^{N}\left\|p_{i,i} - p_{o,i}\right\|_2^2 \tag{3}$$

where $p_{i,i}$ and $p_{o,i}$ are the ith points of the input and output point clouds, respectively. For completeness, the network achieved CD $= (1.39 \pm 0.05)$E$-$04, which is within the same order of magnitude as in the previous analyses. Despite the changes in the data set and loss function, the features obtained at the last convolutional layer (Fig. 6) have similar patterns of activations as the results reported in [27].

Based on the visualization of the features, we observed that $\mathcal{F}_4$ and $\mathcal{F}_9$ represent the rearmost region of the car shape, similar to a rear spoiler. By transferring these features to the representation of a different shape, one generates a crossover design, which includes the spoiler as a "module" but preserves the remaining characteristics of the initial shape (Fig. 7). Compared to alternative data-driven representations, the proposed autoencoder represents and operates on more localized regions of the shapes, which is a potential advantage in highly nonlinear optimization problems [45].

Hence, in an MFO scenario, where the mixing of tasks has resulted in overlapping sets of latent features, the offspring generated from parents with different skill factors combine geometric characteristics of designs that potentially excel in different tasks. Additionally, common geometric characteristics between the designs can be achieved without the explicit definition of constraints.

Algorithm 1 Pseudocode of the GA Used in the Multitask Optimization Experiments

Require: Trained autoencoder $(\mathcal{E}, \mathcal{D})$, rmp, λ
INITIALIZATION
for Each task T_j, $j = (1, \ldots, K)$ **do**
  Calculate the latent representation of the initial design S_j using the trained encoder: $\mathbf{Z}_j^0 = \mathcal{E}(S_j)$
  Mutate λ/K times the initial solution $\mathbf{Z}_j^0$ and append to the population P
  Assign the skill factor of $\mathbf{Z}_j^0$ to each offspring $\mathbf{Z}_i$: $\tau_i = j$
end for
for Each individual in p_i, $i = (1, \ldots, \lambda)$ **do**
  Transfer the features in $\mathbf{Z}_j'$, from $\mathbf{Z}_i$ to $\mathbf{Z}_j^0$, where $j = \tau_i$, generating a temporary individual $\mathbf{Z}_j^{0*}$
  Reconstruct the shape S_i in the Cartesian space using the trained decoder: $S_i = \mathcal{D}(\mathbf{Z}_j^{0*})$
  Evalaute the design S_i for the task τ_i and calculate Ψ_i
end for
ITERATIONS
while (termination criteria are not achieved) **do**
  Calculate the factorial rank r_i and the scalar fitness φ_i of the individuals
  Select individuals to generate the population in the next iteration
  Perform **assortative mating** and generate offspring C
  Assign τ_i through **vertical cultural transmission algorithm**
  for Each individual in c_i, $i = (1, \ldots, \lambda)$ **do**
    Transfer the features in $\mathbf{Z}_j'$, from $\mathbf{Z}_i$ to $\mathbf{Z}_j^0$, where $j = \tau_i$, generating a temporary individual $\mathbf{Z}_j^{0*}$
    Reconstruct the shape S_i in the Cartesian space using the trained decoder: $S_i = \mathcal{D}(\mathbf{Z}_j^{0*})$
    Evalaute the design S_i for the task τ_i and calculate Ψ_i
  end for
end while

C. Multifactorial Optimization Algorithm

For the optimization experiments, we adapted the MFEA proposed in [1] to our framework (Algorithm 1).

We assumed that each task $T_j, j = (1, \ldots, K)$ has an initial design S_j^0 with a corresponding latent representation $\mathbf{Z}_j^0$ obtained with a trained autoencoder AE$(\mathcal{E}, \mathcal{D})$. We also consider that the designer selects a set of latent features $\mathbf{Z}_j' \subseteq \mathbf{Z}$ for each task, which, during the optimization, is transferred from the MFEA solutions to the initial representation $\mathbf{Z}_j^0$ and

Algorithm 2 Pseudo-Code of the Assortative Mating Mechanism

Require: Parents (p_a, p_b), rmp

Generate a random number $rand \in [0, 1]$
if $(\tau_a = \tau_b)$ or $(rand < \text{rmp})$ **then**
  Parents p_a and p_b crossover, generating two offspring individuals c_a and c_b
else
  Parent p_a is mutated, generating an offspring c_a
  Parent p_b is mutated, generating an offspring c_b
end if

Algorithm 3 Pseudocode of the Vertical Cultural Transmission Mechanism

Require: Offspring c_i

if c_i has two parents **then**
  Generate a random number $rand \in [0, 1]$
  if $(rand < 0.5)$ **then**
    c_i imitates p_a: $\tau_i \leftarrow \tau_a$
  else
    c_i imitates p_b: $\tau_i \leftarrow \tau_b$
  end if
else
  c_i imitates its single parent
end if

forwarded to the decoder $\mathcal{D}$ to recover the 3-D representations in the Cartesian space.

In the initialization step, we modified the algorithm to generate the population P with individuals $\mathbf{Z}_i, i = (1, \lambda)$ by mutating λ/K times each initial solution $\mathbf{Z}_j^0$. For each offspring $\mathbf{Z}_i$, the algorithm assigns $\tau_i = j$, where j is the task of the design $\mathbf{Z}_j^0$ that originated the individual $\mathbf{Z}_i$. Our assumption is that if each solution S_j was designed to solve the task T_j, the mutations of S_j potentially solve T_j better than the other tasks. Hence, we can reduce the computational effort in the initial generation by evaluating each individual for a single task. The algorithms for assortative mating (Algorithm 2) and vertical cultural transmission (Algorithm 3) were maintained as proposed in [1].

We verified our implementation using four benchmark functions: 1) Sphere; 2) Rastrigin; 3) Ackley; and 4) Rosenbrock in different combinations. We considered the domain $x_i \in [-50, 50]^{20}$, where the features were linearly mapped to the unified search space $[-1, 1]^{20}$, emulating the operations performed by the autoencoder.

Regarding the hyperparameters, we set the population size as $\lambda = 100$, the maximum number of generations to 300, and the spread factor of the SBX algorithm to $\beta_{SBX} = 10$. Also, we optimized the functions considering rmp $= (0.0, 0.3, 0.5, 0.7, 1.0)$ to assess the effect of increasing the variety introduced by the algorithm into the population through assortative mating. For each case, we performed the optimizations for 30 different sets of initial solutions. We also optimized the functions individually, considering the same conditions and with a GA initialized with the same hyperparameters.

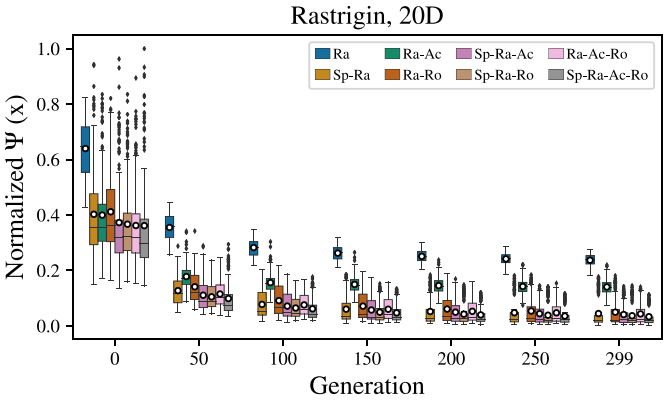

Fig. 8. Factorial cost of the fittest individuals per generation obtained for different function combinations and baseline experiment for the Rastrigin function.

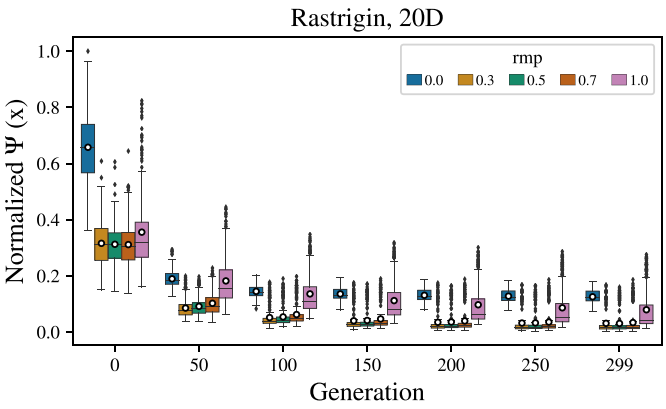

Fig. 9. Factorial cost of the fittest individuals per generation organized according to the rmp values of each experiment using the Rastrigin function.

By analyzing the factorial cost $\Psi(x)$ of the sets containing the fittest individuals in each generation, we observed that for different function combinations (Fig. 8), the MFEA accelerated the convergence and improved the quality of the solutions. The largest improvement was achieved for the Rastrigin function, from (2.37E−01±7.98E−03) to (3.80E−02±3.02E−03) in the last generation (95% confidence interval), which is similar to the results reported in [1]. Also, the variance of the factorial costs obtained for the Rosenbrock function increased with the MFEA approach, varying from $(3.34\text{E}{-}05 \pm 6.98\text{E}{-}06)$ for the baseline experiments to $(4.92\text{E}{-}05 \pm 1.14\text{E}{-}05)$ for the MFEA. This has also been observed in [21], where the authors claim that the MFEA performance decreases when the different tasks have solutions in very different locations of the quality landscape.

In a second verification, we analyzed the same results with respect to the rmp values (Fig. 9). We observed that the cases with intermediate rmp values converged faster and achieved lower values of $\Psi(x)$. In line with the discussion in [1], rmp=0 prohibited knowledge transfer between individuals; hence, during the optimization, the population lost diversity and the optimizer had more difficulty to overcome local minima. When rmp = 1, the algorithm combined pairs of shapes regardless of their cultural traits (τ_i), which excessively increased the diversity of the population and slowed down the optimization.

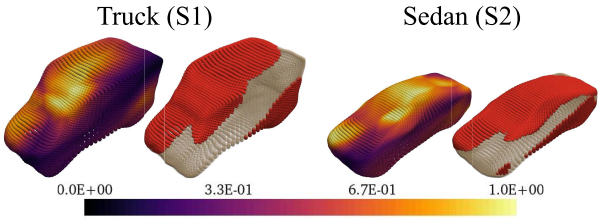

Fig. 10. Visualization of the maximum activations for the selected features for each shape and regions with highest combined activations highlighted in red.

In summary, the performance as well as the properties of the algorithm are what we expected from the literature. To further verify the performance also for the design optimization domain, we run a low-cost shape optimization where the representation is learned with the 3-D point cloud autoencoder.

D. Low-Cost Shape Optimization Experiment

In this section, we test the MFEA framework for shape design optimization problems, i.e., the optimization of two car shapes: 1) a truck (S1) and 2) a sedan (S2), for which we minimize the volume. We defined the factorial cost as follows:

$$\Psi(\mathbf{Z}_i) = V(S_i) + \rho \mathrm{MSD}\big(S_i, S_{i,N}\big) \tag{4}$$

where $V(S_i)$ is the volume of the shape S_i. Shape $S_{i,N}$ is the shape in the data set with the closest representation in the latent space to the representation of S_i, and ρ is the penalty factor. The penalty term in $\Psi(\mathbf{Z}_i)$ drives the optimizer to regions with higher density of learned shapes, which avoids the reconstruction of noisy point clouds and shapes without geometric characteristics of cars. Based on preliminary experiments, we defined $\rho = 0.75$.

For optimizing the shapes, we selected the set of features $\mathbf{Z}'_1 = ((1, 6, 7), 2, 5, 8)$ and $\mathbf{Z}'_2 = ((1, 6, 7), 0, 2, 3, 4, 5, 8, 9)$, where the first three map to regions in the underbody of the car shapes (Fig. 10). Our motivation was that while the sedan shape allows more freedom in the design of the shape, in the optimization of the truck, we constrained the modifications to the region of the passenger cabin to preserve volume for transporting material.

We configured the MFEA to optimize the shapes with population size $\lambda = 15$ and for at most 30 generations to set a computational budget that is realistic for scenarios using more complex simulations as described in the next section. We defined the rmp as 0.3, which achieved the best compromise between convergence speed and fitness of the optimized individuals in the previous experiments. Finally, we set $\rho = 7.5$ and we performed the optimization for 30 different initializations for statistical analysis.

In our first analysis, we compared the convergence behavior with respect to the baseline experiments (Fig. 11). We observed that the MFEA accelerated the optimization of the truck, improving the normalized volume of the optimal shape from $(8.46 \pm 0.02)\mathrm{E}{-1}$ to $(4.96 \pm 0.06)\mathrm{E}{-1}$. In the optimization of the sedan, the results with the MFEA were slightly worse, yet with volumes within 95% confidence

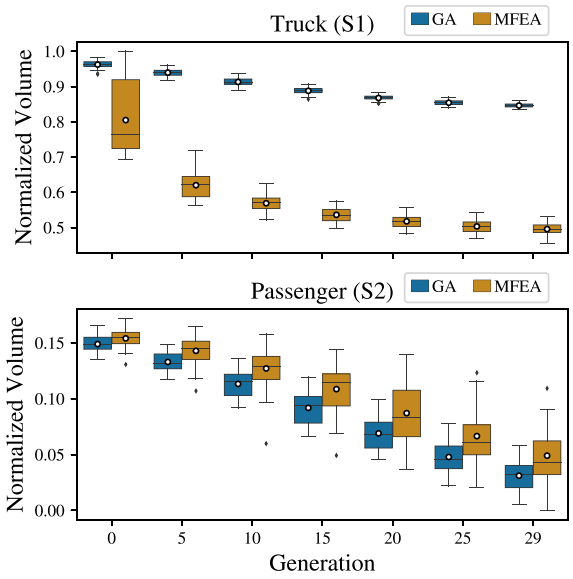

Fig. 11. Fitness of the best individuals per generation.

interval of the baseline results: $(3.11 \pm 0.54)\mathrm{E}{-2}$ achieved with the baseline and $(4.90 \pm 0.93)\mathrm{E}{-2}$ with the MFEA.

In order to evaluate the results, we analyzed how the latent features evolved during the optimizations (Fig. 12). In a first step, we calculated the Euclidean distance in the latent space between the fittest individuals assigned to S1 and S2 at each generation. We observed that the distance decreases over the generations in the MFEA framework, which is in line with the working principle of the MFEA and feature transfer results. In a second step, we projected the representations of the optimal designs into a 2-D space using uniform manifold approximation and projection for dimensionality reduction (UMAP) [46], [47]. We observed that the MFEA displaced the truck designs toward the sedan designs, which lay in a similar region in both optimizations with MFEA and GA. Since the sedan had an initial volume smaller than the truck, we concluded that by assigning sedan designs to the task of the truck, the algorithm generated offspring of the truck design with characteristics closer to the sedan design, thus accelerating the optimization. The opposite condition, however, had a negative impact on the sedan task, and thus, we could not observe any improvement in the optimization of the sedan.

In a further analysis, we reconstructed the mean optimal shapes obtained with each algorithm. The MFEA yielded shapes with similar windshield geometry, which was a region covered by both sets of latent features selected for the optimization of the shapes. Furthermore, we observed that the mean truck shape yielded by the MFEA was smaller than by the GA, and both algorithms resulted in similar sedan shapes, which is in line with the numerical results.

In summary, we conclude that the representation learned by the proposed autoencoder is suitable as a unified representation for MFO problems. The autoencoder learned transferrable geometric features, which combined with the knowledge transfer mechanisms of the MFEA yielded optimized shapes with shared geometric features. Having verified the algorithm on

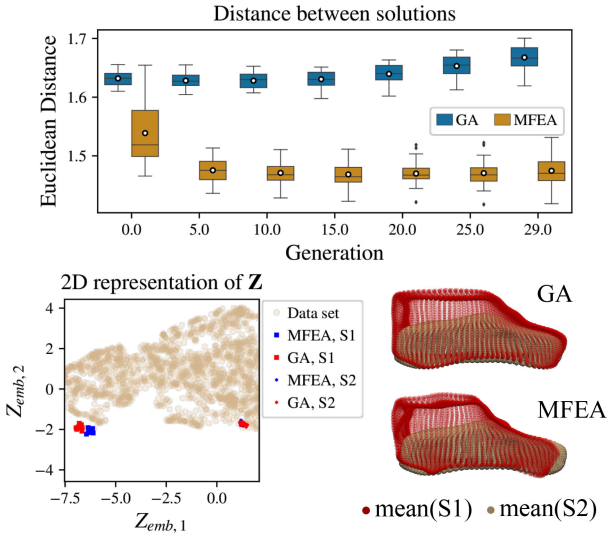

Fig. 12. Euclidean distance between the fittest designs of each task obtained at each generation (top), 2-D projection of the data set and optimal designs for the optimizations with GA and MFEA (bottom left), and reconstruction of the mean optimal shapes (bottom right).

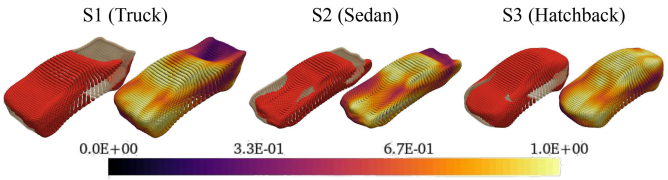

Fig. 13. Visualization of the features selected for the modifying the shapes during the optimization. The points with combined activation higher than the average of the point cloud were highlighted in red.

benchmark tests and a simplified design optimization task, we test its performance on a practical and more realistic vehicle shape aerodynamic optimization problem in the next section.

IV. Multitask Vehicle Aerodynamic Drag Optimization

The optimization of a vehicle shape to reduce the aerodynamic drag is a common problem in vehicle design. At the same time, to increase efficiency in manufacturing and maintenance, it is beneficial if vehicles share platforms and use common modules. Therefore, in this section, we apply the proposed MFEA method to a multitask vehicle shape optimization problem where we target both, optimal shapes with regard to drag minimization and sharing of a similar underbody structure.

A. Experimental Settings

For this set of experiments, we retrained the 3-D point cloud autoencoder on the same data set with $L = 20$ and the settings discussed in Section III-A to refine the regions learned by the autoencoder in the latent space. After training, the mean CD calculated on the test data was CD $=$ (1.15 ± 0.04)E$-$04, which is comparable to the quality of the previous configuration.

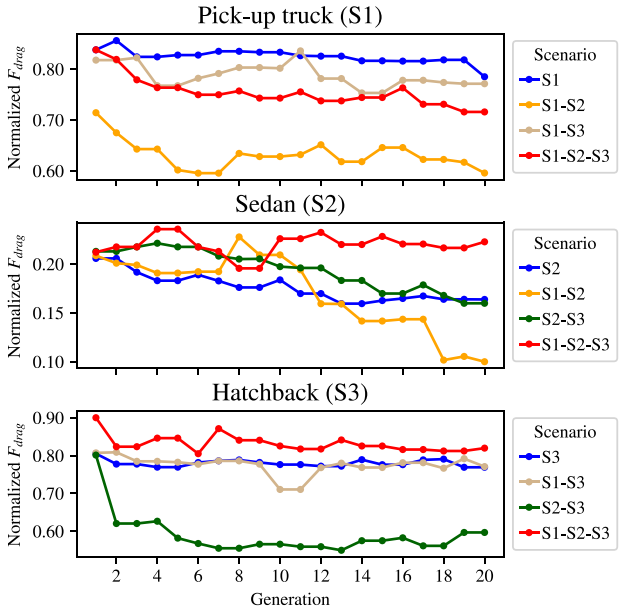

Fig. 14. Normalized drag force of the fittest designs of each generation for all optimization scenarios.

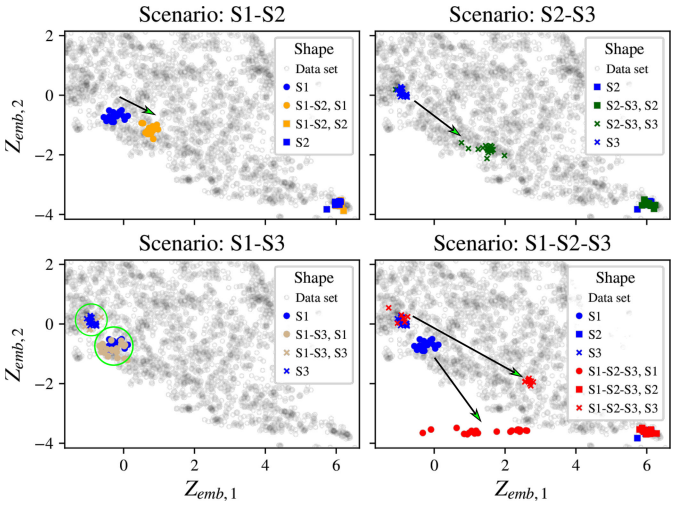

Fig. 15. Visualization of fittest designs embedded in a 2-D space from the latent features for comparing the regions in the latent space occupied by the designs in different optimization scenarios.

We selected three vehicle shapes for the multitask optimization: 1) a pick-up truck (S1); 2) a sedan (S2); and 3) a hatchback (S3). Our motivation was that the shapes have fundamentally different styles, and thus, we could observe in the optimized shapes how the geometric features were merged to reduce the aerodynamic drag. Based on the visualization of the latent features, we selected the sets of features $\mathbf{Z}'_1$, $\mathbf{Z}'_2$, and $\mathbf{Z}'_3$ for modifying the shapes S1, S2, and S3 during the optimization, respectively, each having five features in common that represent the region in the underbody of the car shapes (Fig. 13).

We evaluated the performance of the designs with CFD simulations using OpenFOAM.[2] The point clouds generated by

[2]https://www.openfoam.com/

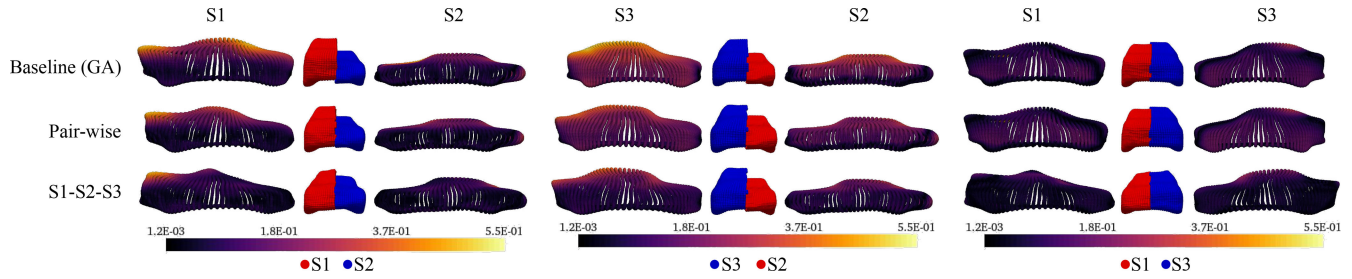

Fig. 16. Analysis of the similarity between optimal shapes obtained in different scenarios. The colormap indicates the distance between the nearest points in the compared shapes.

the autoencoder were converted to polygonal meshes using the shrink wrapping algorithm and embedded in a large virtual fluid domain for the simulation. We used the boundary conditions for a vehicle driving on a free road at 110 km/h and considered symmetry of the shapes with respect to the *xy*-plane to reduce the computational effort. We defined the factorial cost analogously to the low-cost shape optimization problem. We replaced the shape volume in (4) by the aerodynamic drag force, calculated by the CFD simulation. The penalty factor was chosen to be $\rho = 75$, since the aerodynamic drag values are two orders of magnitude greater than the volume of the shapes obtained in the previous optimization, and thus, the effects of the penalization would be similar.

Since the computational effort is considerably higher than required for computing the volume of the shapes, we reduced the population size to $\lambda = 15$ and the maximum number of generations to 20. We maintained rmp $= 0.3$, however, we reduced the SBX spread factor to $\beta_{SBX} = 7.5$ in order to avoid abrupt feature modifications, which could yield low-quality point cloud reconstructions. We implemented the workflow on a cluster consisting of machines with 2 Westmere 4 Core Xeon E5620 clocked at 2.4 GHz, which allowed us to solve each CFD simulation in parallel with 16 processors each. With this set up, each generation took up to 75 min and a complete optimization took on average 23h.

Finally, we optimized the shapes considering the following scenarios: the pairs (S1, S2), (S2, S3), and (S1, S3) and the three shapes simultaneously (S1, S2, S3). For assessing the performance of the optimization, we also optimized the shapes individually utilizing a GA with identical settings. Furthermore, we performed a single optimization for each scenario due to computational budget constraints.

B. Results and Discussion

First, we verified the convergence behavior (Fig. 14), where we observed that in most of the cases, the MFEA approach improved the fitness of the individuals. The largest improvements with respect to the baseline optimization were achieved in the pairwise optimizations, where the drag force of the shapes (S1-S2) and (S2-S3) was reduced by (24.11, 38.95)% and (2.41, 22.39)%, respectively. In the optimization with three shapes, apart from S1, none of the shapes improved in performance with respect to the baseline. A potential justification for this behavior is that shapes S1 and S3 have similar performance, yet considerably worse than S2, which has a smaller projected frontal area and thus, better aerodynamic performance [48]. Hence, the features that potentially drive the population to better fitness were stored in the minority of the population and the chances of negative transfer of knowledge was higher than in the pairwise optimizations. This conclusion is also in line with the optimization results of the pair (S1-S3), which yielded a slower convergence behavior and optimal designs with $(-1.74, +0.23)$% of variation with respect to the baseline optimizations.

In order to analyze the evolution of the designs, we embedded the latent space into a 2-D representation using UMAP and visualized the fittest individuals for different optimization scenarios (Fig. 15). Similar to the optimization with the low-cost geometric function, the MFEA drove the designs toward the fittest solution among the tasks, which in these scenarios were the shapes close to the sedan design. When the difference in performance was not significant (scenario S1-S3), the designs remained in similar regions in the latent space. When measuring the Euclidean distances in the latent space between the optimal designs, the largest differences to the baseline cases were observed for the scenario S1-S2-S3, where the distances of the optimal pairs [(S1,S2), (S2,S3), (S1,S3)] changed from (1.74, 1.78 0.58) to (1.28, 1.09, 0.58).

We confirmed our interpretations of the results by reconstructing and measuring the similarity between the optimal shapes (Fig. 16). Although the inital underbody structures were similar, it was particularly clear in comparisons using S2 and S3 that the transfer of features caused by the MFEA increased the similarity between the regions in the bottom and rear of the car shapes, adding the typical shape of the sedan to the hatchback. Similarly, the optimizer considerably changed the front of the pick-up toward the sedan geometry. The region in the back of the pick-up, which was not covered by the set Z_1', remained nearly identical to the initial shape. Therefore, we conclude that the representation learned in the latent space of the autoencoder can be used as a unified design representation for engineering multitask shape optimization problems. The algorithm improved the performance compared to single-task optimization with similar parameterization. Furthermore, the transfer of features within the MFO fosters commonalities between the shapes, which can be beneficial for manufacturing and maintenance efficiency.

V. Conclusion

Multitask evolutionary optimization algorithms solve simultaneously multiple self-contained tasks. However, mapping the different task design spaces to a unified representation is a

key challenge for design exploration. In shape optimization problems, finding a unified, yet practical representation is challenging. In this article, we proposed an MFEA framework utilizing the latent space learned by a 3-D point cloud autoencoder to balance the conditions of universality and practicability. Since the autoencoder learns exclusively on geometric data, the network yields a domain-independent representation and maps the designs directly from the unified space to the CAE representation in the Cartesian space.

In a simplified car shape optimization problem, we showed that the transfer of latent features between representations yields designs with interpolated characteristics, which in an MFO allows to increase the diversity in the set of individuals with the same skill factor and to foster common geometric characteristics between designs. Finally, we applied the MFEA for shape optimization to a real-world inspired multitask vehicle optimization problem, where we minimized the aerodynamic drag of three car shapes in four MFO scenarios. Additionally, our framework enables the transfer of latent features that represented the underbody of the car shapes, such that the optimal designs would share a nearly identical platform, a characteristic that is known to potentially increase manufacturing and maintenance efficiency. Our proposed framework outperformed single-task optimizations in both aerodynamic drag minimization and in platform sharing. In our future research, we aim at improving the knowledge transfer and exploration capabilities of the autoencoder by, e.g., including domain adaptation techniques [49] to the MFEA framework.

REFERENCES

[1] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: Toward evolutionary multitasking," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 343–357, Jun. 2016.

[2] S. N. Skinner and H. Zare-Behtash, "State-of-the-art in aerodynamic shape optimisation methods," *Appl. Soft Comput.*, vol. 62, pp. 933–962, Jan. 2018.

[3] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Proces. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[4] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, "Deep learning advances in computer vision with 3D data," *ACM Comput. Surveys*, vol. 50, no. 2, pp. 1–38, 2017.

[5] T. Friedrich, N. Aulig, and S. Menzel, "On the potential and challenges of neural style transfer for three-dimensional shape data," in *Proc. Int. Conf. Eng. Optim.*, 2019, pp. 581–592.

[6] N. Umetani, "Exploring generative 3D shapes using autoencoder networks," in *Proc. SIGGRAPH Asia Techn. Briefs*, 2017, pp. 1–4.

[7] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 80, 2018, pp. 40–49.

[8] S. Saha, S. Menzel, L. L. Minku, X. Yao, B. Sendhoff, and P. Wollstadt, "Quantifying the generative capabilities of variational autoencoders for 3D car point clouds," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, Canberra, ACT, Australia, 2020, pp. 1469–1477.

[9] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 2, pp. 1–34, 1994.

[10] A. Gupta, J. Mańdziuk, and Y.-S. Ong, "Evolutionary multitasking in bi-level optimization," *Complex Intell. Syst.*, vol. 1, nos. 1–4, pp. 83–95, 2015.

[11] A. Gupta, Y.-S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1652–1665, Jul. 2017.

[12] K. K. Bali, Y.-S. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: MFEA-II," *IEEE Trans. Evol. Comput.*, vol. 24, no. 1, pp. 69–83, Feb. 2020.

[13] Z. Liang, W. Liang, X. Xu, and Z. Zhu, "A two stage adaptive knowledge transfer evolutionary multi-tasking based on population distribution for multi/many-objective optimization," 2020. [Online]. Available: arXiv:2001.00810.

[14] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci.*, Nagoya, Japan, 1995, pp. 39–43.

[15] L. Feng *et al.*, "An empirical study of multifactorial PSO and multifactorial DE," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Donostia, Spain, 2017, pp. 921–928.

[16] B. Zhang, A. K. Qin, and T. Sellis, "Evolutionary feature subspaces generation for ensemble classification," in *Proc. Genet. Evol. Comput. Conf.*, 2018, pp. 577–584.

[17] Z. Tang and M. Gong, "Adaptive multifactorial particle swarm optimisation," *CAAI Trans. Intell. Technol.*, vol. 4, no. 1, pp. 37–46, 2019.

[18] M. W. Hauschild, M. Pelikan, K. Sastry, and D. E. Goldberg, "Using previous models to bias structural learning in the hierarchical BOA," in *Proc. 10th Annu. Conf. Genet. Evol. Comput.*, 2008, pp. 415–422.

[19] L. Feng, Y.-S. Ong, M.-H. Lim, and I. W. Tsang, "Memetic search with interdomain learning: A realization between CVRP and CARP," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 644–658, Oct. 2015.

[20] A. T. W. Min, R. Sagarna, A. Gupta, Y.-S. Ong, and C. K. Goh, "Knowledge transfer through machine learning in aircraft design," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 48–60, Nov. 2017.

[21] J. Ding, C. Yang, Y. Jin, and T. Chai, "Generalized multitasking for evolutionary optimization of expensive problems," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 44–58, Feb. 2019.

[22] T. Rios, B. Sendhoff, S. Menzel, T. Bäck, and B. van Stein, "On the efficiency of a point cloud autoencoder as a geometric representation for shape optimization," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, Xiamen, China, 2019, pp. 791–798.

[23] L. Feng, Y.-S. Ong, S. Jiang, and A. Gupta, "Autoencoding evolutionary search with learning across heterogeneous problems," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 760–772, Oct. 2017.

[24] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," 2019. [Online]. Available: https://arxiv.org/abs/1912.12033.

[25] O. Sorkine, "Laplacian mesh processing," in *Proc. Eurograph. State Art Rep.*, 2005, pp. 53–70.

[26] A. Ranjan, T. Bolkart, S. Sanyal, and M. J. Black, "Generating 3D faces using convolutional mesh autoencoders," in *Computer Vision (ECCV)*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer Int., 2018, pp. 725–741.

[27] T. Rios, B. van Stein, S. Menzel, T. Bäck, B. Sendhoff, and P. Wollstadt, "Feature visualization for 3D point cloud autoencoders," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2020, pp. 1–9, doi: 10.1109/IJCNN48605.2020.9207326

[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5105–5114.

[29] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, 2017, pp. 77–85.

[30] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 206–215.

[31] M. Gadelha, R. Wang, and S. Maji, "Multiresolution tree networks for 3D point cloud processing," in *Computer Vision (ECCV)*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham, Switzerland: Springer Int., 2018, pp. 105–122.

[32] A. X. Chang *et al.*, "ShapeNet: An information-rich 3D model repository," 2015. [Online]. Available: arXiv:1512.03012.

[33] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2463–2471, doi: 10.1109/CVPR.2017.264.

[34] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vision*, vol. 40, no. 2, pp. 99–121, Nov. 2000.

[35] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," in *Proc. 13th Annu. Conf. Comput. Graph. Interact. Techn.*, 1986, pp. 151–160.

[36] M. Berger *et al.*, "State of the art in surface reconstruction from point clouds," in *Proc. Eurograph. State Art Rep. (STARs)*, vol. 1, 2014, pp. 161–185.

[37] N. Sharp and M. Ovsjanikov, "PointTriNet: Learned triangulation of 3D point sets," in *Computer Vision (ECCV)*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham, Switzerland: Springer Int., 2020, pp. 762–778.
[38] R. Hanocka, G. Metzer, R. Giryes, and D. Cohen-Or, "Point2Mesh: A self-prior for deformable meshes," *ACM Trans. Graph.*, vol. 39, no. 4, pp. 1–12, 2020.
[39] T. Rios *et al.*, "Back to meshes: Optimal simulation-ready mesh prototypes for autoencoder-based 3D car point clouds," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, 2020, pp. 942–949.
[40] L. P. Kobbelt, J. Vorsatz, U. Labsik, and H. P. Seidel, "A shrink wrapping approach to rRmeshing polygonal surfaces," *Comput. Graph. Forum*, vol. 18, no. 3, pp. 119–130, 1999.
[41] Y. K. Lee, C. K. Lim, H. Ghazialam, H. Vardhan, and E. Eklund, "Surface mesh generation for dirty geometries by shrink wrapping using cartesian grid approach," in *Proc. 15th Int. Meshing Roundtable*, 2006, pp. 393–410.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–15.
[43] B. K. Koo, Y. K. Choi, C. W. Chu, J. C. Kim, and B. T. Choi, "Shrink-wrapped boundary face algorithm for mesh reconstruction from unorganized points," *ETRI J.*, vol. 27, no. 2, pp. 235–238, 2005.
[44] J. Vollmer, R. Mencl, and H. Müller, "Improved Laplacian smoothing of noisy surface meshes," *Comput. Graph. Forum*, vol. 18, no. 3, pp. 131–138, 1999.
[45] T. Rios, B. van Stein, P. Wollstadt, T. Bäck, B. Sendhoff, and S. Menzel, "Exploiting local geometric features in vehicle design optimization with 3D point cloud autoencoders," in *Proc. IEEE Congr. Evol. Comput.*, 2021.
[46] E. Becht *et al.*, "Dimensionality reduction for visualizing single-cell data using UMAP," *Nat. Biotechnol.*, vol. 37, no. 1, pp. 38–44, Jan. 2019. [Online]. Available: https://doi.org/10.1038/nbt.4314
[47] M. Ali, M. W. Jones, X. Xie, and M. Williams, "TimeCluster: Dimension reduction applied to temporal data for visual analytics," *Vis. Comput.*, vol. 35, no. 6, pp. 1013–1026, Jun. 2019.
[48] W.-H. Hucho, "Chapter 1—Introduction to automobile aerodynamics," in *Aerodynamics of Road Vehicles*, W.-H. Hucho, Ed. London, U.K.: Butterworth-Heinemann, 1987, pp. 1–46.
[49] R. Lim, A. Gupta, Y.-S. Ong, L. Feng, and A. N. Zhang, "Nonlinear domain adaptation in transfer evolutionary optimization," *Cogn. Comput.*, vol. 13, no. 2, pp. 290–307, Mar. 2021. [Online]. Available: https://doi.org/10.1007/s12559-020-09777-7

Thiago Rios (Member, IEEE) received the master's degree in mechanical engineering from the Federal University of Santa Catarina, Florianópolis, Brazil, in 2018.

Since 2018, he has been with Honda Research Institute Germany, Offenbach, Germany, where he is currently a Scientist with the Optimization and Creativity Group and Early-Stage Researcher of the ECOLE Project, which is an Innovative Training Network funded by the EU Horizon 2020. His current interests include geometric processing and geometric deep learning for design representation in evolutionary optimization problems, and application of evolutionary algorithms for solving real-world problems in the automotive engineering domain.

Bas van Stein was born in Sassenheim, The Netherlands, in 1989. He received the master's degree in computer science and the Ph.D. degree from the Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands, in 2013 and 2018, respectively.

He is currently a part-time Postdoctoral Researcher with Leiden University in automated machine learning and neural architecture search. Next to his academic career, he founded several companies in software development and artificial intelligence products such as Smartnotation B.V., Leiden. His current research interests are algorithm design and improvements for neural architecture search and hyperparameter optimization.

Thomas Bäck (Senior Member, IEEE) received the Diploma degree in computer science and the Ph.D. degree from the University of Dortmund, Dortmund, Germany, in 1990 and 1994, respectively.

He is a Professor of Computer Science with the Leiden Institute of Advanced Computer Science, Leiden University, Leiden, The Netherlands. His research interests include evolutionary computation, machine learning, and their real-world applications, especially in sustainable smart industry and health.

Prof. Bäck was a recipient of the IEEE COMPUTATIONAL INTELLIGENCE SOCIETY EVOLUTIONARY COMPUTATION Pioneer Award in 2015. He received the Best Ph.D. Thesis Award from the German Society of Computer Science (GI) in 1995. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and an Area Editor for the *ACM Transactions on Evolutionary Learning and Optimization*. He was elected as a Fellow of the International Society of Genetic and Evolutionary Computation in 2003.

Bernhard Sendhoff (Senior Member, IEEE) received the Ph.D. degree in applied physics from Ruhr-Universität Bochum, Bochum, Germany, in 1998.

He was with Honda Research Institute Europe GmbH, Offenbach, Germany, from 2003 to 2010, as a Chief Technology Officer, and from 2011 to 2017, as a President. Since 2017, he has been an Operating Officer with Honda Research and Development Ltd., Tokyo, Japan, and the Head of the Global Operation, Honda Research Institutes. He is an Honorary Professor with the Technical University of Darmstadt, Darmstadt, Germany. He has authored or coauthored over 180 scientific publications.

Dr. Sendhoff is a Senior Member of ACM and a member of SAE.

Stefan Menzel received the Dipl.-Ing. degree in civil engineering from RWTH Aachen, Aachen, Germany, in 1998, and the Ph.D. degree in civil engineering from Technical University Darmstadt, Darmstadt, Germany, in 2004.

Since 2004, he has been with the Honda Research Institute Europe, Offenbach, Germany, where he is currently a Chief Scientist with the Optimization and Creativity Group. His current research interests include evolutionary optimization with special focus on adaptive representations, machine learning for knowledge transfer, and multidisciplinary optimization for real-world applications.