**National Technical University of Athens**
**School of Mechanical Engineering**
**Fluids Section**
**Parallel CFD & Optimization Unit**

# Convolutional Neural Networks as Surrogate Models in the Early Stages of the Automobile Design process

Diploma Thesis

**Faliakos Vasileios**

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2025

# Acknowledegments

I would like to begin by expressing my sincere gratitude to my supervisor, Professor Kyriakos C. Giannakoglou. I am deeply grateful for his continuous support and insightful guidance throughout the course of this Diploma Thesis. His distinguished academic stature as a capable, passionate and inspirational educator were evident in every interaction during the years of my studies. I deeply appreciate his unique mentoring spirit, offering valuable direction and encouraging independent thinking. I am truly inspired by his methodical approach to address problems, combined with clarity of thought and sophisticated insight, which undeniably contributed to my academic development, helping me grow as a prospective engineer and as a person.

Secondly, I am truly grateful to all the members of the PCOpt/NTUA team for providing a stimulating and supportive environment. I am wholeheartedly thankful to Dr. Marina Kontou, for her continuous presence and willingness to engage at every stage of the process. Her enthusiasm for problem-solving, openness to discussion, and sophisticated academic advice were invaluable throughout the development of this Thesis. I am deeply inspired by her passion, and truly grateful for the academic consultation she provided during this work.

Finally, I would like to thank my family—my parents and my brother—and my close friends for supporting me throughout my studies and offering unforgettable memories along the journey.

**National Technical University of Athens**
**School of Mechanical Engineering**
**Fluids Section**
**Parallel CFD & Optimization Unit**

# Convolutional Neural Networks as Surrogate Models in the Early Stages of the Automobile Design process

Diploma Thesis

**Faliakos Vasileios**

Advisor: Kyriakos C. Giannakoglou, Professor NTUA

Athens, 2025

## Abstract

The objective of this Diploma Thesis is the implementation of Artificial Neural Networks (ANNs), combining Convolutional Neural Networks (CNNs) and Deep Neural Networks (DNNs), as local data-driven surrogate models in the early stages of the automobile design process, to substitute costly steps followed in the established conventional CFD-based approach, and guide the design space exploration toward a more sensible direction.

In this work, Convolutional Neural Networks are employed to predict automobiles' aerodynamic drag directly from their sketch-like representations, thus bypassing the costly stages of 3D modeling, meshing and simulation. Starting from a pre-existing design, the models can assess geometric changes imposed on it, helping stylists to rapidly accept or reject potential modifications, based both on aesthetics and aerodynamic criteria. As a result, they can allow designers to narrow down the exploration domain early in the design process, and steer it in a sensible direction, thus preserving computational resources for high-fidelity optimization in later stages of the design.

The developed models derive from a sequence of applications of different objectives and ascending complexity. Since the 3D car geometries are going to be evaluated by their 2D representations, a key prerequisite is the development of an advanced CNN architecture that outperforms the equivalent conventional network in cost-effectiveness for 2D aerodynamic applications. This backbone configuration is developed through a foundational application concerning airfoils. Architectural

adaptations to extend this configuration to 3D car geometries are then examined, specifically three distinct approaches, differing either in their input shape or their architectural symmetry. The superior configuration is then selected and evaluated on a challenging dataset of automobiles and their corresponding drag force values.

The studies propose that statistically-driven adaptations and affordable fine-tuning can successfully lead to the development of local CNN surrogate models that evaluate cars with great precision in a cost-efficient manner, within a constrained space of the design domain.

**Εθνικό Μετσόβιο Πολυτεχνείο**
**Σχολή Μηχανολόγων Μηχανικών**
**Τομέας Ρευστών**
**Μονάδα Παράλληλης Υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης**

# Συνελικτικά Νευρωνικά Δίκτυα ως Μεταμοντέλα στα Πρώιμα Στάδια της Διαδικασίας Σχεδιασμού Αυτοκινήτων

Διπλωματική Εργασία

**Φαλιάκος Βασίλειος**

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2025

## Περίληψη

Ο στόχος αυτής της Διπλωματικής Εργασίας είναι η εφαρμογή Τεχνητών Νευρωνικών Δικτύων (ΤΝΔ), συνδυάζοντας Συνελικτικά Νευρωνικά Δίκτυα (ΣΝΔ) και Βαθειά Νευρωνικά Δίκτυα (ΒΝΔ), ως τοπικά μεταμοντέλα στα πρώιμα στάδια της διαδικασίας σχεδιασμού αυτοκινήτων, με σκοπό την αντικατάσταση κοστοβόρων βημάτων που ακολουθούνται κατά την συμβατική -βασιζόμενη στην ΥΡΔ- προσέγγιση, και την καθοδήγηση της εξερεύνησης του χώρου σχεδιασμού προς μια πιο ουσιώδη κατεύθυνση.

Κατά την προτεινόμενη προσέγγιση, τα συνελικτικά νευρωνικά δίκτυα χρησιμοποιούνται για την πρόβλεψη της αεροδυναμικής αντίστασης αυτοκινήτων απευθείας από διδιάστατες αναπαραστάσεις τους σε μορφή σκίτσου, παρακάμπτοντας έτσι τα κοστοβόρα στάδια της μοντελοποίησης, πλεγματοποίησης και προσομοίωσης με ΥΡΔ. Ξεκινώντας από ένα προϋπάρχον σχέδιο, τα μοντέλα μπορούν να χρησιμοποιηθούν για την αξιολόγηση γεωμετρικών αλλαγών που του επιβάλλονται, επιτρέποντας στους σχεδιαστές να αποδεχθούν ή να απορρίψουν ταχέως πιθανές τροποποιήσεις, με γνόμωνα τόσο την αισθητική όσο και τα αεροδυναμικά κριτήρια. Ως αποτέλεσμα, η χρήση τους δύναται να περιορίσει τον σχεδιαστικό χώρο από τα πρώιμα στάδια, διατηρώντας έτσι υπολογιστικούς πόρους για πιο υποσχόμενα σχέδια σε επόμενα στάδια του σχεδιασμού.

Τα αναπτυχθέντα μοντέλα προκύπτουν μέσα από μία ακολουθία εφαρμογών διαφορετικών στόχων και αυξανόμενης πολυπλοκότητας. Δεδομένου ότι οι 3Δ γεωμετρίες των αυτοκινήτων πρόκειται να αξιολογηθούν μέσω των 2Δ αναπαραστάσεών τους,

βασική προϋπόθεση είναι η ανάπτυξη μιας προηγμένης αρχιτεκτονικής που να υπερισχύει των συμβατικών αρχιτεκτονικών σε αποδοτικότητα και ακρίβεια, για δισδιάστατες αεροδυναμικές εφαρμογές. Αυτή η βασική αρχιτεκτονική αναπτύσσεται μέσω μιας θεμελιώδους εφαρμογής που αφορά αεροτομές. Στη συνέχεια, εξετάζονται αρχιτεκτονικές προσαρμογές της ώστε να επεκταθεί σε εφαρμογές $3\Delta$ γεωμετριών αυτοκινήτων, συγκεκριμένα τρεις διαφορετικές προσαρμογές, οι οποίες διαφέρουν είτε στο σχήμα εισόδου είτε στη συμμετρία της αρχιτεκτονικής. Η υπερισχύουσα διαμόρφωση επιλέγεται και αξιολογείται σε ένα απαιτητικό σύνολο δεδομένων αυτοκινήτων, ως προς τις τιμές της αεροδυναμικής τους αντίστασης.

Οι μελέτες δείχνουν ότι στατιστικά καθοδηγούμενες προσαρμογές και υπολογιστικά προσιτές διερευνήσεις μπορούν να οδηγήσουν στην ανάπτυξη εξειδικευμένων τοπικών διαμορφώσεων ΣΝΔ, ικανά να αξιολογούν γεωμετρίες αυτοκινήτων με ικανοποιητική ακρίβεια και χαμηλό υπολογιστικό κόστος, εντός ενός προκαθορισμένου πλαισίου του χώρου σχεδιασμού.

# Contents

# Chapter 1
# Introduction

## 1.1 Artificial Intelligence and Machine Learning

The field of Artificial Intelligence (AI) has evolved from a theoretical concept into a transformative force that is now integrated in nearly every aspect of modern engineering and industrial design. AI, broadly defined as the capability of machines to perform tasks that typically require human intelligence, includes a vast array of computational techniques designed to solve complex problems, recognize patterns, and make informed decisions [38]. Within this domain, Machine Learning (ML) has emerged as one of the most powerful and practical subfields, enabling systems to automatically learn and improve from experience without being explicitly programmed for every application [33].

The automotive industry, in particular, has witnessed a paradigm shift in how design processes are conceived, executed, and optimized. Traditional engineering approaches, while robust and well tested, often rely on iterative prototyping and extensive computational simulations that can be both time-consuming and resource-intensive. The integration of AI and ML techniques into automotive design workflows represents a significant opportunity to accelerate innovation, reduce development costs, and enhance the overall quality of vehicle systems.

Machine Learning, as a subset of AI, focuses on the development of algorithms that can identify patterns in data and make predictions or decisions based on them. The fundamental premise of ML is that systems can be trained to perform specific tasks by analyzing large datasets, extracting meaningful features, and developing mathematical models that generalize well to unseen data [6]. In a time of increased availability of computational power and exponential growth in data generation and storage capabilities, ML techniques can be applied to increasingly complex engineering problems. In the context of automotive design, this translates to the ability to process vast amounts of simulation data, experimental results, and operational feedback to develop more accurate and efficient design tools.

## 1.2 Types of Learning in Machine Learning

Machine Learning algorithms can be broadly categorized into three fundamental types of learning, each suited to different problem domains and data availability scenarios, summarized in Fig. [1.1].

**Supervised Learning** represents the most common and intuitive form of machine learning, where algorithms learn from labeled training data to make predictions on new, unseen examples. In supervised learning, the system is provided with input-output pairs during the training phase, allowing it to learn the mapping function that connects the two [16]. This approach is particularly well-suited for problems where empirical data with known outcomes is available.

**Unsupervised Learning** addresses scenarios where only input data is available without corresponding target outputs. These algorithms seek to discover hidden patterns, structures, or relationships within the data without explicit guidance about what to search [34]. Common unsupervised learning tasks include clustering, dimensionality reduction, and anomaly detection.

**Reinforcement Learning** takes a different approach by focusing on learning optimal actions through interaction with an environment. Rather than learning from static datasets, reinforcement learning agents receive feedback in the form of rewards or penalties based on their actions, gradually improving their decision-making capabilities through trial and error [44].

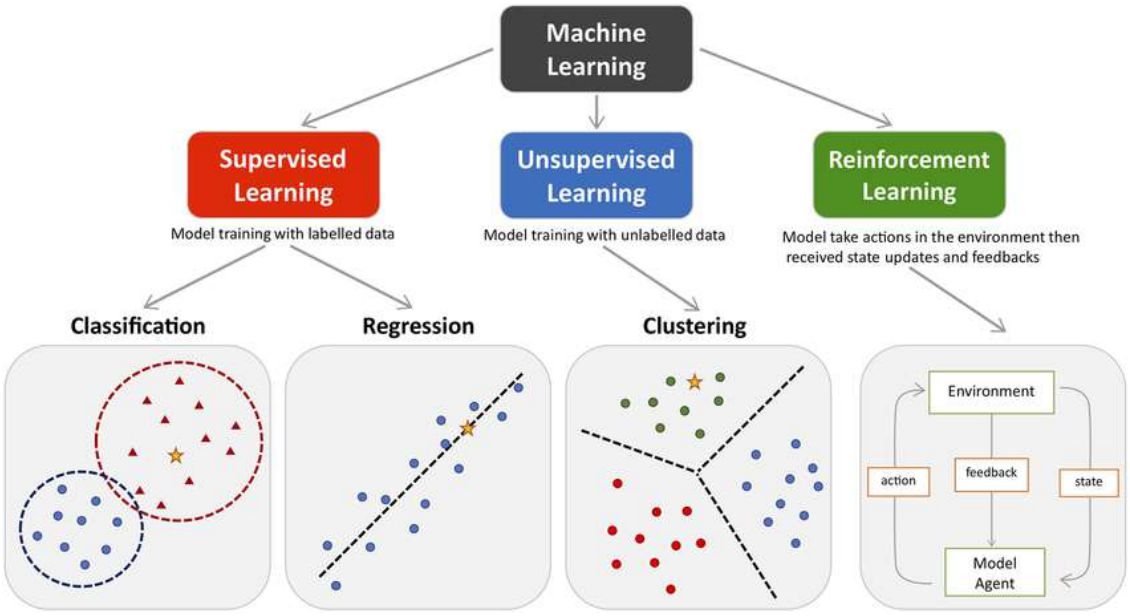



**Figure 1.1:** *The different types of learning in ML.*

## 1.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) represent one of the most versatile and powerful classes of supervised machine learning algorithms, inspired by the structure and function of biological neural systems [15]. The fundamental building block of a neural network is the artificial neuron, which receives multiple inputs, applies weights to these inputs, sums them together with a bias term, and passes the result through an activation function to produce an output [37]. This simple computational unit, when combined with many others, organized in layers in complex architectures, can approximate virtually any continuous function given sufficient data and appropriate training [22]. The structure of an ANN is graphically presented in Fig. [1.2]

ANNs are associated with either Regression or Classification tasks. Regression problems involve predicting continuous numerical values based on input features. In regression, the goal is to learn a mapping function that can estimate real-valued outputs with minimal error. Classification problems, in contrast, involve assigning input examples to discrete categories or classes. The objective is to learn decision boundaries that can accurately separate different classes in the feature space [6].

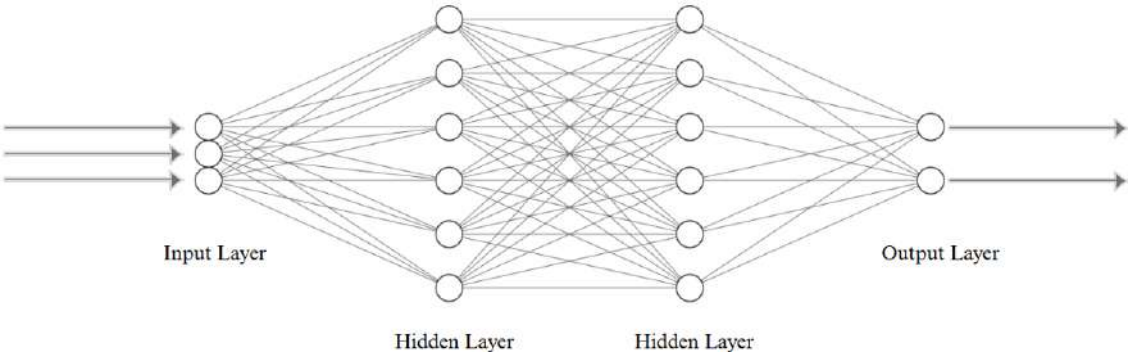



**Figure 1.2:** *Typical architecture of an ANN.*

The universal approximation capabilities of neural networks make them particularly effective in developing surrogate models in engineering applications. Neural networks can adapt their trainable parameters during training to capture the underlying physics and relationships present in the data and produce accurate estimations [28].

## 1.4 Motivation

The motivation for this Diploma Thesis stems from the inherent inefficiencies and substantial costs associated with the conventional design methodology employed throughout the automotive industry. This traditional approach follows a sequential, multi-stage process that has remained largely unchanged for years, despite its recognized limitations in terms of time consumption and resource allocation.

The conventional automotive design workflow can be characterized by the following sequential phases and graphically presented in Fig. [1.3]:

- The styling team initiates the process by developing a comprehensive set of design concepts, with primary emphasis placed on aesthetic appeal and visual impact rather than functional performance characteristics.
- The digital modeling team is tasked with translating these conceptual designs into detailed three-dimensional CAD models while incorporating necessary geometric constraints.
- An extensive CFD preparation phase follows, during which the 3D models undergo further geometric modifications to ensure computational compatibility. This stage involves geometric refinement and complex meshing procedures.
- Finally, high-fidelity high-cost CFD solvers are deployed to simulate the complex flow phenomena around the proposed vehicle geometries, extracting aerodynamic performance quantities such as drag coefficient, downforce characteristics and pressure distributions.

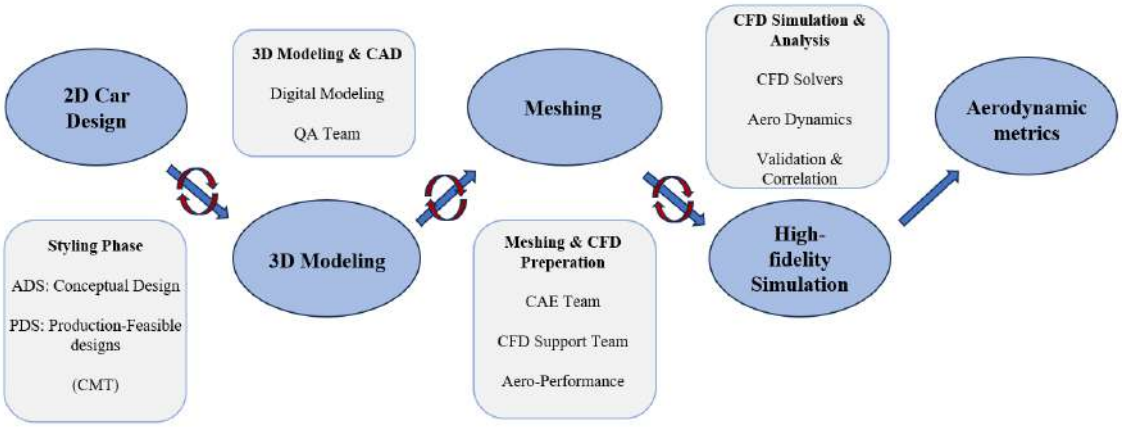



**Figure 1.3:** *Schematic representation of the conventional automotive design process workflow.*

This established methodology requires repeating the entire workflow for each design variant, creating significant bottlenecks in development timelines. The complexity increases when additional stakeholders such as marketing teams, project managers, and regulatory specialists contribute their requirements, extending evaluation periods and adding procedural overhead. Additionally, each team operates with different objectives and criteria, often creating conflicting requirements that demand iterative modifications. According to [4], (re)meshing accounts for approximately 25% of total CFD project time, while model pre-processing (including geometry cleanup) constitutes 35% of the workload. Moreover, the substantial computational resources needed for high-fidelity CFD simulations require careful planning, naturally limiting the frequency of design iterations. Additionally, communication delays and coordination difficulties further slow down design evaluation.

Notably, the automotive industry holds a valuable but underutilized resource: extensive databases containing conceptual designs, production vehicle geometries, and their aerodynamic performance data. This accumulated knowledge spans decades of design experience and experimental validation, offering an excellent foundation for advanced supervised machine learning approaches. These comprehensive datasets present an opportunity to develop predictive models that can accelerate the design process while maintaining acceptable accuracy, particularly during preliminary phases where absolute precision is less critical.

This Diploma Thesis proposes the integration of CNNs as sophisticated data-driven surrogate models for aerodynamic quantity regression. The approach aims to bypass the steps of 3D modeling, mesh generation, and numerical simulation by directly predicting aerodynamic properties from 2D design sketches.

Implementing such a methodology would enable real-time aerodynamic evaluation within styling workflows. This approach would provide immediate feedback on design modifications, allowing rapid exploration of the design space while preserving computational resources for detailed analysis on the most promising candidates.

## 1.5 Thesis Outline

Following the Introduction, this Thesis is organized as follows:

- **Chapter 2:** This chapter covers the fundamental architectural components and operating principles of DNNs, including their gradient-based optimization processes. Additionally, it presents advanced building blocks and techniques that significantly enhance model predictive accuracy and interpretational capabilities used throughout this work.

- **Chapter 3:** It provides a comprehensive overview of the software tools and methodologies employed to generate the working datasets (both input and output) for subsequent applications, as well as an algorithmic description of their application in recurring processes. Additionally, it introduces Evolutionary Algorithms, which are consistently utilized to fine-tune the architectural composition and component parameters of the developed models.

- **Chapter 4:** It is the first application of this work, focusing on predicting aerodynamic and geometric properties of 2D airfoils. It aims at the development of a superior CNN architecture that balances the trade-off between cost-efficiency and accuracy, serving as the foundational structure for models in Applications II and III. Additionally, it identifies ineffective regions within the evolutionary search domain, enabling their systematic exclusion to reduce computational requirements for subsequent, more complex applications.

- **Chapter 5:** This chapter investigates adaptive modifications to the established architecture from Application I, extending its application to aerodynamic 3D automotive geometry problems. It evaluates three distinct network configurations that differ primarily in input data structure and architectural symmetry. Through comprehensive analysis and performance comparisons, it identifies the optimal configuration for implementation in the final Application III.

- **Chapter 6:** Builds upon Application II with the primary distinction being increased dataset complexity. The objective of this Application is the accurate prediction of automobiles' aerodynamic Drag Force. The conducted studies emphasize the importance of implementing statistically-informed modifications within the training process to avoid excessive computational costs and address fundamental challenges in applications with limited (and statistically poor) datasets.

- **Chapter 7:** Synthesizes key findings from previous chapters and emphasizes on the adaptability of the selected configuration. Demonstrates that CNNs can effectively function as local surrogate models in automotive design applications, provided a proper architectural and compositional fine-tuning.

# Chapter 2
# Deep Neural Networks

## 2.1 Introduction

Neural Networks have emerged as a fundamental tool in modern ML, capable of approximating highly complex phenomena across diverse applications. The present chapter examines the core architectural components of a NN as well as their parameters, and offers a compact presentation of its working principle. Additionally, it presents the main working component exploited in this Thesis, the Squeeze and Excitation Block, which allowed the development of accurate and cost-efficient networks in the challenging subsequent implementations.

## 2.2 Network Architecture and Working Principle

Deep Neural Networks (DNNs) are a subcategory of ANNs. In order to be classified as a DNN, a network must have multiple hidden layers. The complexity of a DNN's architecture and neural computations allows for the better interpretation of complex patterns in the input data, making them successful in various tasks where shallow networks fail. The number and types of hidden layers in a DNN can vary, depending on the nature of the objective, the shape of the input data, the type of the desired output and the complexity of the task. Two of the most common categories of layers used in DNNs are dense layers and convolutional layers.

In *dense* (or *fully connected*) layers, each neuron receives information from all the neurons of the immediately preceding layer(s) and passes it on to all the neurons of the subsequent layer(s), through connections called synapses. For a single neuron, each synapse's information is multiplied by a weight uniquely associated with the

synapse, and then summed up. A unique bias associated with the neuron is added and an activation function is then applied, forming the output of the neuron, as shown in Fig. [2.1].

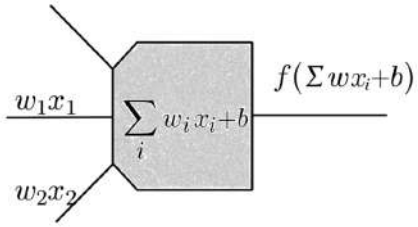



**Figure 2.1:** *Working principle of a neuron.*

*Convolutional layers* are the fundamental components of Convolutional Neural Networks (CNNs), a subcategory of DNNs, used in tasks associated with computer vision and image recognition. They are designed to adaptively identify spatial hierarchies of features present in their input data, which is typically an image or a set of feature maps. They operate by applying a set of learnable filters or kernels to their inputs. The fixed-sized filter slides across the input, overlapping regions of the data. The dot product of the filter and the local region is then computed, and this process continues until the entire input has been altered. This transformed representation of the input is then passed through an activation function to introduce non-linearity, forming the output of the layer, a feature map, extracting a specific pattern (Fig. [2.2]). Multiple filters result in multiple output feature maps (channels).

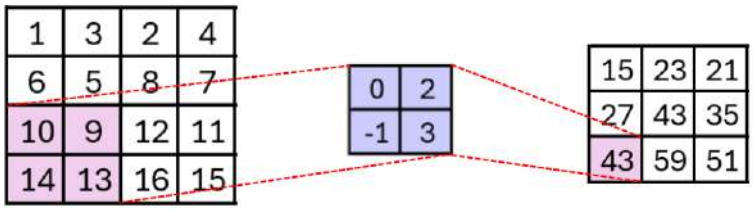



**Figure 2.2:** *Operation of a 2D convolutional layer.*

For a convolutional layer with $C$ filters and input $\mathbf{X} \in \mathbb{R}^{H' \times W' \times C'}$ (where $H', W'$ denote the input height and width respectively, and $C'$ denotes the input channels), let $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_C]$ be the filter bank where each $\mathbf{v}_c \in \mathbb{R}^{k \times k \times C'}$ is composed of 2D kernels $\mathbf{v}_c^s \in \mathbb{R}^{k \times k}$. The output channel $\mathbf{u}_c \in \mathbb{R}^{H \times W}$ is computed as:

$$\mathbf{u}_c = \sigma \cdot \left( \sum_{s=1}^{C'} \mathbf{v}_c^s * \mathbf{x}^s + b_c \right),$$

where $*$ denotes cross-correlation (commonly - and herein - implemented in deep learning frameworks instead of the traditional convolution operator), $b_c \in \mathbb{R}$ is the bias term, $\sigma$ denotes the activation function and $\cdot$ denotes its element-wise application. The layer's full output is formulated

$$\mathbf{U} = [\mathbf{u}_1, ..., \mathbf{u}_C] \in \mathbb{R}^{H \times W \times C} \tag{2.1}$$

Multi-dimensional inputs naturally pose a risk to the model's accuracy, as images or videos often contain noise, which obstructs the feature extraction process of the convolutional layers. In addition, the produced feature maps can be large in size, increasing the complexity of the model and posing a risk of overfitting. In order to address these challenges, convolutional layers are often used in conjunction with max pooling layers, which perform a downsampling operation on the data. The input of a max pooling layer is divided into non-overlapping regions of a predefined size, greater than a data unit, and the contained data points are replaced by the local region's maximum value. As a result, the spatial dimensions of the data are reduced while retaining the most prominent features, thus improving generalization and reducing the computational load. A demonstration of the max pooling operation is depicted in Fig. [2.3].

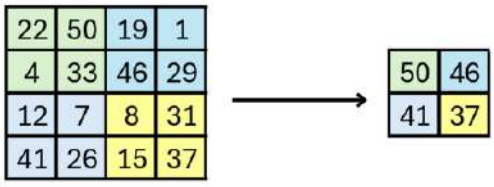



**Figure 2.3:** *Operation of a Max Pooling layer.*

## 2.3 Neural Network Training process

### 2.3.1 The gradient-based optimization problem

The training process of a Neural Network is an iterative gradient-based optimization problem, in which the trainable parameters of the network's components (weights, biases etc.) are adjusted in order to minimize a defined loss function's output. A typical Network's training process comprises the following steps:

1. **Trainable Parameters' Initialization:** The trainable parameters to be optimized are initialized. Proper initialization can potentially prevent the model from sticking at local minima in the training process [30].
2. **Forward Propagation:** The input data is fed into the input layer, and propagates through the various layers, transformed according to the current trainable parameters' values. This process continues until information reaches the output layer, where the final prediction is generated.
3. **Loss Value Calculation:** The output of the model is computed and compared to the true value with the use of a pre-selected loss function.
4. **Back Propagation:** After computing the loss value, the gradients of the loss w.r.t the network's trainable parameters are calculated using the chain rule of calculus. The calculated gradients indicate the direction and magnitude of the adjustments necessary to reduce the loss [6].
5. **Gradient - Based Optimization:** The adjustment of the trainable parameters is achieved through an optimization process based on the calculated gradients. Typically, gradient based optimization algorithms are used, with the most common being the Adam optimizer [18]. Optimizers have their own hyperparameters, such as the learning rate or the momentum, which play a crucial role in the convergence's speed and stability [15].
6. **Update of parameters:** The trainable parameters of the model are then updated accordingly.

The presented process is also called Forward Feed - Back Propagation.

Steps 2 to 6 are repeated for a number of iterations, called epochs, in which the entire training dataset is processed in batches. The batch size is a significant hyperparameter of the training process, referring to the number of training samples processed simultaneously in a single forward and backward propagation, and can greatly affect both the computational efficiency and the accuracy of the calculated gradients. Larger batch sizes can lead to steadier gradient estimates, but require more memory and often slow down convergence [43]. Smaller batch sizes can potentially prevent the model from sticking at local minima [19], but lead to more frequent updates on the trainable parameters, reducing the generalization of the tuning process and making convergence more unstable [8].

Concerning the effectiveness and quality of the model's training, there are two common problems that need to be monitored.

The first is the vanishing gradient problem; during the back-propagation process, the gradients of the loss function can become exceedingly small, resulting in minimal updates of the trainable parameters, and therefore impeding the network's ability to learn and adjust to the training dataset. This typically occurs when certain activation functions, such as sigmoid or tanh, are employed. Both functions resize input values into small ranges, but, most importantly, they saturate at extreme input values, and the calculated gradients from the training process are approaching machine precision. This problem can be prevented by applying activation functions without saturated regions, such as ReLU [14].

The second challenge concerns two major performance issues in ML; the phenomena of underfitting and overfitting. Overfitting can be described as critical lack of generalization during training, indicating that the network failed to recognize the essential features of the input, but rather captures noise and random fluctuations [50].

An overfitted model performs well on the training data but fails to adjust to unseen data. On the contrary, an underfitted model is either too simplistic or inadequately trained to capture the necessary patterns in the input data, resulting in high errors both in the seen and unseen data [Fig. 2.4].

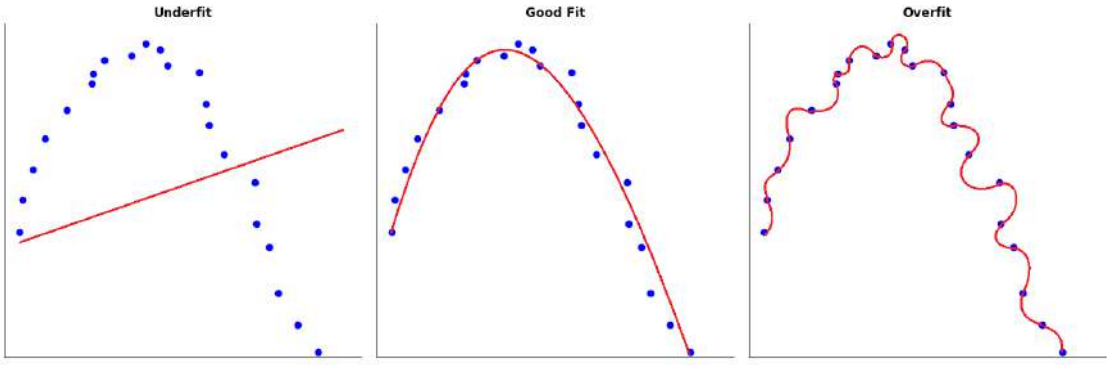



**Figure 2.4:** *(left) Performance of an underfitted model that fails to predict data. (middle) Performance of a well fitted model that captures the patterns in the dataset. (right) Performance of an overfitted model that captures training data but fails to generalize on unseed data.*

Traditionally, to overcome these issues, the validation technique is implemented, according to which a chunk of the dataset is isolated and not fed into the network except for when a complete pass of the training dataset has been performed (end of each epoch). The goal is to monitor how well the model generalizes to unseen data by observing the validation loss [35].

### 2.3.2 Activation Functions

Activation functions are crucial components of ANNs, introducing non-linearity into the model. Without them, a network would perform similar to a single-layer linear model, and fail to predict complex phenomena [21]. Let a neural network be defined as a function $f : \mathbb{R}^k \to \mathbb{R}^m$. For a neuron in layer $\ell$, with input $\mathbf{x}^{(\ell)} \in \mathbb{R}^{n_{\ell-1}}$ (where $n_0 = k$), weights $\mathbf{w}^{(\ell)} \in \mathbb{R}^{n_{\ell-1}}$, bias $b^{(\ell)} \in \mathbb{R}$ and activation function $\phi : \mathbb{R} \to \mathbb{R}$, its output is computed as:

$$y^{(\ell)} = \phi\left((\mathbf{w}^{(\ell)})^\top \mathbf{x}^{(\ell)} + b^{(\ell)}\right)$$

Below follows a presentation of the most common activation functions used in NNs, which are included in the parametric explorations in the subsequent Applications.

· *ReLU*: Rectified Linear Unit is a piecewise linear activation function commonly used in machine learning tasks. It maps all negative inputs to zero, reducing unnecessary computations. However, this introduces the risk of neurons being inactive and not contributing to the learning process, often referred to as the "Dying ReLU Problem"

$$ReLU(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

· *GELU*: Gaussian Error Linear Unit is a smoother approximation of ReLU, filtering the neuron's input by its probability under a Gaussian distribution rather than its sign. Additionally, GELU has a continuous gradient in comparison to ReLU's, which has piecewise continuity. All in all, GELU outperforms ReLU in deep architectures.

$$GELU(x) = \frac{x}{2}(1 + erf(\frac{x}{\sqrt{2}})) = \frac{x}{2}(1 + \frac{2}{\sqrt{\pi}} \int_0^{x/\sqrt{2}} e^{-t^2}\, dt)$$

· *Leaky ReLU*: Leaky ReLU is nearly identical to ReLU, with the only difference being the presence of a non-zero slope for negative input values. The latter addresses the dying ReLU problem and prevents dead neurons, while retaining most of ReLU's advantages. Parameter $\alpha$ is set to a small value, typically 0.01.

$$Leaky\ ReLU(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

· *ELU*: The Exponential Linear Unit activation function behaves identically to ReLU and Leaky ReLU for positive inputs. Unlike them, ELU maintains its smoothness everywhere, improving optimization and training stability. Additionally, ELU pushes the mean activation closer to zero, which can reduce vanishing or exploding gradients and improve learning speed. Parameter $a$ is typically set to 1.

$$ELU(x) = \begin{cases} x & \text{for } x > 0 \\ \alpha(e^x - 1) & \text{for } x \leq 0 \end{cases}$$

· *Sigmoid*: The Sigmoid activation function is a smooth, differentiable function that projects all inputs to the range (0,1). However, it is prone to the vanishing gradient problem, since it saturates near the boundaries of it's domain of definition.

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

· *Tanh*: The Hyperbolic Tangent activation function maps any real inputs to the range (0,1). Compared to the sigmoid function, tanh is often preferred due to its steeper gradients, which can accelerate training and allow better weight optimization. However, it is still prone to the vanishing gradient problem.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

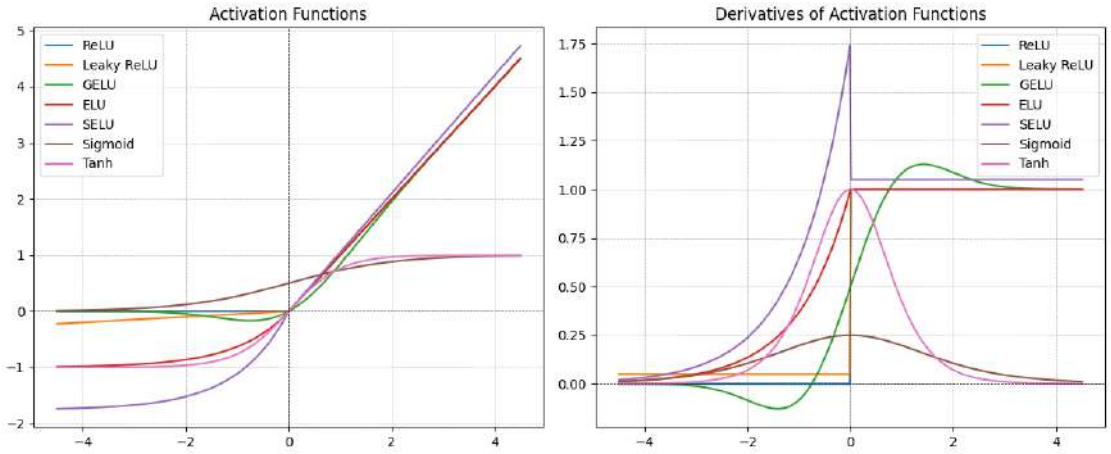



**Figure 2.5:** *Behavior of the presented activation functions (left) and their derivatives (right).*

### 2.3.3 Loss functions

Loss functions are mathematical functions quantifying the discrepancy between the prediction provided by the model to the ground truth. Essentially, they serve as the objective function of the training process optimization problem, whose gradient indicates how the trainable parameters should be adjusted to improve accuracy. The most common loss functions in regression problems are the the *Mean Absolute Error* loss function *(MAE)* and the *Mean Squared Error* loss function *(MSE)*, formulated below accordingly.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$N$ is the total number of samples, $y_i$ with $i = 1, 2, ..., N$ is the target value of each sample and $\hat{y}_i$ is the corresponding prediction of the network.

*MAE* is less sensitive to outliers and anomalous points [48], treating all errors equally, and provides a measure that is easy to interpret in terms of the actual units. However, its undefined gradient at zero can pose challenges to the optimization process of training. On the other hand, *MSE* has a smooth gradient at all points, allowing for a more stable optimization. It penalizes large errors heavily due to the squaring of the error, which is often desirable in regression tasks. On the downside, this leads to increased sensitivity to outliers, sometimes sabotaging the model's ability to generalize properly. The selection between *MAE* and *MSE* depends both on the dataset and the nature of the problem [9]. When working with a noise-free dataset with few outliers, *MSE* can often aid the construction of an accurate (and efficient) model. When the working dataset is of moderate size and noise and contains outliers, selecting the *MAE* loss function will allow for a more robust and accurate model.

In this Diploma Thesis, two additional loss functions are employed, the *Mean Relative Error* (*MRE*) and the *Mean Absolute Relative Error* (*MARE*), defined respectively as:

$$\text{MRE} = \frac{1}{N}\sum_{i=1}^{N}\frac{y_i - \hat{y}_i}{y_i} \tag{2.2}$$

$$\text{MARE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - \hat{y}_i|}{|y_i|} \tag{2.3}$$

These metrics are particularly advantageous when the target values exhibit small variance, as relative errors normalize deviations by the target magnitude [7].

### 2.3.4 The Adam Optimizer

The Adaptive Moment Estimation Optimizer (Adam Optimizer) [25], is an adaptive gradient-based optimization algorithm that combines the benefits of two earlier optimization methods, the Adaptive Gradient Algorithm (AdaGrad) [10] and Root Mean Square Propagation (RMSProp) [45]. It is a versatile optimization algorithm that offers several advantages over conventional gradient-based optimization methods, by balancing adaptivity and convergence speed. Adam is used throughout this entire Thesis as the optimizer of the models' training.

The fundamental principle of Adam is the computation of individual adaptive learning rates for different parameters, based on estimates of both first-order and second-order moments of the gradients.

Let $f(\theta)$ be the objective function to be minimized, where $\theta$ denotes the (vector of) parameters to be optimized, herein the model's trainable weights. At each timestep $t$, the gradient of the function w.r.t $\theta$ is computed:

$$g_t = \nabla_\theta f_t$$

Adam maintains and updates the exponential moving averages of the gradient $m_t$ (estimate of the first moment of the gradient) and the squared gradient $\upsilon_t$ (estimate of the second moment of gradient) as follows

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$\upsilon_t = \beta_2 \upsilon_{t-1} + (1 - \beta_2) g_t^2$$

$\beta_1, \beta_2 \in [0, 1)$ are the hyperparameters controlling the exponential decay rates of the gradient and squared gradient moving averages respectively. Typically, $\beta_1$ is set to 0.9 and $\beta_2$ is set to 0.999. Since the moving averages are initialized at zero, they are biased toward zero, particularly during the initial training steps. To counteract this, the Adam optimizer calculates the bias-corrected moment estimates $\hat{m}_t$ and $\hat{\upsilon}_t$.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$
$$\hat{\upsilon}_t = \frac{\upsilon_t}{1 - \beta_2^t}$$

Finally, the trainable parameters $\theta$ are updated:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{\upsilon}_t} + \epsilon}$$

where $\alpha$ is the learning rate and $\epsilon$ is a small value (typically set to $10^{-8}$) to prevent division by zero and enhance numerical stability. Division by the term $\sqrt{\hat{\upsilon}_t}$ ensures an appropriately scaled parameter update based on the past gradients, allowing for an adaptive step size across different parameters.

Adam is computationally efficient and requires only first-order derivatives, making it well-suited for large-scale optimization problems. In addition, the inclusion of momentum accelerates convergence in the direction of consistent gradient descent, preventing oscillations in curved loss surfaces. It is the most commonly used optimizer for machine learning tasks and neural networks' training.

## 2.4 Squeeze-and-Excitation Blocks

Squeeze-and-Excitation Blocks (SE Blocks) [23], are building units that mimic biological vision, in which more attention is given to more salient stimuli [47]. They are designed to enhance the representational power of CNNs, configuring channel-wise feature responses and thus allowing the network to emphasize on informative feature maps and suppress less useful ones. SE Blocks are the fundamental component

of this Thesis's networks and played a critical role to their accuracy and efficiency.

Consider the output of a Convolutional Layer of Eq. (2.1). An SE block enhances conventional convolution by forming channel-wise feature responses through three consecutive operations: *squeeze*, *excitation*, and *scale*, collectively denoted as $\mathbf{F}_{SE}$.

- **Squeeze Operation:** This stage is responsible for aggregating global spatial information across each channel. It employs Global Average Pooling (GAP) to generate channel-wise statistics, contained in a channel descriptor. The squeeze operation $\mathbf{F}_{sq} : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^C$ transforms the feature tensor $\mathbf{U}$ into the channel descriptor $\mathbf{z} \in \mathbb{R}^C$:

$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j)$$

  where $u_c(i, j)$ represents the value at position $(i, j)$ in the $c$-th channel of the block's input. Each statistic $z_c \in \mathbb{R}$ possesses a global receptive field and provides a representation of the entire spatial extent of the corresponding channel.

- **Excitation Operation (Adaptive Recalibration):** After the squeeze operation, the goal is to capture inter-channel dependencies and generate a set of modulation weights. This is achieved through a gating mechanism parameterized by a two-layer Multi-Layer Perceptron (MLP) with a bottleneck structure.

  The excitation transformation $\mathbf{F}_{ex} : \mathbb{R}^C \rightarrow \mathbb{R}^C$ is formulated:

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma(\mathbf{W}_2 \delta(\mathbf{W}_1 \mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2)$$

  where $\mathbf{W}_1 \in \mathbb{R}^{\frac{C}{r} \times C}$ and $\mathbf{W}_2 \in \mathbb{R}^{C \times \frac{C}{r}}$ are the weight matrices of the first and second fully connected layers respectively, $\mathbf{b}_1 \in \mathbb{R}^{\frac{C}{r}}$ and $\mathbf{b}_2 \in \mathbb{R}^C$ are the corresponding bias vectors, $\delta$ refers to a non-linear activation function (typically ReLU), $\sigma$ denotes the sigmoid activation function and $r \geq 1$ is the reduction ratio hyperparameter controlling block capacity and computational cost (typically set to 16). The first transformation $\mathbf{W}_1 \mathbf{z}$ reduces dimensionality from $C$ to $\frac{C}{r}$, constraining model capacity and computational complexity. The subsequent expansion via $\mathbf{W}_2$ restores the original channel dimensionality. The ReLU activation $\delta$ allows for the capture of non-linear inter-channel dependencies and the sigmoid activation $\sigma$ ensures that the weights $s_c \in [0, 1]$, where $s_c \approx 0$ indicates channel suppression and $s_c \approx 1$ indicates channel emphasis.

- **Scale Operation:** Finally, the scale operation applies the learned channel-

wise weights to the original feature maps through element-wise multiplication.

$$\tilde{\mathbf{x}}_c = \mathbf{F}_{scale}(\mathbf{u}_c, s_c) = s_c \odot \mathbf{u}_c$$

where $\mathbf{u}_c \in \mathbb{R}^{H \times W}$ is the original feature map, and $\tilde{\mathbf{x}}_c \in \mathbb{R}^{H \times W}$ represents the recalibrated feature map.

The complete SE block transformation can be expressed as the composition:

$$\tilde{\mathbf{X}} = \mathbf{F}_{SE}(\mathbf{U}) = \mathbf{F}_{scale}(\mathbf{U}, \mathbf{F}_{ex}(\mathbf{F}_{sq}(\mathbf{U})))$$

where $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots, \tilde{\mathbf{x}}_C] \in \mathbb{R}^{H \times W \times C}$ represents the final recalibrated feature tensor.

The SE block introduces little computational overhead while providing substantial performance improvements. Briefly, the additional trainable parameters introduced are:

$$\mathrm{P}_{SE} = \underbrace{\overbrace{C \times \frac{C}{r}}^{\mathbf{W}_1} + \overbrace{\frac{C}{r}}^{b_1}}_{\text{FC1}} + \underbrace{\overbrace{\frac{C}{r} \times C}^{\mathbf{W}_2} + \overbrace{C}^{b_2}}_{\text{FC2}} = \frac{2C^2}{r} + \frac{C}{r} + C \tag{2.4}$$

The additional floating-point operations (FLOPs) per forward pass are:

$$\mathrm{FLOP}_{SE} = \underbrace{C \times H \times W}_{\text{GAP}} + \underbrace{C \times \frac{C}{r} + \frac{C}{r}}_{\text{FC1}} + \underbrace{\frac{C}{r} \times C + C}_{\text{FC2}} + \underbrace{C \times H \times W}_{\text{Scale}}$$

$$= 2 \times C \times H \times W + \frac{2C^2}{r} + 2C \tag{2.5}$$

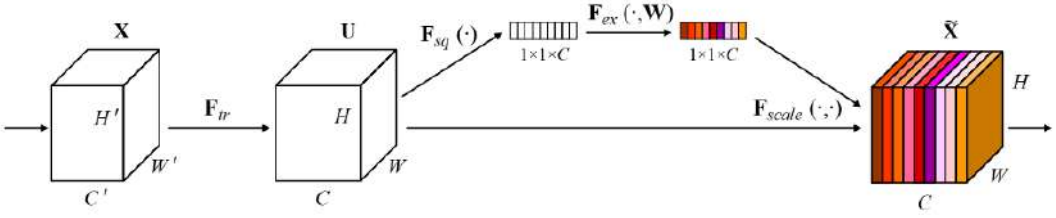



**Figure 2.6:** *Configuration of a Squeeze-and-Excitation Block. Image taken from [23]*

This series of operations allows models to focus on the most informative feature maps and controls the influence of those less relevant. Integrating SE blocks into existing CNN architectures has been shown [23] to significantly improve performance while imposing minimal additional computational cost, leading to more robust and efficient performances. As a result, it is common to integrate SE Blocks in pre-existing model schemes, e.g. the ResNet module [17], forming the SENet or SE-ResNet model (Fig. [2.7]) which achieved 1st place in the ILSVRC 2017 classification competition [24].

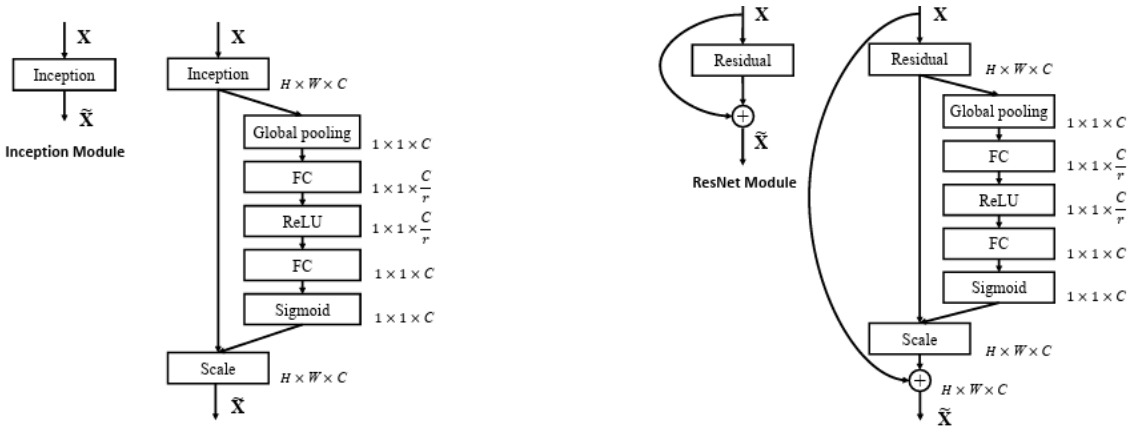



**Figure 2.7:** *Integration of SE Blocks in the original Inception module (left). The ResNet module, forming the SENet or SE-ResNet module (right). Image taken from [23].*

## 2.5 Regularizers

Regularization is a fundamental technique in ML applications, implemented to prevent overfitting and aid generalization on unseen data [6]. Regularization is often employed in complex applications with large sets of trainable parameters, to ensure a robust performance.

Consider a network with weights $\mathbf{w} \in \mathbb{R}^d$. Regularization modifies the model's loss function, introducing a regularization term that penalizes large weight magnitudes.

$$\mathcal{L}_{reg}(\mathbf{w}) = \mathcal{L}(\mathbf{w}) + \lambda \cdot \Omega(\mathbf{w})$$

where $\mathcal{L}$ is the original loss function, $\Omega$ is the regularization term and $\lambda \geq 0$ is a hyperparameter called regularization strength. In this Thesis, the two most common regularization techniques are presented and implemented, *Lasso* or *L1 Regularization* and *Ridge* or *L2 Regularization*. Both introduce a penalty term derived from the norms of the model's weights [15].

- **L2 Regularization (Ridge Regression / Weight Decay)** penalizes the squared magnitude of weights, with the regularization term:

  $$\Omega(\mathbf{w}) = ||\mathbf{w}||_2^2 = \sum_{i=1}^{d} w_i^2 \quad \text{with} \quad \nabla_{\mathbf{w}}(\lambda\Omega) = \nabla_{\mathbf{w}}(\lambda||\mathbf{w}||_2^2) = 2\lambda\mathbf{w}$$

  During optimization in the training process, say with the gradient descent, the weights are updated:

  $$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\nabla\mathcal{L}(\mathbf{w}_t) + 2\lambda\mathbf{w}_t)$$

  where $\eta$ is the learning rate. This update shrinks the trainable weights proportionally to their magnitude. Application of L2 drives the network to distribute learned information across more parameters, leading to a reduced sensitivity to noise and a better ability to generalize.

- **L1 regularization (Lasso Regression)** penalizes the absolute magnitude of the weights, with the regularization term:

$$\Omega(\mathbf{w}) = ||\mathbf{w}||_1 = \sum_{i=1}^{d} |w_i| \quad \text{with} \quad \nabla_{\mathbf{w}}(\lambda\Omega) = \nabla_{\mathbf{w}}(\lambda||\mathbf{w}||_1) = \lambda \cdot sign(\mathbf{w})$$

where $sign(\mathbf{w})$ is undefined at zero. During optimization in the training process with the gradient descent, the weights are updated:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\nabla\mathcal{L}(\mathbf{w}_t) + \lambda \cdot sign(\mathbf{w}_t))$$

Due to the non-differentiability at zero, proximal methods or soft-thresholding operators are employed at $w_i = 0$.

Unlike L2, L1 regularization can drive some weights exactly to zero, performing feature selection, which is particularly useful in high-dimensional problems where many features could be irrelevant.

In practice, it is common to implement a hybrid approach, combining both regularizations (Elastic Net) [51], modifying the loss function according to:

$$\mathcal{L}_{reg}(\mathbf{w}) = \mathcal{L} + \lambda_1||\mathbf{w}||_1 + \lambda_2||\mathbf{w}||_2^2$$

# Chapter 3
# Implementation Practice

## 3.1 Introduction

This chapter outlines the methodology employed in all three subsequent Applications. Initially, it introduces the in-house software and techniques employed across the case studies for dataset generation, flow simulation and model fine-tuning (in this order). It then provides a comprehensive description of the procedural pipeline followed, and presents fundamental setups that are consistently reapplied and implemented, thereby eliminating redundancy and ensuring legibility.

## 3.2 Free-Form Deformation and Morphing Boxes

Free-Form Deformation (FFD) is a geometric modeling technique that allows the manipulation of shapes through the control of an underlying lattice structure [41]. A set of Control Points (CPs) is arranged in such a way that a cubical grid is defined, fully enclosing the geometry to be deformed. This control grid serves as a parameterization framework for the geometry; the initial positions of the CPs define the undeformed state of the geometry, while displacement of the control points is linearly propagated to the underlying geometry, altering its shape.

In [41], the working scheme parameterizing the deformation is based on trivariate Bernstein polynomials. Consider a point $p = (x, y, z)$ in the original (undeformed) space. It's deformed position $p' = (x', y', z')$ is calculated using a trivariate tensor product of Bernstein polynomials as:

$$p' = \sum_{i=0}^{n} \sum_{j=0}^{m} \sum_{k=0}^{l} B_i^n(u) B_j^m(v) B_k^l(w) P_{ijk}^{CP}$$

where $P_{ijk}^{CP}$ are the CPs forming the FFD regular control grid, $(u, v, w)$ are the local coordinates of $p$ within the unit cube of the control lattice and $B_i^n$ denotes the

Berstein polynomial of degree n:

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

Movement of a control point only affects the embedded geometry locally, to an extent dictated by the Bernstein basis. Specifically, if a CP $P_{ijk}^{CP}$ is moved, the change in the geometry is restricted to the region where the corresponding basis functions are non-zero. This locality of the propagated influence makes Free-Form Deformation a very powerful tool in engineering applications like shape optimization and design exploration. Additionally, FFD can be extended beyond cubic lattices and incorporate more sophisticated basis functions for greater control and precision [29].

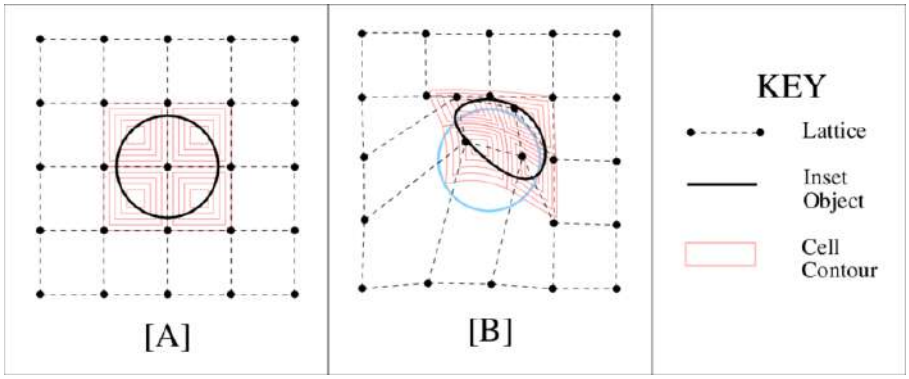



**Figure 3.1:** *Stages of Free-Form Deformation. [A] Pre-deformation stage, depicting the object fully enclosed in the control lattice in its undeformed state. [B] Post-deformation: Displacement of the CPs caused a local deformation to the inset object. Image taken from [12]*

In this Thesis, *Morphing Boxes* are employed. Following the core concept of FFD, Morphing Boxes use splines to parameterize the deformation and do not necessarily require the geometry-to-be-deformed to be fully embedded within the control grid. Specifically, in Applications II and III of this Thesis, only predefined regions of the geometry are enclosed by the lattice.

The method of morphing boxes is implemented through the PUMA software [2], which offers an application of the FFD methodology based on Non Uniform Rational B-Splines (NURBS). Each time the NURBS lattice's control points are displaced, the geometry encapsulated by the grid is deformed and the CFD mesh is adapted to it. This allows for a time-efficient creation of a diverse dataset.

The displacement of the control points is performed using the Latin Hypercube Sampling (LHS) method. LHS is a popular statistical method used in Design of Experiments (DOE) applications to generate near-random samples from a mul-

tidimensional space, ensuring a uniform and representative distribution of points [32]. In a n-dimensional problem, the core concept of LHS is to divide each dimension of the n-dimensional design space into equal, non-overlapping intervals (strata) and ensure that each stratum is sampled exactly once, creating a stratified sampling scheme that minimizes clustering of the sample points and guarantees a well-distributed sample set. Application of the LHS method allows for the creation of a representative dataset without the need for excessive sampling. It has been demonstrated [31] that applying LHS allows for more efficient sampling of the design space, relative to other popular methods like the Monte Carlo sampling.

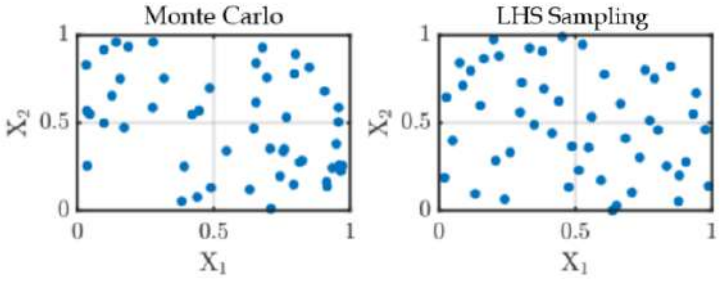



**Figure 3.2:** *Comparison of the Monte Carlo and LHS Sampling techniques with the same sample size, demonstrating LHS's ability to cover the design space more evenly. Image taken from [40]*

## 3.3 The PUMA CFD Solver

All flow simulations are performed employing the in-house GPU-accelerated Parallel Unstructured Multirow Adjoint (PUMA), an advanced CFD solver developed by the PCOpt/NTUA [2], [46]. It is designed to leverage the computational capabilities of GPUs, allowing for efficient and accurate simulations of complex fluid flows, particularly in turbomachinery applications.

PUMA solves numerically the 3D Reynolds-Averaged Navier-Stokes equations (RANS) for compressible and incompressible fluids. In this work, the incompressible variant is used (the pseudo compressible for Application I). The flow and the RANS adjoint equations are discretized on unstructured/hybrid meshes using the vertex-centered finite volume method.

The steady residuals of the viscous flow for an incompressible fluid read

$$R_n = \frac{\partial f_k^{inv}}{\partial x_k} - \frac{\partial f_k^{vis}}{\partial x_k} = 0$$

where $f_k^{inv}$ and $f_k^{vis}$ are the inviscid and viscous fluxes respectively, formulated:

$$f_k^{inv} = \begin{bmatrix} \rho\upsilon_k \\ \rho\upsilon_k\upsilon_1 + p\delta_{1k} \\ \rho\upsilon_k\upsilon_2 + p\delta_{2k} \\ \rho\upsilon_k\upsilon_3 + p\delta_{3k} \\ \rho\upsilon_k h_t \end{bmatrix} \qquad f_k^{vis} = \begin{bmatrix} 0 \\ \tau_{1k} \\ \tau_{2k} \\ \tau_{3k} \\ \upsilon_l\tau_{lk} + q_k \end{bmatrix}$$

In this notation, $\rho, p, \upsilon_k$ and $h_t$ are the fluid's density, pressure, velocity components and total enthalpy respectively. $\delta_{kkm}$ denotes the Kronecker symbol. The viscous stress tensor is given by:

$$\tau_{km} = \mu(\frac{\partial\upsilon_k}{\partial x_m} + \frac{\partial\upsilon_m}{\partial x_k} - \frac{2}{3}\delta_{km}\frac{\partial\upsilon_l}{\partial x_l})$$

where $\mu$ is the fluid's bulk viscosity and $q_k$ is the heat flux. PUMA includes the inviscid, laminar and viscous flow models, with all computations performed with a second-order accuracy. In the case of a turbulent flow, PUMA allows the application of a variety of turbulence models; herein the Spalart-Allmaras turbulence model is employed.

PUMA offers a high parallel efficiency in both the flow and the adjoint solvers by using the Mixed Precision Arithmetics (MPA) [2], which reduces memory usage and memory transactions between the GPU threads and the device memory with no effects on the code's accuracy.

## 3.4 Evolutionary Algorithms and the EASY Software

Evolutionary Algorithms (EAs) are population-based optimization methods inspired by biological evolution. These algorithms iteratively improve a set of candidate solutions (chromosomes) by evaluating their performance against an objective function. The best-performing solutions are selected for reproduction through crossover operations, while mutation introduces controlled variations to explore new regions of the design space [11]. This stochastic approach allows EAs to escape local optima, making them particularly effective for complex, nonlinear optimization

problems where gradient-based methods often struggle.

EASY (Evolutionary Algorithms System) is a versatile optimization framework developed by the PCOpt/NTUA [1]. It implements these evolutionary principles while incorporating advanced features such as distributed computing and surrogate modeling [27]. These capabilities make EASY well-suited for computationally demanding engineering optimization tasks, including cases where explicit mathematical formulations are unavailable or impractical.

In this Thesis, EASY is employed for single-objective, unconstrained optimization, specifically for (hyper)parameter tuning of the developed models and their training. Its adaptive search strategy ensures efficient convergence while maintaining flexibility across different problem configurations.

## 3.5 Procedural Pipeline

The first part of this subsection concerns the established methodology that is followed across the subsequent Applications, to generate the corresponding working datasets. Essentially, it comprises two steps; the generation of a dataset from a baseline geometry, and (in Applications II and III) the pre-processing of the samples to match a desired style for the context of this Thesis.

The second part regards the EASY software's design space and general setup in the parametric studies conducted in each case, concerning the shape and composition of the examined network configurations.

A major downside of data-driven models is the restriction of their output predictions within a constrained design space established by the training dataset's bounds. This constraint introduces an inherent locality to these models; regardless of their accuracy and computational efficiency, their utility is restricted to specific regions of the design space. This limitation establishes a practical ceiling on the computational resources that can be justifiably expended to model development and training, creating a cost-effectiveness trade-off that must be taken into consideration before establishing any recurring processes.

A surrogate model's total cost is calculated as the sum of the dataset generation cost (LHS, FFD, CFD computations to obtain ground truth, Image preprocessing in Applications II-III), the training cost (depends on network trainable parameters and hyperparameters of training) and the fine-tuning cost (EASY domain of exploration

and setup).

### 3.5.1 Methodology Overview

In a real-world industrial setting, an automotive company would typically store a database of sketches and variants of conceptual or existing designs, along with their corresponding performance data. These stylistically similar records (assuming they were designed by the same styling team) could be directly used to train this Thesi's models. Herein, synthetic datasets are generated, following the sequence of steps presented below.

Initially, NURBS Morphing Boxes are defined at certain position(s) of a baseline geometry. Their CPs are displaced according to the LHS method, creating $N_{db}$ instances, and the FFD technique is applied to deform the underlying geometry, resulting in the generation of $N_{db}$ variations of the baseline geometry.

The output dataset of all cases is obtained via CFD simulations performed by PUMA. The calculated properties/forces are considered the ground truth. Key mesh and flow parameters for each case are summarized in tables within their respective introductory subsections.

Lastly, in Applications II and III, filters are applied to images of the various geometries to mimic the desired sketch-like style and resemble industrial concept drawings.

Since this algorithm relies on expensive CFD simulations, the sample size $N_{db}$ for each case should remain limited; excessive sampling and flow solving would contradict the primary goal of creating a computationally efficient local surrogate model and undermine its importance. $N_{db}$ is defined 100 in Applications I, II and 366 in Application III.

### 3.5.2 EASY setup for fine tuning

In subsequent Applications, certain hyperparameters of the models' training process are adjusted via Trial-and-error studies. However, the different network configura-

tions are examined entirely by EASY, which is used for cost-efficient, case-dependent fine tuning, rather than overall model optimization.

In all cases, the models are parameterized by:

- The total Number of Convolutional Layers
- The filter size for each Convolutional Layer
- The activation function applied by Convolutional Layers (shared)
- The total Number of fully connected Layers
- The neuron size for each fully connected Layer
- The activation function applied by fully connected Layers (shared)

and EASY evaluates distinct configurations. The following setups are implemented in each case, unless specified otherwise in a subcase.

| | | **Application I** | **Application II** | **Application III** |
|---|---|---|---|---|
| **Exploration Domain** | Number of CNN Layers | Up to 8 (*ceil*) | | |
| | Number of CNN Layers | Up to 8 (*ceil*) | | |
| | CNN $L_1$-$L_{ceil}$ filter size (pow. of 2) | Up to 9 | | |
| | DNN $L_1$-$L_{ceil}$ neuron size (pow. of 2) | Up to 9 | Up to 12 | Up to 12 |
| | Activation function CNN & DNN Layers | ReLU, GELU, tanh, sigmoid, SELU, ELU, Leaky ReLU | ReLU, GELU, tanh, sigmoid | |
| | SE Block Integration | – | Implicitly via empirical rules | |
| **EAs Setup** | EASY Evaluations* | 125 | 250 | 250 |
| | Population Size* | 30 | 30 | 45 |
| | Parent:Offspring ratio | 3 | | |
| | Elite Population size | 15 | | |

**Table 3.1:** *Summary of the EASY framework's setup in the subsequent Applications. (*) indicates that the presented values may vary, if specified so in the corresponding subsection.*

The use of EASY allows for the proper adjustment of the networks' architectural composition and component parameters, unlocking and demonstrating the true capabilities of the examined configurations. During fine-tuning, the kernel size as well as the stride length of the 2D Convolutional layers have predefined shapes $(3, 3)$ and $(1, 1)$ respectively and remain fixed. The same applies to the pool size in the MaxPooling layers, which is fixed to the shape $(2, 2)$.

# Chapter 4
# Application I - Isolated Airfoil Properties Prediction

## 4.1 Introduction

Application I focuses on predicting airfoils' geometric and aerodynamic properties (area, lift coefficient $C_L$ and drag coefficient $C_D$) using CNNs. The primary objective is the development of a general CNN architecture that achieves superior cost-effectiveness in 2D aerodynamic-governed phenomena, when compared to conventional approaches.

Additionally, a second, implicit goal involves investigating the network's structural components and hyperparameters. This exploration aims to constrict EASY's domain of exploration in subsequent Applications II and III, which naturally demand greater computational resources and involve more training parameters.

As a result, Application I is essentially a foundational study for the development of the final 3D car drag prediction network, focusing both in accuracy and computational efficiency (both in training and fine-tuning).

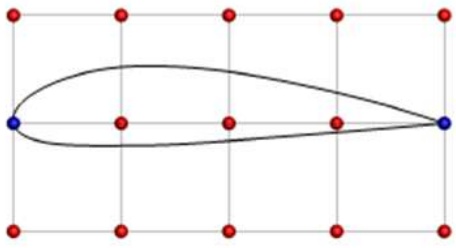


**Figure 4.1:** *The NACA4318 airfoil geometry embedded in the* $5 \times 3$ *NURBS box. Blue CPs are fixed while red CPs are allowed to move.*

The baseline geometry (NACA4318 Isolated Airfoil) is controlled by the 5 x 3 NURBS box of Fig. [4.1]. The blue CPs are fixed, while the red ones are allowed to

move by ±10% of their reference position in the chordwise and normal-to-the-chord directions, resulting in 26 design variables in total. A dataset of $N_{db,I} = 100$ sample airfoils in the form of black-and-white images (Fig. [4.2]) is generated. A structured grid of $30K$ cells is generated around each airfoil, and the flow of Table [4.1] is solved using the PUMA CFD software. The working dataset, originally developed in [26] for different objectives, is provided in a ready-to-use format.

| Quantity | Symbol | Value |
|---|---|---|
| Freestream Mach Number | $M_\infty$ | 0.13 |
| Reynolds Number ($\cdot 10^6$) | $Re$ | 3.8 |
| Angle of Attack (°) | AoA | 2.2 |

**Table 4.1:** *Flow conditions of Application I.*

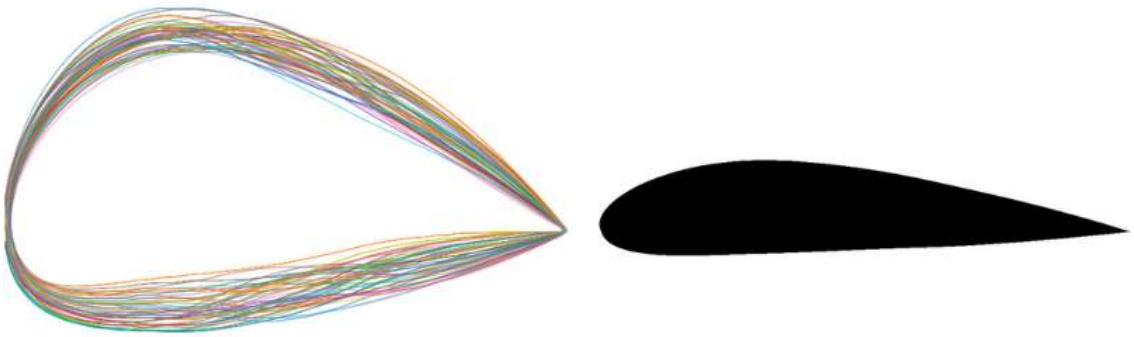


**Figure 4.2:** *(left) A (scaled) depiction of the outlines of the NACA4318 airfoil's variations that comprise this Application's dataset. (right) A sample airfoil displayed in the black-and-white form that it will be inserted into the model.*

The lift and drag coefficients are given by the following formulas:

$$C_L = \frac{L}{0.5\rho U_\infty^2 A} \quad , \quad C_D = \frac{D}{0.5\rho U_\infty^2 A}$$

where $L$ and $D$ are the lift and drag forces respectively, $\rho$ is the air density and $U_\infty$ is the freestream velocity.

## 4.2 Proposed Baseline Architecture

Capture of the non-linear phenomena governing the aerodynamic problem requires *1)* successfully capturing features in the input images (Image Recognition - CNN) and *2)* correctly interpreting these features (DNN). Thus, the developed models follow the simple architecture of Fig. [4.3]; the first half comprises subsequent pairs of Convolutional and Max Pooling layers, while the second half is a sequence of dense layers leading to a scalar output. The black-and-white depictions of the generated samples (Fig. [4.2]) are fed into the network in the form of $[640 \times 320]$ tensors. Out of the 100 samples, 20% are isolated and used to evaluate the model post training. Out of the 80 remaining samples, 20% is used to form a validation set for the model to be evaluated in each epoch during the training process.

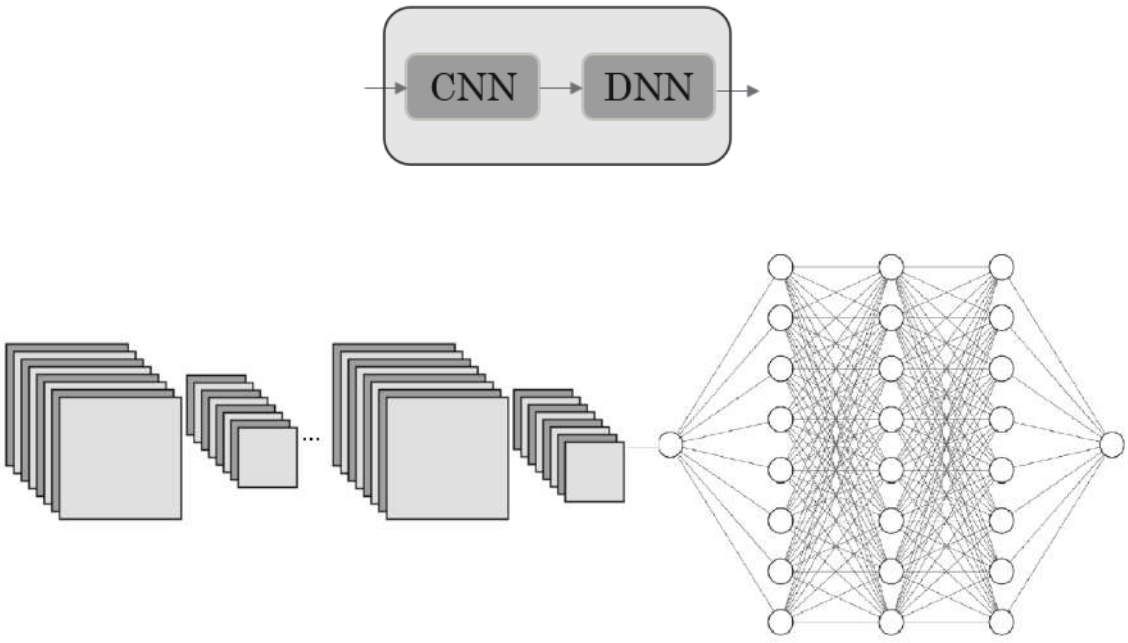



**Figure 4.3:** *The single-branch CNN model architecture implemented in the present application, illustrated both simplified (top) and more detailed (bottom), featuring subsequent pairs of 2D convolutional layers and Max Pooling layers, followed by a series of fully connected layers. The output layer consists of one neuron, corresponding to the single scalar value of interest.*

## 4.3 Case I - Airfoil's lift coefficient

In Case I, the goal is the accurate prediction of the airfoils' $C_L$. The elite conventional configurations proposed by the evolutionary algorithms share a small number of layers, limited to a maximum of 5 2D Convolutional/MaxPooling pairs and 3 dense hidden layers. The proposed models mainly used the ReLU and GELU activation functions for the 2D Convolutional and dense layers respectively, however models using GELU for both types of layers achieved more accurate predictions.

Fig. [4.4] presents the predictions of the three elite conventional model configurations proposed by EASY and summarized in Table [4.2]. The models only differ in terms of the activation function they use. In the first model, the GELU activation function is selected and used in both the Convolutional and the dense layers (G-G configuration). The second model features the ReLU and GELU activation functions in the 2D Convolutional and dense hidden layers respectively (R-G configuration) and the last model uses the ReLU function in all hidden layers (R-R configuration) (Table 4.2).

| **Parameter** | **Model 1 (G-G)** | **Model 2 (R-G)** | **Model 3 (R-R)** |
|---|---|---|---|
| Number of CNN Layers | 5 | 5 | 5 |
| Number of DNN Layers | 3 | 3 | 3 |
| CNN L1 filter size (pow. of 2) | 3 | 3 | 3 |
| CNN L2 filter size (pow. of 2) | 5 | 5 | 5 |
| CNN L3 filter size (pow. of 2) | 5 | 5 | 5 |
| CNN L4 filter size (pow. of 2) | 7 | 7 | 7 |
| CNN L5 filter size (pow. of 2) | 8 | 8 | 8 |
| DNN L1 neuron size (pow. of 2) | 7 | 7 | 7 |
| DNN L2 neuron size (pow. of 2) | 6 | 6 | 6 |
| DNN L3 neuron size (pow. of 2) | 5 | 5 | 5 |
| act. function CNN Layers | GELU | ReLU | ReLU |
| act. function DNN Layers | GELU | GELU | ReLU |
| kernel size (constant) | (3,3) | | |
| strides (constant) | (1,1) | | |
| pool size (constant) | (2,2) | | |
| batch size | 8 | | |
| epochs | 400 | | |

**Table 4.2:** *Summary of the proposed conventional models after the exploration of the design space by EASY.*

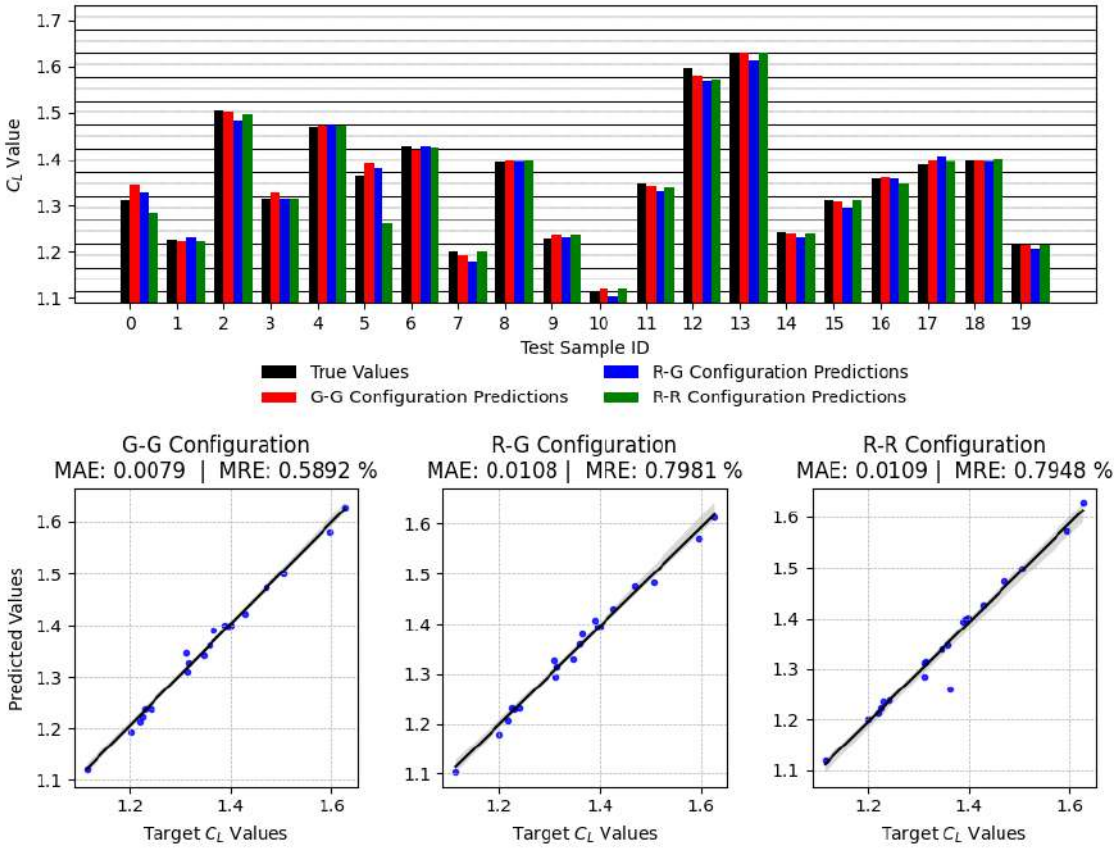



**Figure 4.4:** *Performance of the proposed G-G, R-G and R-R models. (top) Bar plot comparing the target $C_L$ values to the predictions of the three proposed models. (bottom) Regression plots for each model, illustrating the agreement of its predictions with the target $C_L$ values.*

All models perform well, however application of the ReLU activation function seems to downgrade performance, mainly concerning samples closer to the dataset's boundaries. Application of GELU in both the 2D Convolutional and dense layers results in better agreement of the model's prediction with the target values.

To improve accuracy and achieve a more consistent performance, SE Blocks are integrated between each Convolutional and MaxPooling layer. In this Application, all SE-Blocks share the same configuration; a reduction ratio of 16 is defined and the excitation process is performed by two fully connected layers with $floor(C/ratio)$ and $C$ neurons respectively, using the ReLU activation function.

Respecting the updated architecture, fine-tuning proposes model configurations of higher complexity, featuring a larger number of dense hidden layers and neurons. The GELU activation function is used both for the Convolutional and the fully connected layers, with only a single model applying the ReLU function in the Convolutional layers. No other activation functions are applied in the elite networks.

Two proposed elite configurations, extensions of the G-G and R-G models with SE Blocks, are selected (denoted G-G-SE and R-G-SE), presenting an increased accuracy and overall performance when compared to the models lacking the SE-blocks, as depicted in Fig. [4.5].

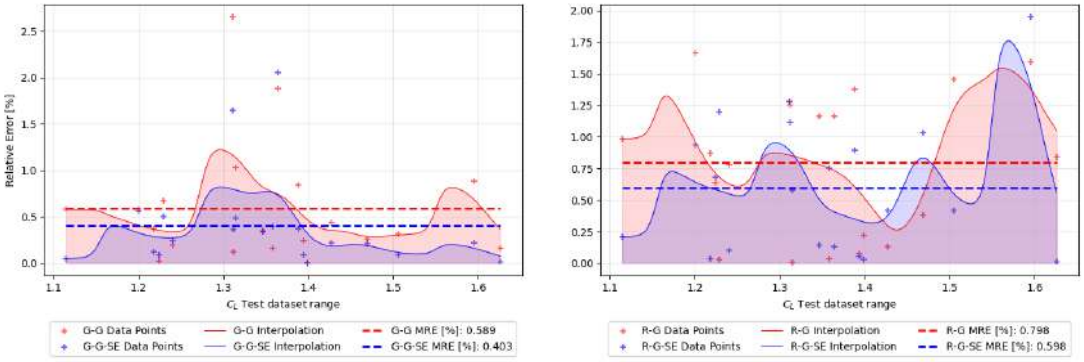



**Figure 4.5:** *Demonstration of the effect of SE-Blocks on the models' performance, illustrating the (distribution of) ARE of the G-G models (left) and R-G models (right) evaluated on the Test Dataset.*

This error plot format is used multiple times throughout this Application. It presents a visualization of the error distribution of models' evaluations. The scattered points correspond to the test samples comprising the evaluation dataset. Their $x-$ value corresponds the true value of the sample, while the $y-$axis refers to the ARE of the corresponding predictions. The test data range is divided into segments, and the local mean error is calculated for all points within each segment. The curved line represents an interpolation of these mean error values, providing an overview of the error distribution across the entire dataset. The dashed line illustrates the $MARE$ for each Model.

Configuration G-G-SE, featuring the characteristics of Table 4.3, performs accurately across the entire range of the dataset. Integration of SE-Blocks resulted in an almost 31% decrease of the mean error, achieving a $MARE$ of 0.403%, which is more evenly distributed between the test samples. Additionally, convergence of the error was achieved in fewer epochs. The induced relative computational drawback is approximated at 0.79%.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Number of CNN Layers | 5 | Number of DNN Layers | 8 |
| CNN L1 filter size (pow. of 2) | 3 | DNN L1 neurons (pow. of 2) | 6 |
| CNN L2 filter size (pow. of 2) | 2 | DNN L2 neurons (pow. of 2) | 6 |
| CNN L3 filter size (pow. of 2) | 6 | DNN L3 neurons (pow. of 2) | 5 |
| CNN L4 filter size (pow. of 2) | 5 | DNN L4 neurons (pow. of 2) | 9 |
| CNN L5 filter size (pow. of 2) | 3 | DNN L5 neurons (pow. of 2) | 9 |
| CNN L6 filter size (pow. of 2) | – | DNN L6 neurons (pow. of 2) | 9 |
| CNN L7 filter size (pow. of 2) | – | DNN L7 neurons (pow. of 2) | 6 |
| CNN L8 filter size (pow. of 2) | – | DNN L8 neurons (pow. of 2) | 9 |
| act. function CNN layers | GELU | act. function DNN layers | GELU |
| kernel size (constant) | (3, 3) | batch size | 8 |
| strides (constant) | (1, 1) | epochs | 400 |
| pool size (constant) | (2, 2) | | |

**Table 4.3:** *Summary of the proposed G-G-SE model after the exploration of the design space by EASY.*

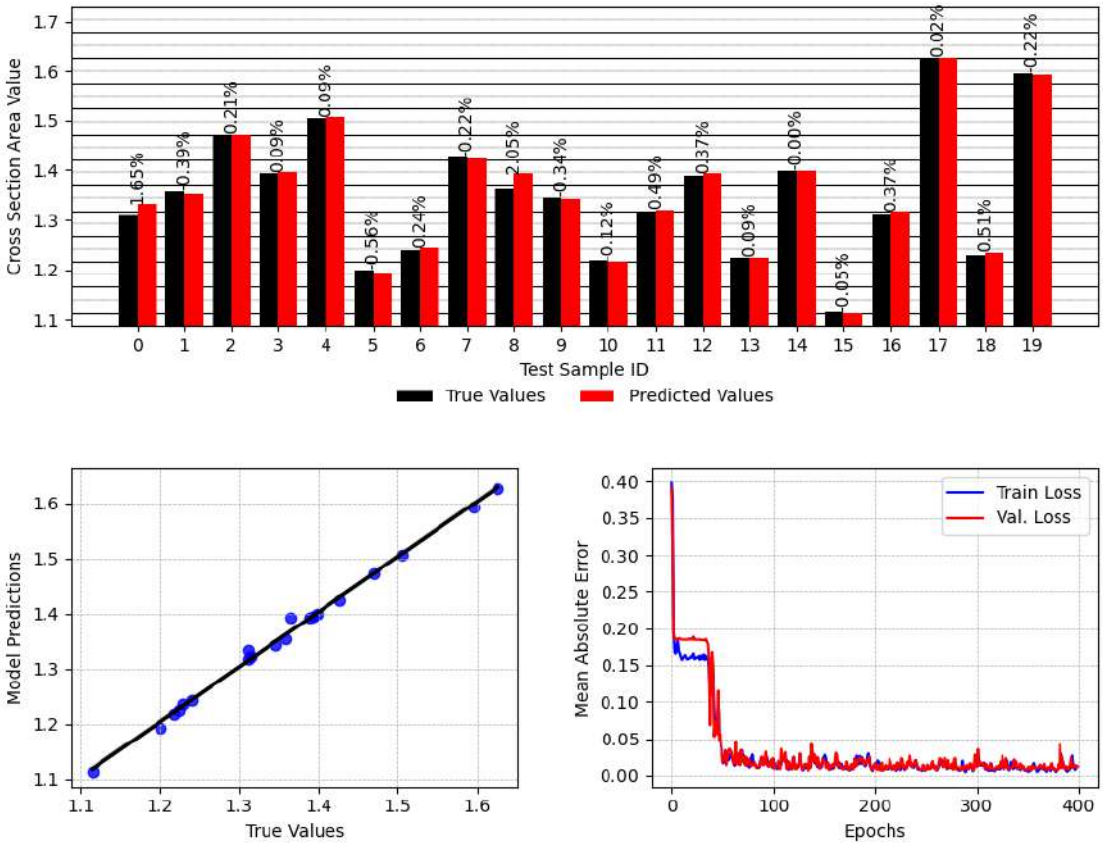



**Figure 4.6:** *Performance of the selected G-G-SE model. (top) Bar plot comparing the target $C_L$ values to the predictions of the model, displaying the Relative Error on each sample's prediction (MARE: 0.403%). (left) Regression plot of the selected model, illustrating the agreement of its predictions with the target $C_L$ values. (right) Convergence of the MAE of the model over the training epochs.*

## 4.4 Case II - Airfoil's drag coefficient

Case II concerns the prediction of airfoils' $C_D$. Initially, the baseline network architecture of Case I is used. The models proposed by EASY apply mostly the ReLU activation function in both convolutional and dense layers, and achieve a minimum *MARE* of 0.5463% across the test dataset. Similarly to Case I, SE-Blocks are integrated, aiming to direct attention to the more informative feature maps that capture nonlinearities governing the case. Surprisingly, this downgrades performance, increasing *MARE* to 0.6143% (Fig. [4.7]).

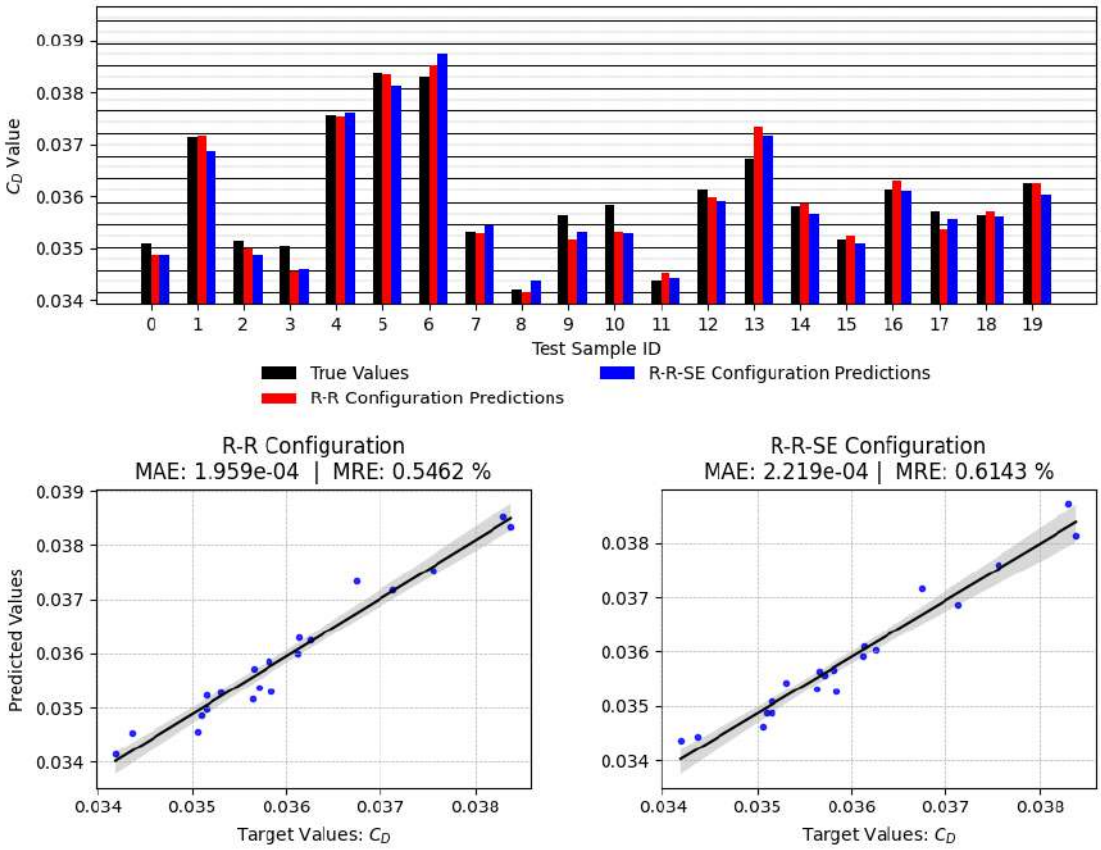



**Figure 4.7:** *Performance of the proposed R-R and R-R-SE models. (top) Bar plot comparing the target $C_D$ values to the predictions of the two proposed models. (bottom) Regression plots for each model, illustrating the agreement of its predictions with the target $C_D$ values.*

The results indicate that moderately-tuned plain SE-models predict the $C_D$ inadequately, suggesting that either SE-Blocks alone do not provide sufficient representational power, or that extensive attention needs to be given to the interpretational part of the network. To enhance performance and improve generalization, $L2$ Regularization is introduced in the fully connected layers, with a regularization strength

$\lambda = 0.001$. Additional exploration by EASY proposes the network summarized in Table 4.4. The combination of SE-Blocks and $L2$ regularization results in the slightly improved overall performance of Fig. [4.8], with minimal additional computational cost (approximately 0.66% increase in training time). While the Elite network achieved an improved $MARE$ (0.523% , $\sim$ 4.24% decrease), distribution remains non-uniform across the dataset, indicating potential limitations in generalization, when insufficient resources are expended in fine-tuning.

| **Parameter** | **Value** | **Parameter** | **Value** |
|---|---|---|---|
| Number of CNN Layers | 5 | Number of DNN Layers | 4 |
| CNN L1 filter size (pow. of 2) | 7 | DNN L1 neurons (pow. of 2) | 7 |
| CNN L2 filter size (pow. of 2) | 9 | DNN L2 neurons (pow. of 2) | 5 |
| CNN L3 filter size (pow. of 2) | 6 | DNN L3 neurons (pow. of 2) | 6 |
| CNN L4 filter size (pow. of 2) | 7 | DNN L4 neurons (pow. of 2) | 7 |
| CNN L5 filter size (pow. of 2) | 2 | DNN L5 neurons (pow. of 2) | – |
| CNN L6 filter size (pow. of 2) | – | DNN L6 neurons (pow. of 2) | – |
| CNN L7 filter size (pow. of 2) | – | DNN L7 neurons (pow. of 2) | – |
| CNN L8 filter size (pow. of 2) | – | DNN L8 neurons (pow. of 2) | – |
| act. function CNN layers | ReLU | act. function DNN layers | GELU |
| kernel size (constant) | (3, 3) | batch size | 8 |
| strides (constant) | (1, 1) | epochs | 700 |
| pool size (constant) | (2, 2) | | |

**Table 4.4:** *Summary of the proposed R-G-SE model after the exploration of the design space by EASY.*

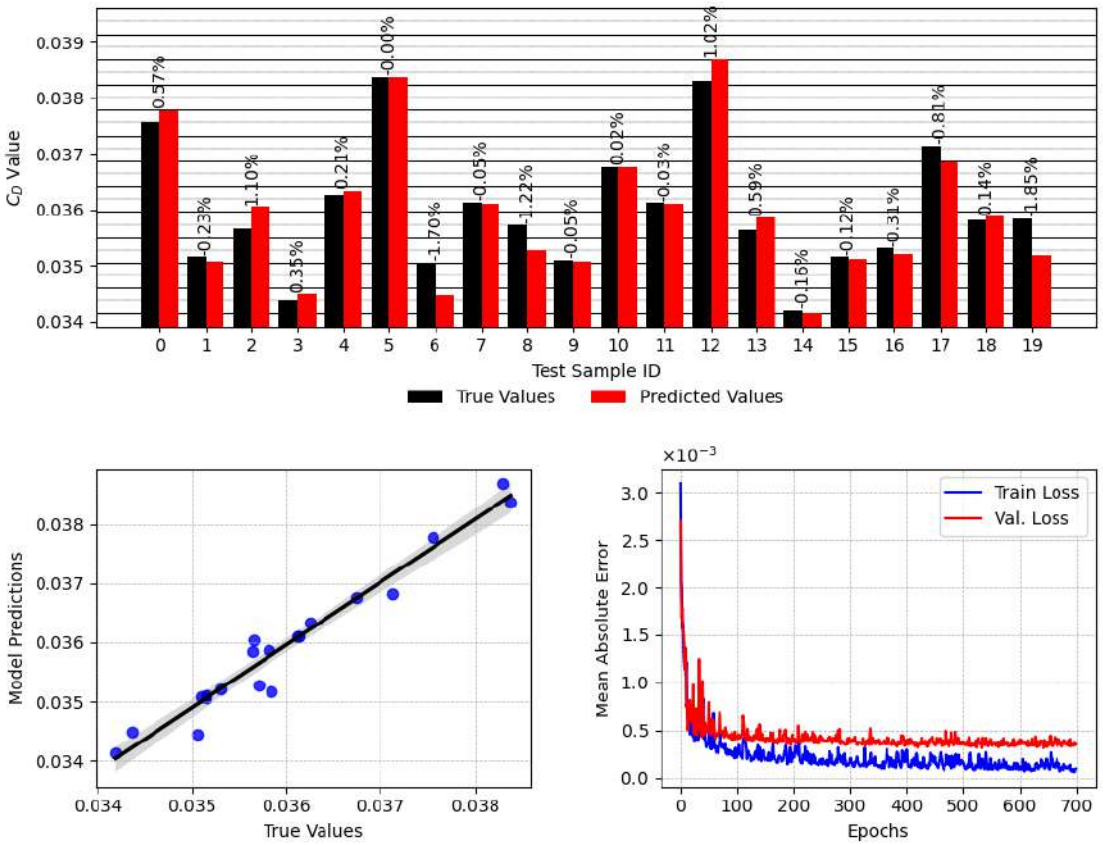



**Figure 4.8:** *Performance of the selected R-G-SE model. (top) Bar plot comparing the target $C_D$ values to the predictions of the model, displaying the Relative Error on each sample's prediction (MARE: 0.523). (left) Regression plot of the selected model, illustrating the agreeent of its predictions with the target $C_D$ values. (right) Convergence of the MAE of the model over the training epochs.*

## 4.5 Case III - Airfoil's cross section area

Similarly to the $C_D$ case, the connection between geometric modifications and cross section area is non-linear, however predicting it supposedly easier, since a pixel-wise approach should theoretically be sufficient. Initially, two models are evaluated, employing SE-Blocks along with $L1$ and $L2$ Regularization respectively (denoted L1-SE, L2-SE). Both follow the baseline architecture of the previous cases. Fine-tuning strictly proposes models applying the ReLU activation function in their Convolutional layers and the GELU activation function in the dense layers.

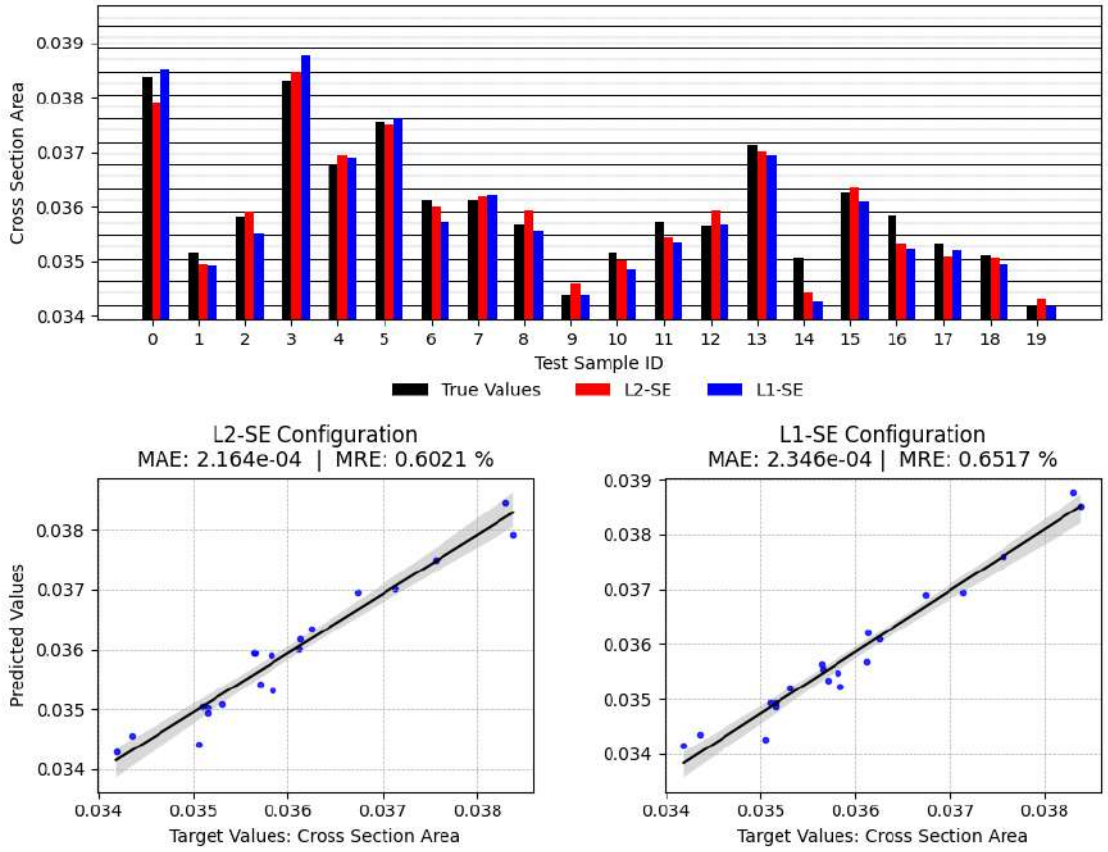



**Figure 4.9:** *Performance of the proposed L1-SE and L2-SE models. (top) Bar plot comparing the target area values to the predictions of the two models. (bottom) Regression plots for each model, illustrating the agreement of its predictions with the target area values.*

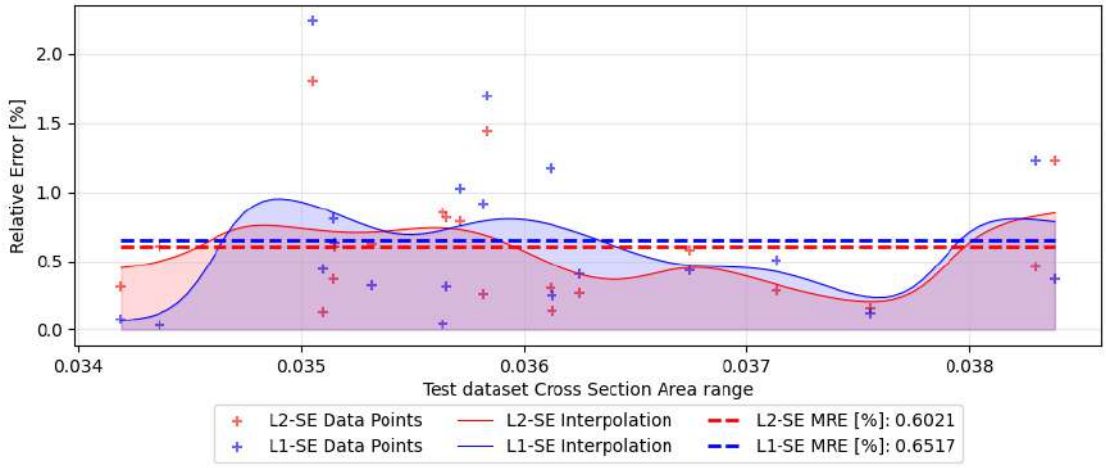



**Figure 4.10:** *Illustration of the (distribution of) ARE of the L1-SE and L2-SE models evaluated on the same Test dataset.*

Fig. [4.9] presents the two Elites' evaluations, showcasing a slightly better performance by the L2-SE model. Additionally, Fig. [4.10] depicts the models' Absolute Relative Error distribution, when evaluated on the same test dataset. The L2-SE configuration performs slightly better.

The L2-SE model is further tuned with EASY. The training epochs are increased to 700 and the regularization strength to 0.0013. The proposed model, denoted L2-SE*, is summarized in Table 4.5. It presents a better overall performance and a *MARE* of 0.3353% ($\sim$ 44.3% decrease) , distributed more evenly throughout the dataset, as shown in Fig. [4.11]. The (SE-Block associated) computational drawback is approximately 0.71%.

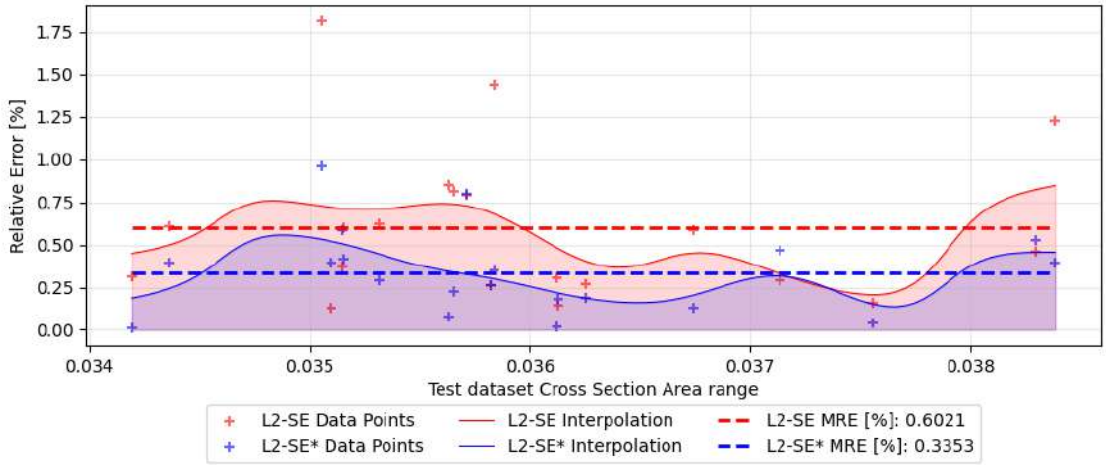



**Figure 4.11:** *Illustration of the (distribution of) ARE of the L2-SE and L2-SE* models evaluated on the same Test dataset.*

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Number of CNN Layers | 5 | Number of DNN Layers | 3 |
| CNN L1 filter size (pow. of 2) | 5 | DNN L1 neurons (pow. of 2) | 8 |
| CNN L2 filter size (pow. of 2) | 9 | DNN L2 neurons (pow. of 2) | 9 |
| CNN L3 filter size (pow. of 2) | 7 | DNN L3 neurons (pow. of 2) | 6 |
| CNN L4 filter size (pow. of 2) | 2 | DNN L4 neurons (pow. of 2) | – |
| CNN L5 filter size (pow. of 2) | 2 | DNN L5 neurons (pow. of 2) | – |
| CNN L6 filter size (pow. of 2) | – | DNN L6 neurons (pow. of 2) | – |
| CNN L7 filter size (pow. of 2) | – | DNN L7 neurons (pow. of 2) | – |
| CNN L8 filter size (pow. of 2) | – | DNN L8 neurons (pow. of 2) | – |
| act. function CNN layers | ReLU | act. function DNN layers | GELU |
| kernel size (constant) | (3, 3) | batch size | 10 |
| strides (constant) | (1, 1) | epochs | 700 |
| pool size (constant) | (2, 2) | | |

**Table 4.5:** *Summary of the selected L2-SE* model after the exploration of the design space by EASY.*

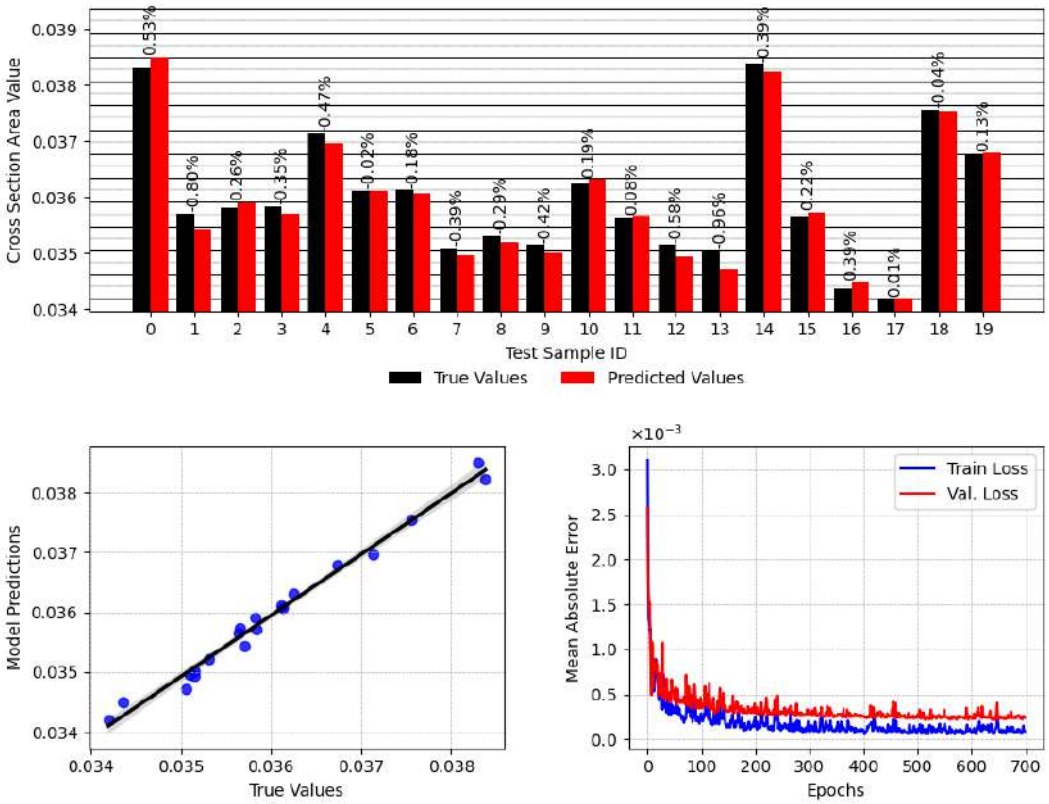



**Figure 4.12:** *Performance of the selected L2-SE* model. (top) Bar plot comparing the target area values to the predictions of the model, displaying the Relative Error on each sample's prediction (MARE: 0.3353%). (left) Regression plot of the selected model, illustrating the agreement of its predictions with the target area values. (right) Convergence of the MAE of the model over the training epochs.*

## 4.6 Overview and Conclusions

Application I demonstrates the capability of neural networks to accurately predict airfoil aerodynamic characteristics directly from image representations. The developed models estimate the lift coefficient $C_L$, drag coefficient $C_D$, and airfoil cross-sectional area. While prediction errors remain low across all cases, some variations occur due to differences in data variance. Since model predictions are constrained by the dataset's target value ranges, direct comparison of relative errors of the different quantities is misleading due to their greatly different scales. For instance, $C_D$ values range from 0.0329 to 0.0396 (a span of $\sim$0.0066), whereas $C_L$ values span from 0.9438 to 1.7490 (a range of $\sim$0.8051, over one hundred times larger) Moreover the three output quantities follow different distributions ($C_L$: $\sigma = 0.1812$, Shapiro-Wilk normality test [42] p-value = 0.2171 , $C_D$ and $Areas$: $\sigma \sim 0.0015$, Shapiro-Wilk p-value $\sim$ 0.082). Consequently, the relative errors for each quantity operate on fundamentally different scales.

To achieve a fair comparison of model accuracy across all cases within their training range, the relative error distribution is computed and analyzed using the standardized data (scaled to a 0-100 range), ensuring a shared scale. Fig. [4.13] presents this error distribution of the three selected model configurations.

The G-G-SE configuration of the $C_L$ case achieves the highest overall accuracy, with a $MARE_{Std,C_L} = 1.4271\%$. Drag coefficient prediction is less accurate, potentially due to its high sensitivity to geometric perturbations. Research has shown that $C_D$ can exhibit first-order sensitivities exceeding 50% with respect to small design variations, indicating that even minor geometric changes can significantly impact drag [49]. The elite network achieved $MARE_{Std,C_D} = 6.8854\%$.

Notably, the optimal model configuration for area prediction achieved $MARE_{Std,area} = 4.5501\%$, which appears suboptimal given that it represents a relatively straightforward computer vision problem, particularly when working with binary input images. This suggests that, while increasing model complexity and employing advanced components can enhance representational and interpretational capabilities, it may over-parameterize geometric relationships, making it more difficult to identify simpler input-output connections.

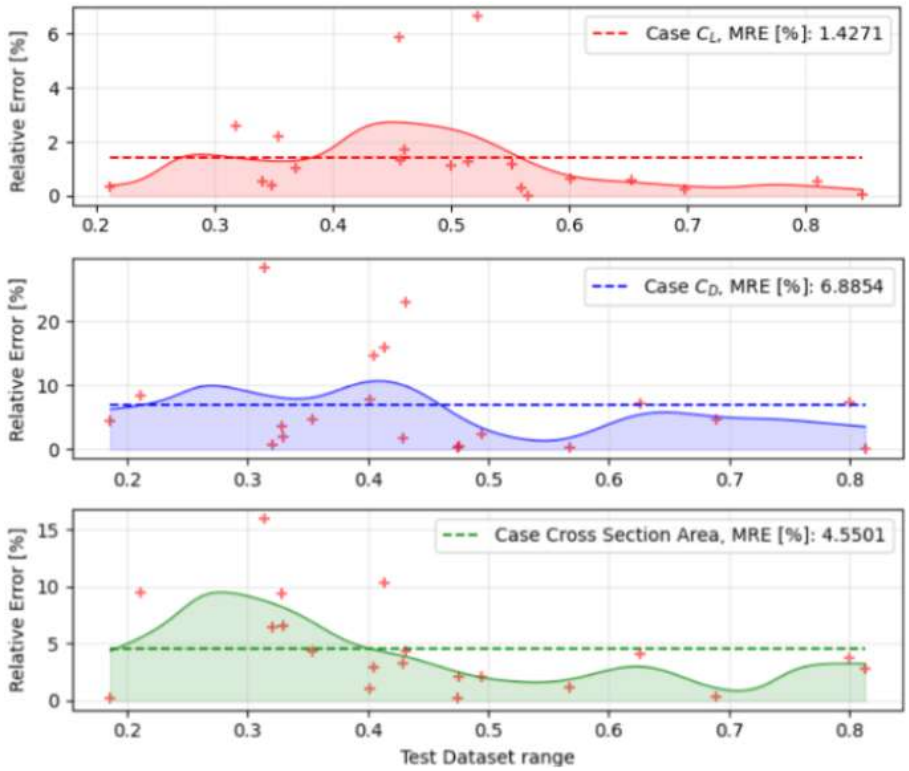



**Figure 4.13:** *Illustration of the (distribution of) ARE of the selected models for the three examined subcases, evaluated on the same test dataset. The Relative Error formula is applied on the transformed data.*

The conducted studies provided a robust baseline network architecture with confirmed superiority over the equivalent conventional CNN and negligible increases in training cost. The results highlight the potential of SE-blocks, both as standalone components and in conjunction with regularization techniques. This general architecture serves as the foundational structure for models developed in the subsequent Applications.

Furthermore, the results of Application I supply valuable insights about the design space explored by EASY. Ineffective regions of the design space can be systematically excluded, thus narrowing down the exploration domain. This allows for reduced computational resource requirements for model fine-tuning in subsequent Applications II and III, by avoiding unnecessary parameter searches.

Specifically, EASY identified ReLU, GELU, sigmoid, and tanh as the most promising candidates as far as activation functions are concerned. The analysis also revealed that dense layer dimensions need not exceed those of convolutional layers, providing architectural constraints that aid both efficiency and performance.

# Chapter 5
# Application II - Automobile's Drag Force and Surface Area (1 Morphing Box)

## 5.1 Introduction

Application II concerns the extension of the previously developed backbone architecture in problems concerning three-dimensional geometries of cars, aiming to accurately predict their surface area and aerodynamic drag. Although the drag is the primary quantity of interest, surface area is also examined due to its non-linear nature. In this way, the model is evaluated on more non-linear data, allowing for the test of its generalization ability, while not spending excessive computational resources on CFD simulations. Different approaches are examined to develop a CNN configuration that balances predictive accuracy and computational efficiency in the context of the task at hand, aiming to identify the most promising one for use in the subsequent Application III.

### 5.1.1 The DrivAer car model

The DrivAer is a publicly available aerodynamic benchmark car model developed by researchers at the Technical University of Munich (TUM) in collaboration with Audi. It was introduced in 2012 as a more realistic alternative to traditional simplified car models (like the Ahmed body) for CFD and wind tunnel testing in automotive aerodynamics research [20]. It represents a modern passenger car with a detailed underbody, wheels, mirrors, and a slanted rear window. Herein, the fastback configuration is used, featuring a smooth, sloping rear, as depicted in Fig. [5.1].

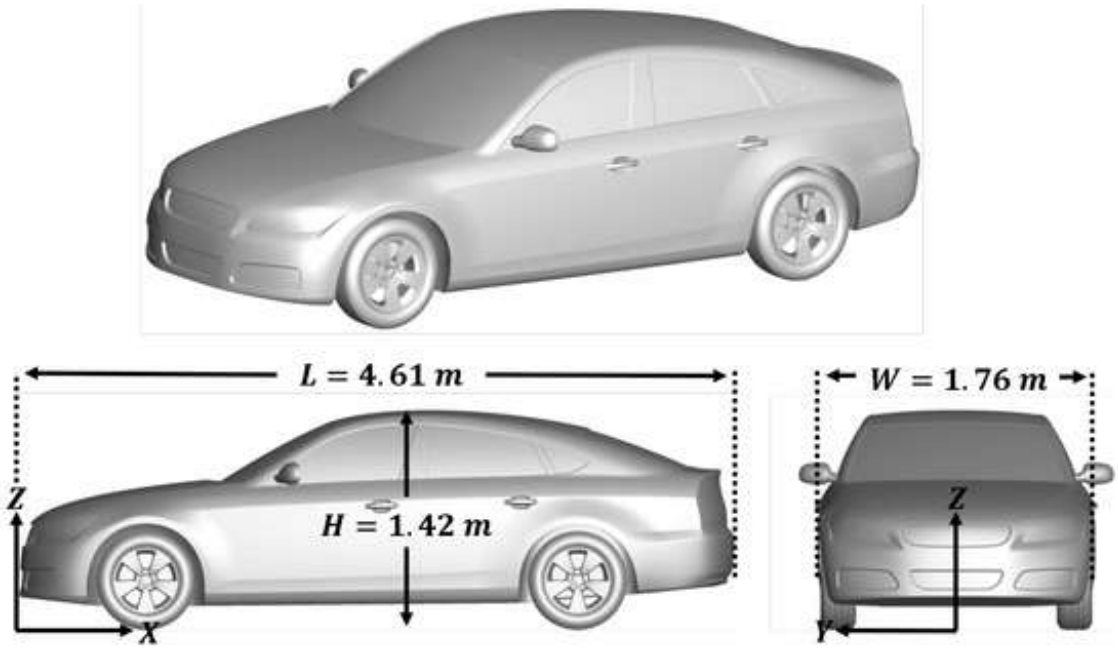



**Figure 5.1:** *The fastback configuration of the DrivAer open-source car model, featuring detailed characteristics and a smooth, sloping rear. Image taken from [3]*

### 5.1.2 Dataset Generation

This problem's dataset, also obtained from [26], results from free-form deforming the rear end of the baseline automobile. Specifically, 210 CPs are arranged in a $7 \times 5 \times 6$ NURBS control lattice and positioned at the rear part of the (half) car, as depicted in Fig. [5.3] (extended to the full car). The CPs in red are allowed to move by $\pm 25\%$ in the longitudinal direction and by $\pm 60\%$ in the normal-to-the-ground direction, resulting in a total of 96 design variables. A dataset of $N_{db,II} = 100$ samples is generated.

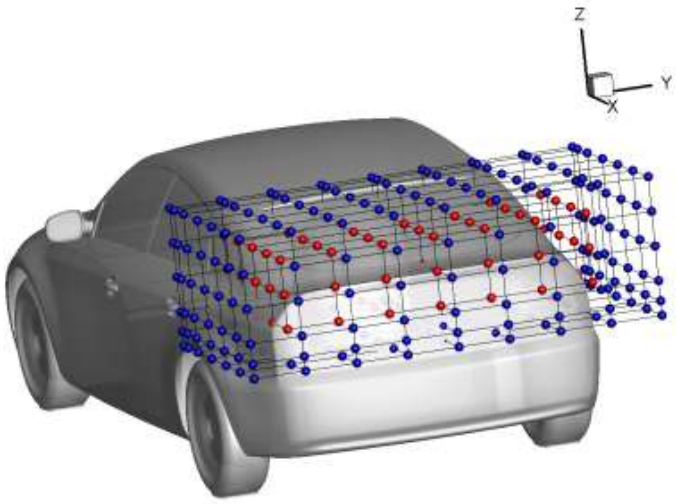



**Figure 5.2:** *Illustration of the FFD control lattice partially enclosing the DrivAer model's rear. Image taken from [26].*

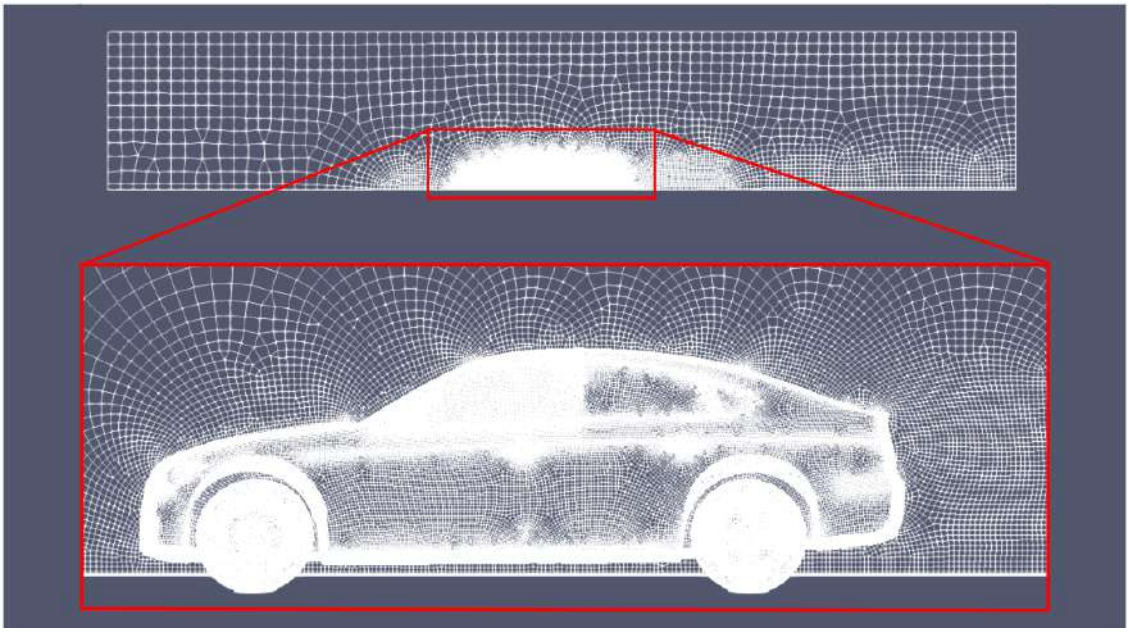


**Figure 5.3:** *The generated mesh around the geometry of the baseline DrivAer model.*

A computational grid of $\sim 1.4M$ nodes is generated around the geometry, and the flow summarized in Table 5.1 is solved, using the PUMA CFD Software. Specifically, the incompressible variant of the PUMA code is employed, that uses the pseudo-compressibility method. The simulation concerns half the car's geometry and uses symmetry conditions (Fig. [5.4])

| Quantity | Symbol | Value |
|---|---|---|
| Freestream Flow Velocity ($\mathrm{m\,s^{-1}}$) | $U_\infty$ | 38.85 |
| Air Kinematic Viscosity ($\cdot 10^{-5}\mathrm{m^2/s}$) | $\nu$ | 1.5 |
| Angle of Attack (°) | AoA | 0.0 |

**Table 5.1:** *Flow conditions of Application II.*

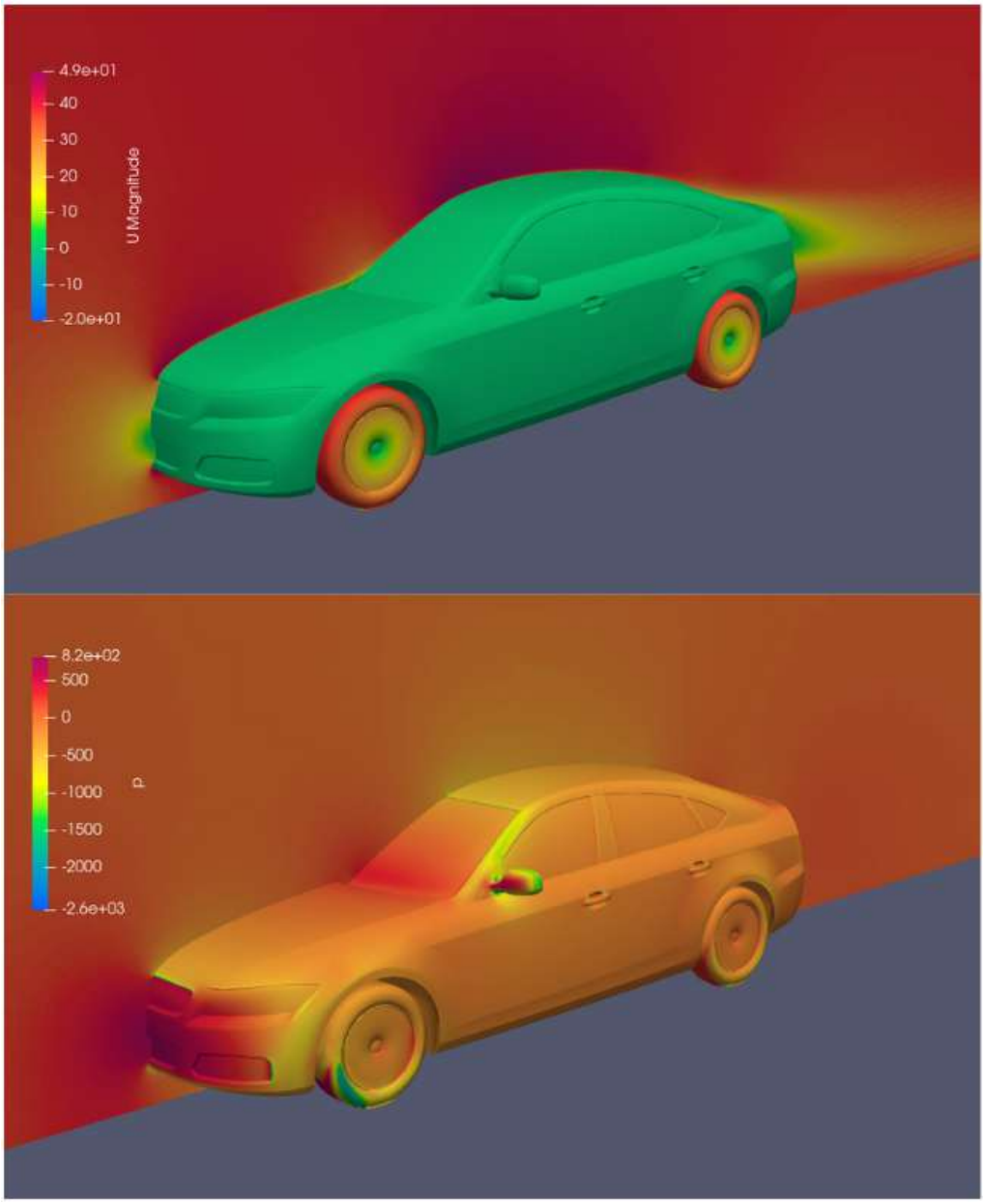



**Figure 5.4:** *(top) Axial velocity and (bottom) pressure fields of the flow around a sample from the generated dataset. Negative pressures are present because the ambient pressure is set to zero, since pressure differences are of interest, not absolute values.*

In the automotive industry, stylists typically illustrate conceptual designs by provid-

ing their rear, side, rear, top and/or rear three-quarter (R34) views to the design engineering team. In order to generate a synthetic dataset mimicking a realistic industrial one, the sample geometries are initially captured from the different viewpoints. These images are then resized to relatively small dimensions, to reduce computational load and complexity. Finally, they undergo a series of transformations aimed at partially reducing the visual information content and giving the images the desired sketch-like appearance. This process ensures that the final dataset reflects the characteristics of real-world design sketches. A sample of the input dataset is given in Fig. [5.5].

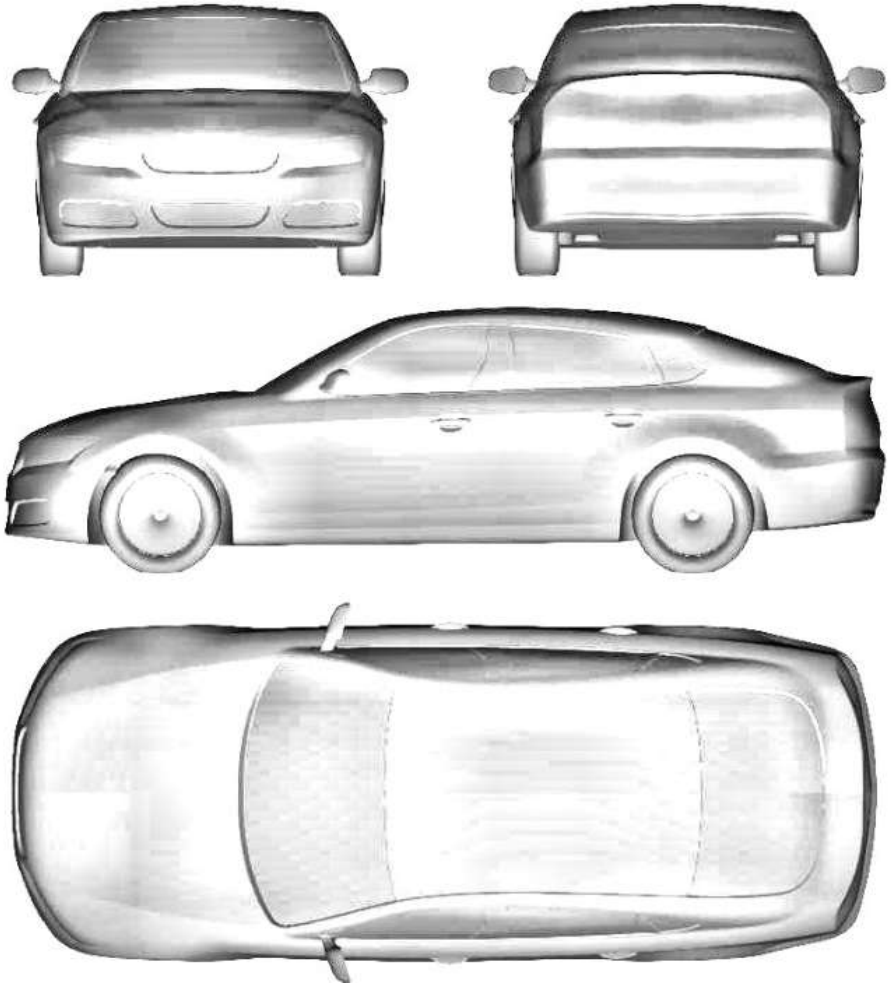


**Figure 5.5:** *Result of preprocessing on images of the front, rear, side and upper views of a sample from the generated dataset. The images are to be fed into the network in this form.*

## 5.2 Statistically Informed Dataset Transformation

The small size of the working dataset ($N_{db,II} = 100$) already poses challenges for training. The output values that the model is tasked with predicting are examined w.r.t. their statistical properties. The results reveal -most notably- an extremely low variance of $\sigma^2 = 1.82$ and a relatively high mean Drag Force value of $\mu = 226.07$ [N]. Such a narrow dynamic range can obstruct model learning by reducing the gradient magnitude, thus weakening the optimization signal during training [13]. The distributional properties of the output dataset are displayed in Fig. [5.6].

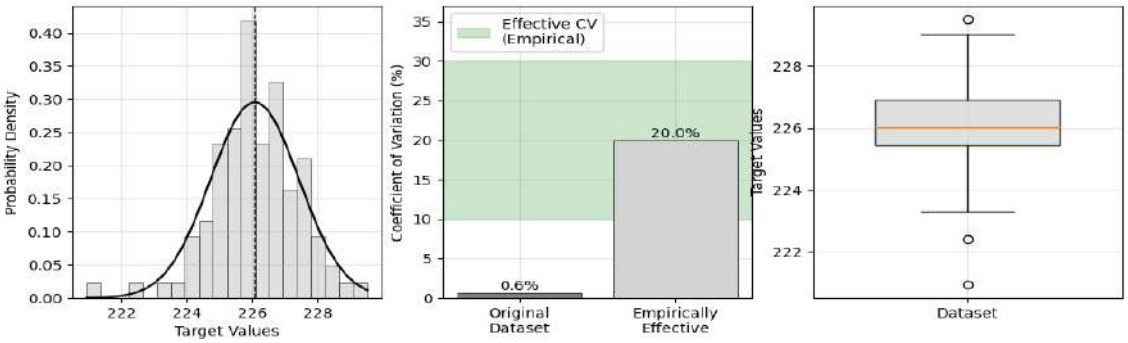



**Figure 5.6:** *Distributional summary of the original output dataset. (left) The dataset's distribution compared to the corresponding normal curve. (center) Comparison of the dataset's coefficient of variation CV against an empirically effective CV range. (right) The dataset's boxplot.*

This plot format, used multiple times throughout the Thesis, summarizes the distributional properties of a dataset. The left plot offers a comparison of the dataset's distribution to a gaussian normal distribution. The central plot compares the dataset's coefficient of variation (CV), defined CV $= \frac{\sigma}{\mu} \times 100$, against a desired range $(10 - 30\%)$, according to studies and observations throughout this work. The third plot presents the dataset's box-and-whisker plot or boxplot. The box's lower boundary is denoted $Q_1$ or 25th percentile and the upper boundary is denoted $Q_3$ or 75th percentile. The range of the dataset below $Q_1$ contains 25% of the samples, as does the one above $Q_3$. The box, with a height of $IQR = Q_3 - Q_1$, contains 50% of the dataset's samples. The red line represents the medial value (denoted $Q_2$), which divides the dataset into two equal halves; its positioning illustrates data symmetry. Lastly, the circles beyond the whiskers correspond to outliers in the dataset, whose values are either smaller than $Q_1 - 1.5 \times IQR$ or greater than $Q_3 + 1.5 \times IQR$. From now on, the range contained by the box's boundaries in each case is referred to as $Q_1Q_3$ or interquartile range.

To address the limitations mentioned earlier, an affine transformation is applied to scale the output values into the interval $[0, 100]$.

$$x_{\text{trans}} = \frac{x - \min(x)}{\max(x) - \min(x)} \times 100 \tag{5.1}$$

where $x$ is the original output, and $x_{\text{norm}}$ is the transformed output. This transformation stretches the data across a wider range while preserving the overall distributional shape. However, it does alter scale-dependent metrics such as the mean and standard deviation. While the coefficient of variation, defined $CV = \frac{\sigma}{\mu} \times 100$ is generally scale-invariant, in this case, its value changed due to the affine nature of the transformation affecting the mean and standard deviation non-uniformly.

The transformation leads to a substantial increase in standard deviation, as expected, since the data is stretched over a broader interval. This change improves the signal amplitude and helps to address the vanishing/exploding gradient problem during backpropagation [13]. The coefficient of variation increased from 0.60% to 26.31%, now falling within the more effective range. Skewness and kurtosis remained unchanged by the transformation, as expected; the distribution retained a slightly left-skewed and peaked shape. The distributional properties of the transformed output dataset are illustrated in Fig. [5.7].

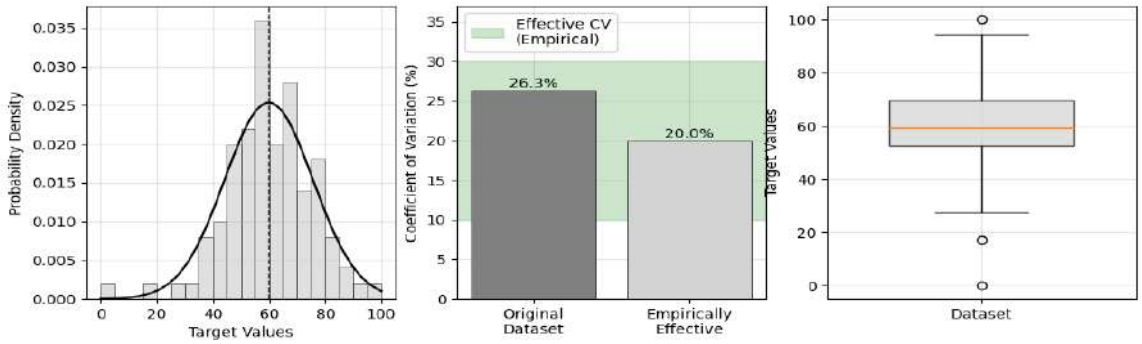



**Figure 5.7:** *Distributional summary of the transformed output dataset. (left) The dataset's distribution compared to the corresponding normal curve. (center) Comparison of the dataset's coefficient of variation CV against an empirically effective CV range. (right) The dataset's boxplot.*

In summary, the transformation successfully expanded the dynamic range and improved the statistical suitability of the output variable for regression. The networks of this Application are trained on the transformed data, and their predictions are rescaled back to the original domain in post-processing. These changes are expected

to improve model training stability and amplify the optimization signal in back-propagation. A summary of the data (pre- and post- transformation) is presented in Table [5.2].

| **Metric** | **Original Data** | **Transformed Data** |
|---|---|---|
| Sample Size ($n$) | 100 | 100 |
| Mean ($\mu$) | 226.07 | 59.87 |
| Standard Deviation ($\sigma$) | 1.35 | 15.75 |
| Variance ($\sigma^2$) | 1.82 | 248.05 |
| Coefficient of Variation (CV) | 0.60% | 26.31% |
| Skewness | -0.4983 | -0.4983 |
| Kurtosis | 1.5305 | 1.5305 |
| Range | 8.57 | 100.00 |

**Table 5.2:** *Statistics of the output dataset before and after transformation.*

## 5.3 Examined Model Configurations

In Application I, a single-branch architecture was implemented. The first part of the model consisted of subsequent Convolutional and MaxPooling layers (as well as advanced Building blocks on some occasions), while the second part consisted of a series of fully connected layers. This structure proved to be adequate for the task at hand, since the airfoil's cross section is two-dimensional, thus only a single input image was needed. However, the present problem is much more complex and concerns three-dimensional geometries.

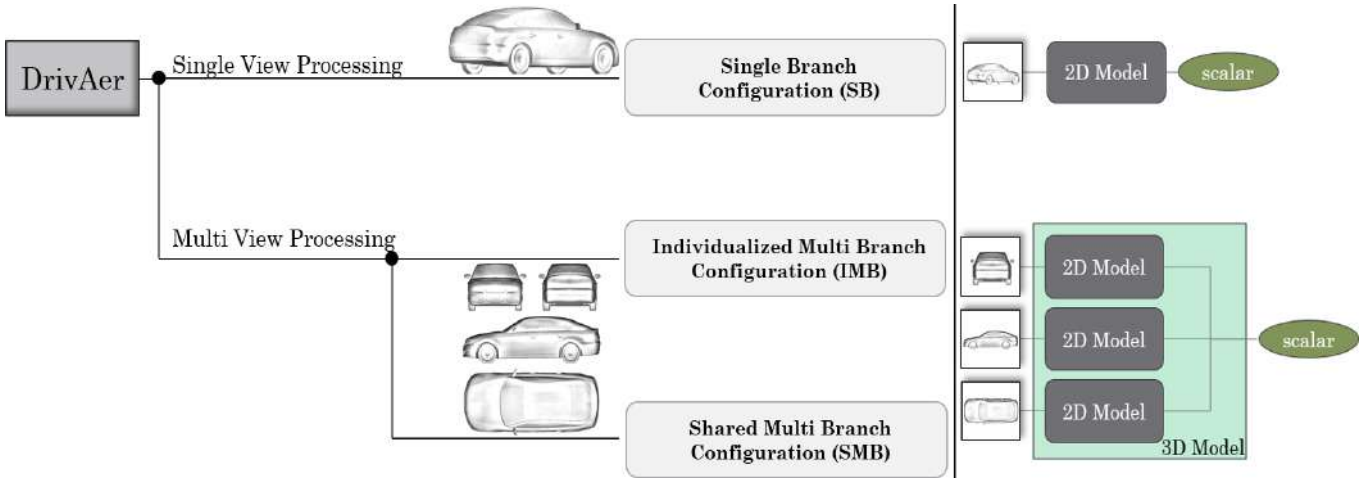



**Figure 5.8:** *Overview of the approaches examined to adaptively extend the configuration of Application I to the current Application's objective, concerning the three-dimensional geometries of automobiles.*

This Thesis examines two approaches, depending on the inputs of the models. The first one suggests a single R34 (isometric-like) image of the car being fed to the model. The core idea relies on the hypothesis that the R34 view contains sufficient information for accurate prediction. Since this approach employs a single input, it requires a single Convolutional branch. Consequently, the network architecture for this implementation remains identical to that of Application I. This model configuration will be referred to as *Single-Branch (SB)*.

The second approach suggests the parallel processing of multiple orthogonal views; each view is processed by a separate Convolutional branch, similar to that of Application I, uniquely associated with it. The outputs of these branches are then concatenated and passed through a series of dense layers for feature interpreta-

tion and processing. While utilizing more images provides a more comprehensive representation of information, it inherently increases both model complexity and training requirements. Additionally, the concatenation step introduces further potential challenges. This baseline architecture will be referred to as *Multi-Branch (MB)*. Two variants of the MB configuration are examined; the *Individualized Multi-Branch (IMB)* configuration and the *Shared Multi-Branch (SMB)* configuration.

In the IMB approach, each Convolutional branch maintains a unique architecture with independently tuned parameters and structural composition during the fine-tuning process. This allows each branch to be specifically adjusted for its corresponding view. In contrast, the SMB approach maintains a uniform architecture across all Convolutional branches. During fine-tuning, this shared architecture is adjusted and implemented by all branches, promoting view-invariant feature learning regardless of the input perspective. An overview of the examined approaches is graphically presented in Fig. [5.8].

During the development of this Application, different architectural setups were tested through a trial-and-error process. Based on observations, an empirical rule was adopted for integrating SE-Blocks and MaxPooling layers within the Convolutional branches. According to this rule, a MaxPooling layer is added after every two Convolutional layers, and an SE-Block is inserted after every three. When both a MaxPooling layer and an SE-Block are to be added after the same Convolutional layer, the SE-Block is placed first. An example of this rule on a branch with seven Convolutional layers is shown in Fig. [6.1]. Its application proved to successfully balance computational efficiency and predictive accuracy.

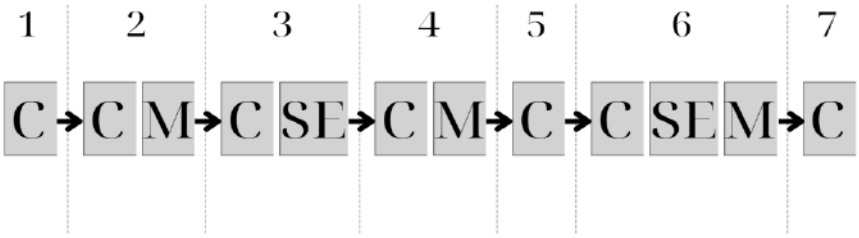



**Figure 5.9:** *Application of the empirical rule on a convolutional branch of 7 convolutional layers. Here, C denotes a 2D Convolutional Layer, M denotes a Max Pooling Layer and SE denotes a Squeeze-and-Excitation Block.*

## 5.4 Single-Branch Model

The dataset of the R34 views is used to train the SB model. The grayscale images are fed to the model in the form of $[157 \times 452]$ tensors. Indicatively, two samples are presented in Fig. [5.10] to illustrate typical differences that the various cars present.

Both for drag and surface prediction models, the number of epochs are set to 200 (idle epochs callback is set to 60) and the batch size is defined 16. The Adam optimizer is used with a learning rate of 0.0013, monitoring the $MAE$ of the predictions on the transformed data. 80 samples are used for training, and 20 are isolated to evaluate the model afterwards.

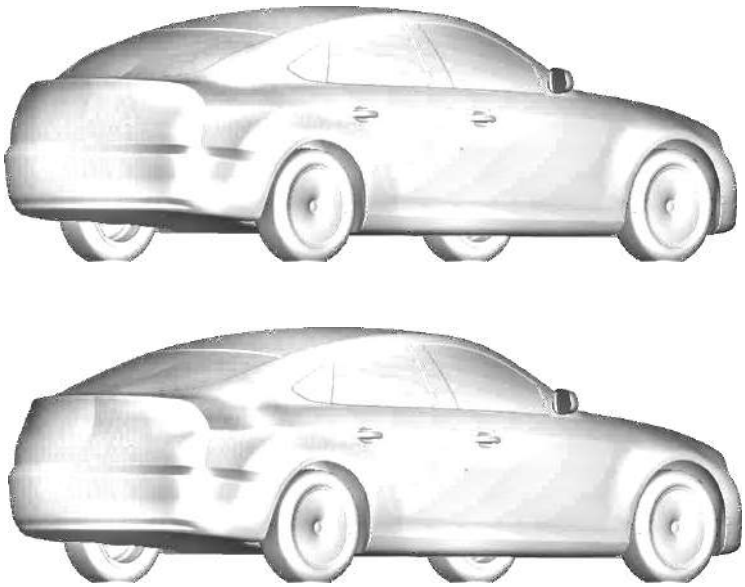


**Figure 5.10:** *Two R34 samples of the SB model's training dataset. The samples are fed into the model in this form.*

### 5.4.1 Drag Force Prediction

In the Drag Force prediction case, EASY proposed the model summarized in Table [5.3].

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Number of CNN Layers | 7 | Number of DNN Layers | 7 |
| CNN L1 filter size (pow. of 2) | 5 | DNN L1 neurons (pow. of 2) | 9 |
| CNN L2 filter size (pow. of 2) | 9 | DNN L2 neurons (pow. of 2) | 7 |
| CNN L3 filter size (pow. of 2) | 6 | DNN L3 neurons (pow. of 2) | 6 |
| CNN L4 filter size (pow. of 2) | 9 | DNN L4 neurons (pow. of 2) | 10 |
| CNN L5 filter size (pow. of 2) | 4 | DNN L5 neurons (pow. of 2) | 6 |
| CNN L6 filter size (pow. of 2) | 4 | DNN L6 neurons (pow. of 2) | 6 |
| CNN L7 filter size (pow. of 2) | 5 | DNN L7 neurons (pow. of 2) | 8 |
| CNN L8 filter size (pow. of 2) | – | DNN L8 neurons (pow. of 2) | – |
| act. function CNN layers | ReLU | act. function DNN layers | GELU |
| kernel size (constant) | (3, 3) | batch size | 16 |
| strides (constant) | (1, 1) | epochs | 200 |
| pool size (constant) | (2, 2) | | |

**Table 5.3:** *Summary of the proposed SB model after the exploration of the design space by EASY.*

Evaluation of the model on the test dataset demonstrates an inability to accurately capture the target values of the Drag Force. Fig. [5.11] displays an inadequate performance, with strong disagreement between the model's predictions and the ground truth. Fig. [5.12 illustrates that, closer to the bounds, the Error takes the highest values, however, even at the center of the dataset, the network is inaccurate. The significant dispersion of predictions in the regression plot, as well as the large slope deviation between the predictions' regression line and the identity line (10.35°) indicate that the model has failed to capture the connection between geometric modifications and their corresponding effects on the Drag Force. These results raise concerns about the Single-Branch Configuration's capabilities in this Application, especially given that the presented model is the elite proposition of a computationally significant design space exploration.

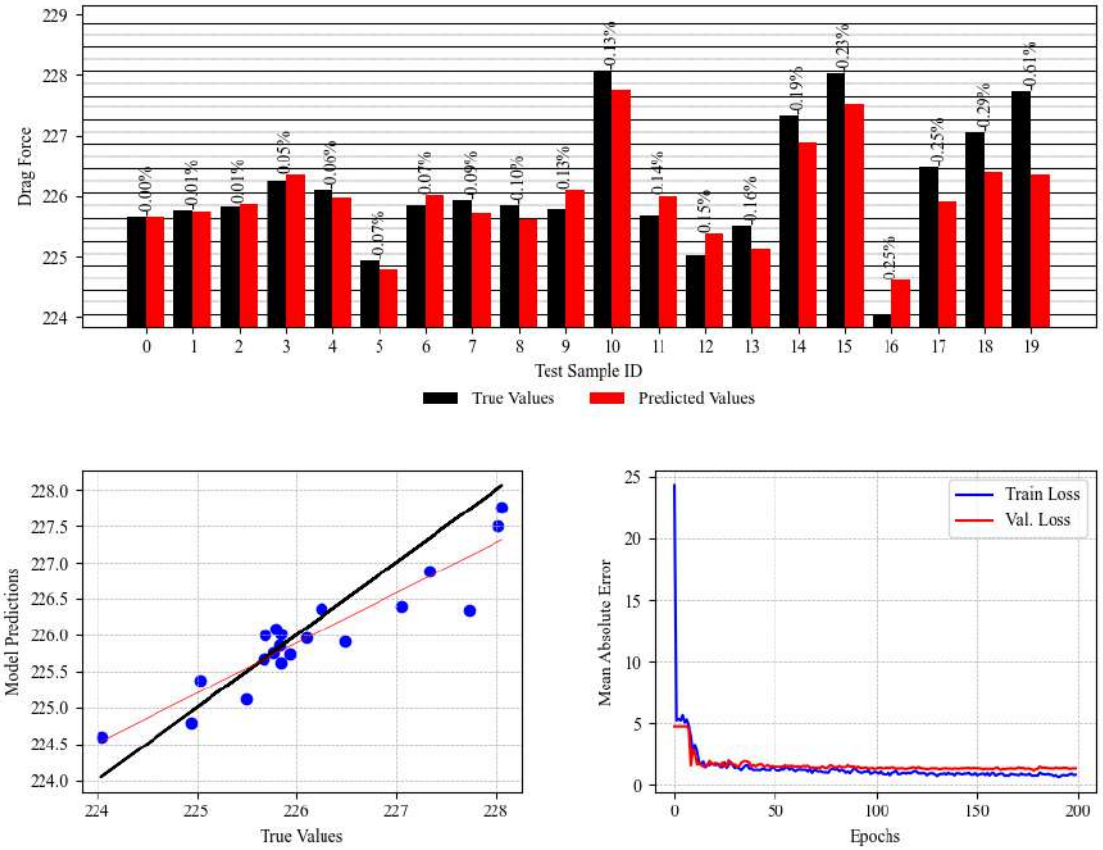



**Figure 5.11:** *Performance of the proposed SB model. (top) Comparison of the target Drag Force values to the predictions of the model, displaying the Relative Error on each sample's prediction ($MARE = 0.1497\%$). (left) Regression plot of the model, illustrating the agreement of its predictions with the target Drag Force values. (right) Convergence of the MAE of the model over the training epochs.*

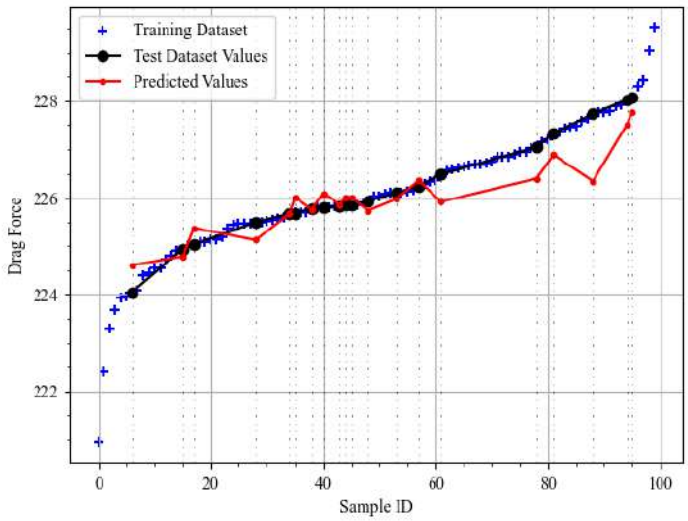



**Figure 5.12:** *The sorted working dataset. Blue crosses illustrate the training samples, while black dots illustrate the test samples and red dots the corresponding predictions of the proposed SB model.*

### 5.4.2 Surface Area Prediction

In the Surface Area prediction case, EAs proposed the model of Table [5.4]. Compared to the SB Elite for Drag prediction, this network presents a far-better overall performance, as depicted in Fig. [5.13]. Specifically, its predictions have a $MARE$ of 0.0104% to the target values. Their regression line has a slope of 0.9492 and deviates by 1.49° to the identity line. The error distribution across the range of the test dataset is slightly uneven, but the regression plot showcases a guaranteed level of accuracy in all regions. However, in Fig. [5.14] it is notable that some predictions deviate significantly to the ground truth, raising concerns for an inadequate architecture in the general context of the Thesis.

| **Parameter** | **Value** | **Parameter** | **Value** |
|---|---|---|---|
| Number of CNN Layers | 7 | Number of DNN Layers | 1 |
| CNN L1 filter size (pow. of 2) | 4 | DNN L1 neurons (pow. of 2) | 7 |
| CNN L2 filter size (pow. of 2) | 4 | DNN L2 neurons (pow. of 2) | – |
| CNN L3 filter size (pow. of 2) | 2 | DNN L3 neurons (pow. of 2) | – |
| CNN L4 filter size (pow. of 2) | 7 | DNN L4 neurons (pow. of 2) | – |
| CNN L5 filter size (pow. of 2) | 6 | DNN L5 neurons (pow. of 2) | – |
| CNN L6 filter size (pow. of 2) | 9 | DNN L6 neurons (pow. of 2) | – |
| CNN L7 filter size (pow. of 2) | 4 | DNN L7 neurons (pow. of 2) | – |
| CNN L8 filter size (pow. of 2) | – | DNN L8 neurons (pow. of 2) | – |
| act. function CNN layers | GELU | act. function DNN layers | ReLU |
| kernel size (constant) | (3, 3) | batch size | 16 |
| strides (constant) | (1, 1) | epochs | 200 |
| pool size (constant) | (2, 2) | | |

**Table 5.4:** *Summary of the proposed SB model after the exploration of the design space by EASY.*

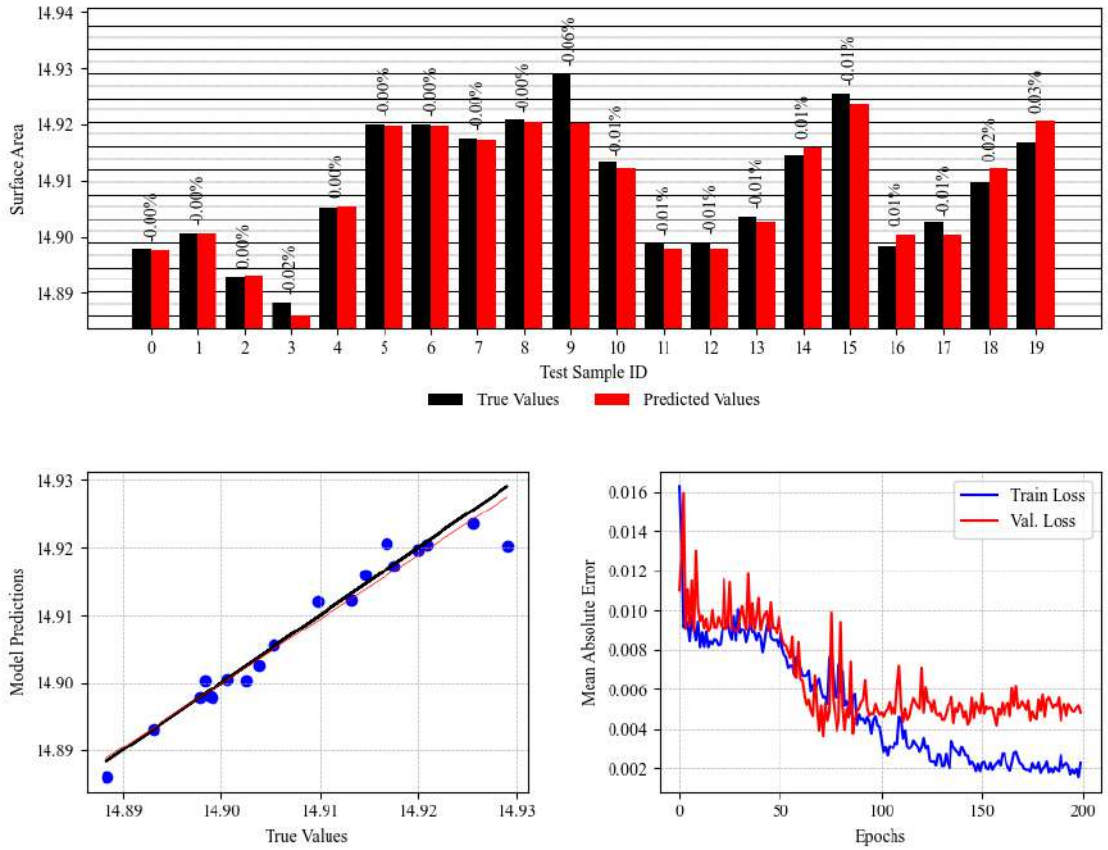



**Figure 5.13:** *Performance of the proposed SB model. (top) Comparison of the target Surface Area values to the predictions of the model, displaying the Relative Error on each sample's prediction (MARE:* 0.0104%*). (left) Regression plot of the model, illustrating the agreement of its predictions with the target Surface Area values. (right) Convergence of the MAE of the model over the training epochs.*

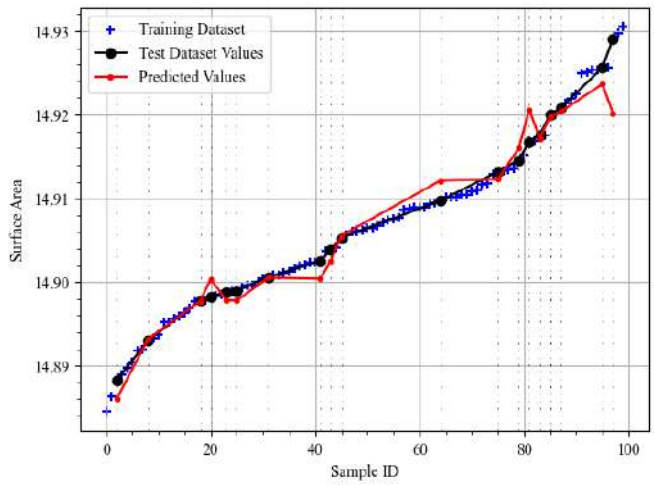



**Figure 5.14:** *The sorted working dataset. Blue crosses illustrate the training samples, while black dots illustrate the test samples and red dots the corresponding predictions of the proposed SB model.*

## 5.5 Multi-Branch Model (IMB - SMB Configurations)

The general architecture of the Multi-Branch Model is illustrated in Fig. [5.15]. The core idea is that more information can be extracted from multiple orthogonal views, allowing for greater precision in the predictions, provided that the combination of the different features is properly addressed.

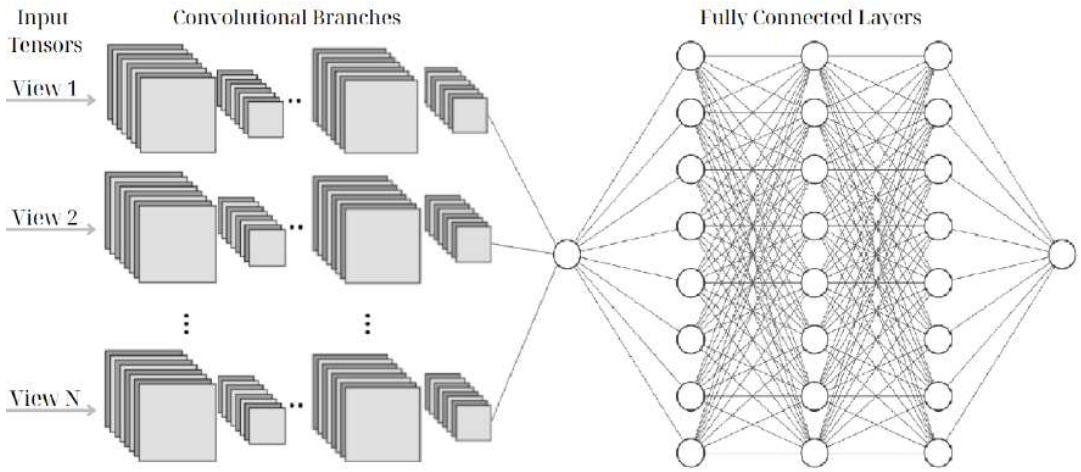



**Figure 5.15:** *Illustration of the general architecture of the SMB configuration.*

### 5.5.1 Drag Force Prediction - IMB, SMB

#### Individualized Multi Branch Configuration - IMB

In the Individualized Multi Branch configuration, the EASY software is employed to define the total number of 2D Convolutional Layers for each branch separately, along with the parameters of each layer. The dense halve of the network is examined similarly to the SB configuration. After exploration of the design space, fine-tuning proposed the model summarized in Table [5.5].

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Number of Side Branch (B1) Layers | 4 | Number of Rear Branch (B2) Layers | 6 |
| Number of Top Branch (B3) Layers | 7 | Number of DNN Layers | 4 |
| B1 L1 filter size (pow. of 2) | 5 | B2 L1 neurons (pow. of 2) | 5 |
| B1 L2 filter size (pow. of 2) | 4 | B2 L2 neurons (pow. of 2) | 7 |
| B1 L3 filter size (pow. of 2) | 8 | B2 L3 neurons (pow. of 2) | 5 |
| B1 L4 filter size (pow. of 2) | 2 | B2 L4 neurons (pow. of 2) | 3 |
| B1 L5 filter size (pow. of 2) | – | B2 L5 neurons (pow. of 2) | 2 |
| B1 L6 filter size (pow. of 2) | – | B2 L6 neurons (pow. of 2) | 5 |
| B1 L7 filter size (pow. of 2) | – | B2 L7 neurons (pow. of 2) | – |
| B3 L1 filter size (pow. of 2) | 7 | B2 L1 neurons (pow. of 2) | 6 |
| B3 L2 filter size (pow. of 2) | 8 | B2 L2 neurons (pow. of 2) | 7 |
| B3 L3 filter size (pow. of 2) | 7 | B2 L3 neurons (pow. of 2) | 7 |
| B3 L4 filter size (pow. of 2) | 5 | B2 L4 neurons (pow. of 2) | 7 |
| B3 L5 filter size (pow. of 2) | 7 | B2 L5 neurons (pow. of 2) | – |
| B3 L6 filter size (pow. of 2) | 6 | | |
| B3 L7 filter size (pow. of 2) | 3 | | |
| act. function B1, B2, B3 layers | ReLU | act. function DNN layers | GELU |
| kernel size (constant) | (3, 3) | batch size | 16 |
| strides (constant) | (1, 1) | epochs | 250 |
| pool size (constant) | (2, 2) | | |

**Table 5.5:** *Summary of the proposed IMB model after the exploration of the design space by EASY.*

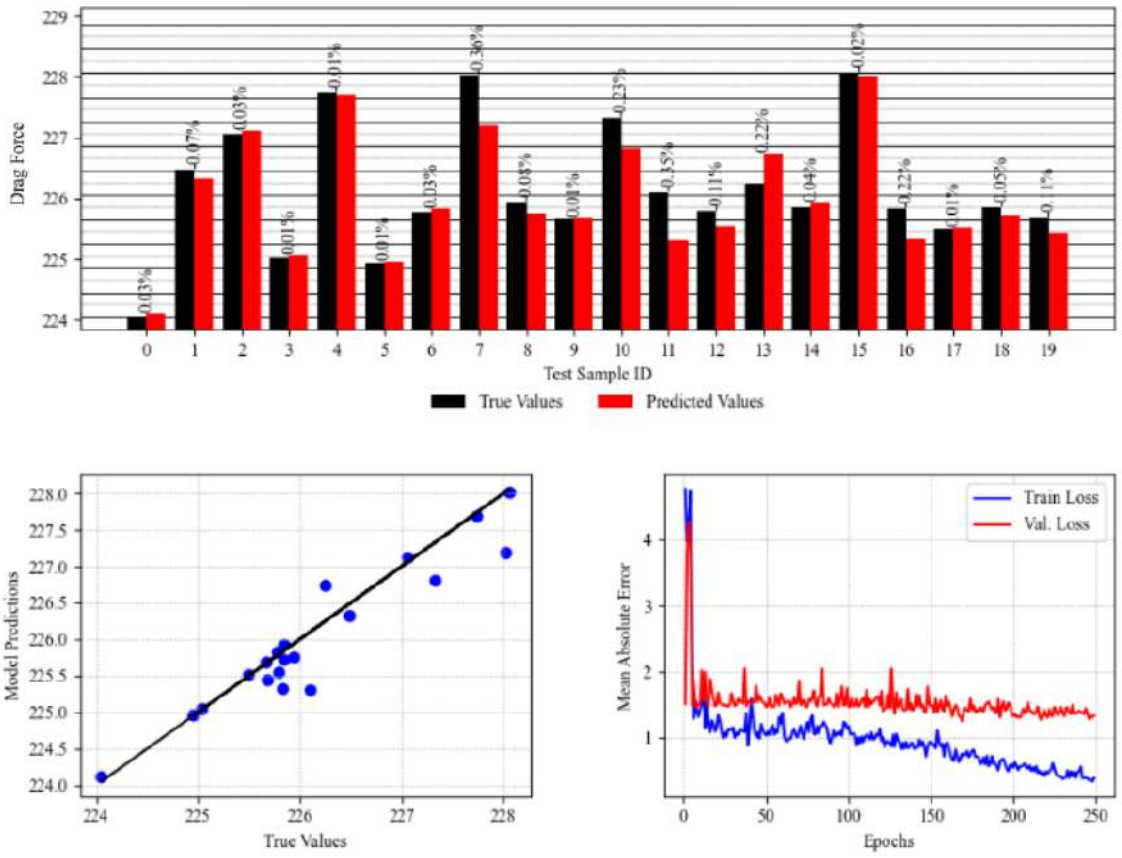



**Figure 5.16:** *Performance of the proposed IMB model. (top) Comparison of the target Drag Force values to the predictions of the model, displaying the Relative Error on each sample's prediction (MARE:* 0.1008%*). (left) Regression plot of the model, illustrating the agreement of its predictions with the target Drag Force values. (right) Convergence of the MAE of the model over the training epochs.*

The regression plot of Fig. [5.16] clearly shows that the model under-performs and struggles to predict the Drag Force accurately throughout the entire range of the dataset. Specifically, certain test samples are predicted with great accuracy, while the rest present a significant deviation to the target values. The overall test $MARE$ is computed 0.1008% ($MRE = -0.0632\%$). In order to achieve a better understanding of the situation, Fig. [5.17] is provided, which has a striking resemblance to Fig. [2.4 (*right*)]. The model is over-fitted in regions of the training dataset with high sample concentration, mainly concerning the interquartile region of the dataset. This leads to the presence of large deviations in test samples close-by. On the contrary, the model performs well in regions with fewer training samples, like the one concerning the first 40 (sorted) training samples.

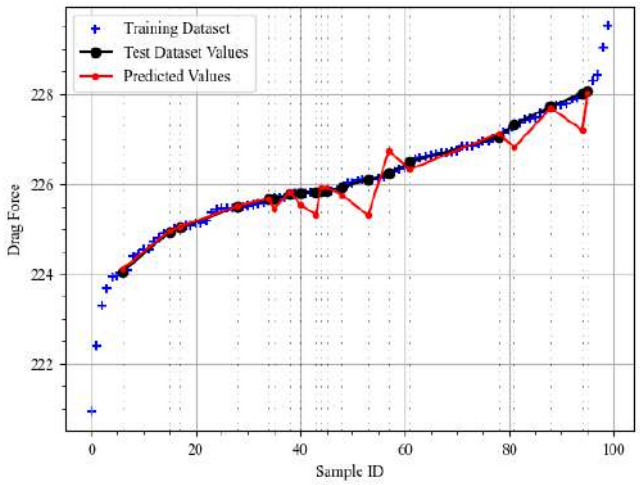



**Figure 5.17:** *The sorted working dataset. Blue crosses illustrate the training samples, while black dots illustrate the test samples and red dots the corresponding predictions of the proposed IMB model.*

## Shared Multi Branch Configuration - SMB

In the SMB implementation, the EASY software is employed to define, among other parameters, the shared architectural composition (and component parameters) of the Convolutional branches. Fine-tuning proposed the model summarized in Table [5.6], which features substantially fewer trainable parameters than the one proposed in the IMB case, and presents a more robust performance throughout the entire range of the dataset. It achieves a $MARE$ of 0.04532% ($MRE = 0.0072\%$). The predictions' regression line deviates by 2.36° from the identity line. Fig. [5.18]

displays a smooth distribution of error, and highlights the increased accuracy and generalization ability of the model, when compared to IMB Elite.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Number of CNN Layers | 7 | Number of DNN Layers | 3 |
| CNN L1 filter size (pow. of 2) | 7 | DNN L1 neurons (pow. of 2) | 9 |
| CNN L2 filter size (pow. of 2) | 4 | DNN L2 neurons (pow. of 2) | 8 |
| CNN L3 filter size (pow. of 2) | 7 | DNN L3 neurons (pow. of 2) | 12 |
| CNN L4 filter size (pow. of 2) | 5 | DNN L4 neurons (pow. of 2) | – |
| CNN L5 filter size (pow. of 2) | 4 | DNN L5 neurons (pow. of 2) | – |
| CNN L6 filter size (pow. of 2) | 6 | DNN L6 neurons (pow. of 2) | – |
| CNN L7 filter size (pow. of 2) | 2 | DNN L7 neurons (pow. of 2) | – |
| CNN L8 filter size (pow. of 2) | – | DNN L8 neurons (pow. of 2) | – |
| act. function CNN layers | ReLU | act. function DNN layers | GELU |
| kernel size (constant) | (3, 3) | batch size | 16 |
| strides (constant) | (1, 1) | epochs | 200 |
| pool size (constant) | (2, 2) | | |

**Table 5.6:** *Summary of the proposed SMB model after the exploration of the design space by EASY.*

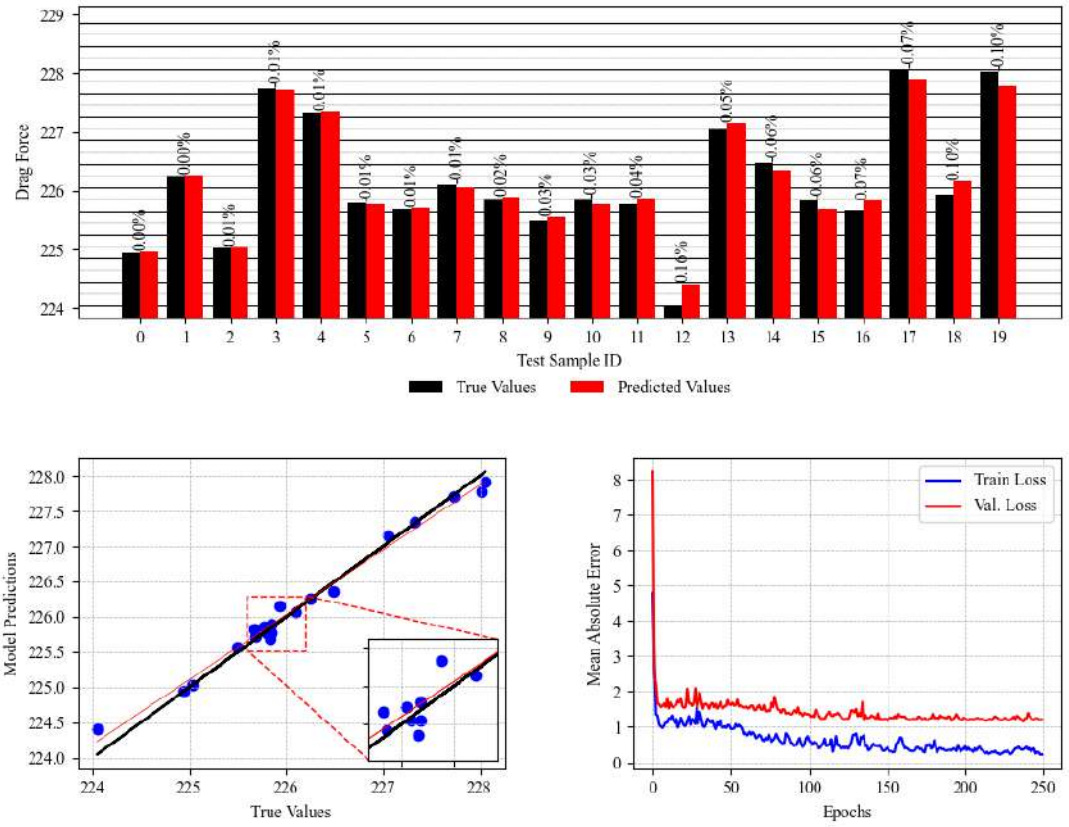



**Figure 5.18:** *Performance of the proposed SMB model. (top) Comparison of the target Drag Force values to the predictions of the model, displaying the Relative Error on each sample's prediction (MARE:* 0.04532%*). (left) Regression plot of the model, illustrating the agreement of its predictions with the target Drag Force values. (right) Convergence of the MAE of the model over the training epochs.*

Fig. [5.19] showcases the proper fitting of the model, presenting a slightly elevated deviation of the predictions and target values in regions mainly closer to the bounds of the dataset. In the central region, which concerns the majority of the samples, the model manages to properly identify and decode patterns in the input images, and translate them to produce predictions of high accuracy.

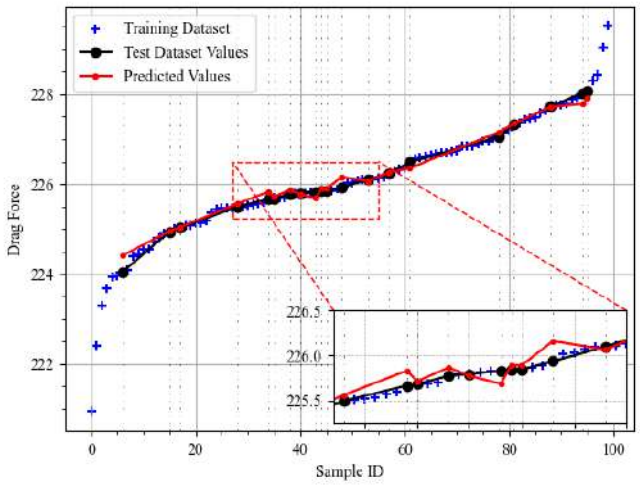



**Figure 5.19:** *The sorted working dataset. Blue crosses illustrate the training samples, while black dots illustrate the test samples and red dots the corresponding predictions of the proposed SMB model.*

The above results demonstrate that, herein, the Shared Multi-Branch Architecture is superior to the Individualized one and manages to accurately predict the cars' Drag Forces. It presents a better overall performance, with a more evenly distributed error and a significantly smaller $MARE$ (almost $2\times$ smaller). It is also notable that the evolutionary algorithms employed to search the optimal Network architectures performed more than $2.5\times$ more evaluations on the Individualized-Branch case, yet the Shared-Branch configuration outperformed it.

These intriguing results and observations, indicating the SMB configuration's dominance, are elaborated in a more detailed discussion in the "Summary and Comparison" subsection.

The superior Shared Multi-Branch configuration is selected and used both in the Drag Force and Surface Area prediction of the Multi-Branch subsection of Application II.

### 5.5.2 Surface Area Prediction - SMB

The above study verifies the Shared-Branch Architecture's superior performance; thus, the same architecture is used for the surface area prediction. Fine-tuning proposes the model of Table [5.7], featuring an increased complexity in its Convolutional-based first half, and an almost "shallow" second half, with a single hidden layer of 64 neurons.

| **Parameter** | **Value** | **Parameter** | **Value** |
|---|---|---|---|
| Number of CNN Layers | 8 | Number of DNN Layers | 1 |
| CNN L1 filter size (pow. of 2) | 4 | DNN L1 neurons (pow. of 2) | 6 |
| CNN L2 filter size (pow. of 2) | 3 | DNN L2 neurons (pow. of 2) | – |
| CNN L3 filter size (pow. of 2) | 3 | DNN L3 neurons (pow. of 2) | – |
| CNN L4 filter size (pow. of 2) | 2 | DNN L4 neurons (pow. of 2) | – |
| CNN L5 filter size (pow. of 2) | 7 | DNN L5 neurons (pow. of 2) | – |
| CNN L6 filter size (pow. of 2) | 2 | DNN L6 neurons (pow. of 2) | – |
| CNN L7 filter size (pow. of 2) | 5 | DNN L7 neurons (pow. of 2) | – |
| CNN L8 filter size (pow. of 2) | 4 | DNN L8 neurons (pow. of 2) | – |
| act. function CNN layers | ReLU | act. function DNN layers | ReLU |
| kernel size (constant) | (3, 3) | batch size | 16 |
| strides (constant) | (1, 1) | epochs | 300 |
| pool size (constant) | (2, 2) | | |

**Table 5.7:** *Summary of the proposed SMB model after the exploration of the design space by EASY.*

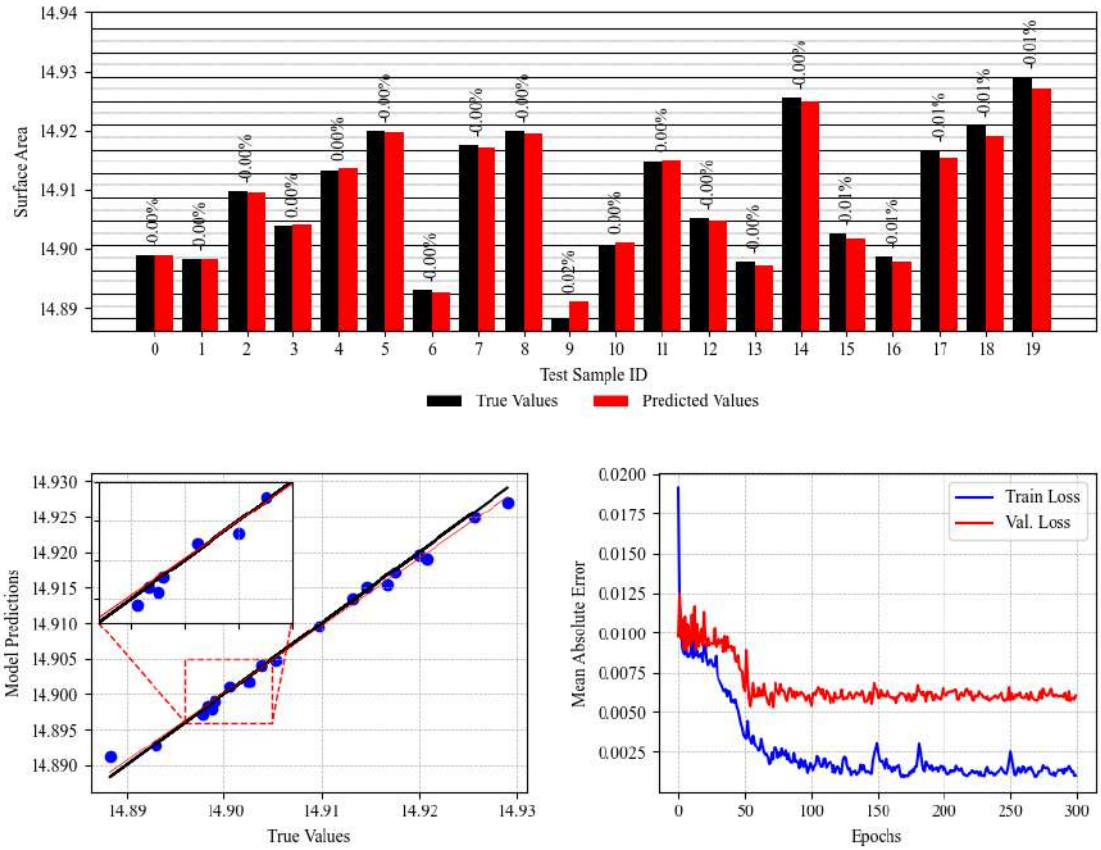



**Figure 5.20:** *Performance of the proposed SMB model. (top) Comparison of the target Surface Area values to the predictions of the model, displaying the Relative Error on each sample's prediction (MARE:* 0.0048%*). (left) Regression plot of the model, illustrating the agreement of its predictions with the target Surface Area values. (right) Convergence of the MAE of the model over the training epochs.*

Fig. [5.21] displays a really satisfying performance of the Elite, and a strong and consistent agreement between its predictions and the target surface area values, even close to the bounds, where the training samples are considerably fewer. The predictions' regression line has a slope of 0.9777 and presents an angle difference of 0.65° with the identity line. The model performed with a *MARE* of 0.0048% and an *MRE* of −0.0021%, when evaluated on the test dataset.

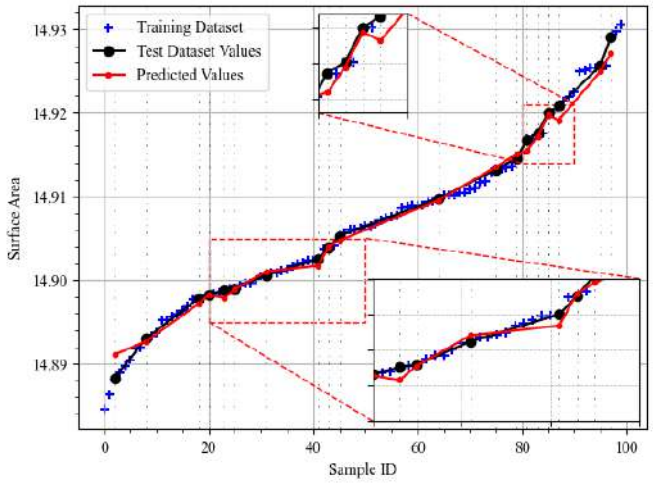



**Figure 5.21:** *The sorted working dataset. Blue crosses illustrate the training samples, while black dots illustrate the test samples and red dots the corresponding predictions of the proposed SMB model.*

## 5.6 Summary and Comparison

This subsection offers a thorough comparison of the examined models, and a commentary justifying the selection of the superior SMB configuration as the working configuration in the subsequent Application III. At the end of the subsection, Table [5.8] provides a compact presentation of the different models' evaluation metrics.

The studies and results of Application II highlight the superiority of the MB Configuration over the SB Architecture Model. The latter proves to be inadequate for the purposes of this Thesis; either the information encoded in the R34 inputs is not sufficient to provide accurate results, or the SB architecture is naturally incapable of capturing and processing patterns in the input tensors of this study.

In the surface area case, the SB configuration performs with a questionably sufficient accuracy and a *MARE* of 0.0104%. Fig. [5.14] presents both regions of high accuracy and ones of low credibility and precision, resulting in an uneven distribution of Error across the dataset. However, excluding a few predictions, the network "decodes" changes in the depicted geometries successfully and captures the target values to a satisfying extent. On the one hand, the complexity of this Applica-

tion's task justifies a couple of inaccuracies on the predictions (to a certain extent). However, being the product of a computationally heavy tuning process, the SB Model does not live up to its expectations, thus raising concerns over its predictive capabilities as a whole.

These concerns are amplified in the Drag Force prediction case, which proves the SB Configuration to be completely unfit for this Application's purposes. After a total of 250 EA evaluations, the proposed model performs poorly and presents a high dispersion in its predictions' regression plot, as displayed in Fig. [5.14]. It is clear that, especially closer to the dataset's bounds, the network fails to properly interpret changes in the depicted cars. The MARE has a value of 0.1497% and the predictions' regression line has a slope of 0.6911, presenting an angle difference of $10.35^o$ with the identity line. Overall, the drag prediction case highlights the SB Configuration's inability to form a robust and reliable model for the purposes of this Application.

In order to properly compare the examined networks based on their accuracy and distribution of error, the following errors are formulated.

$$MARE_{Tr} = \frac{1}{n}\sum_{i=0}^{n} ARE_{Tr,i} = \frac{1}{n}\sum_{i=0}^{n} \frac{|y_i - \hat{y}_i|}{y_i} 100\% \tag{5.2}$$

$$MAE^N_{Tr} = \frac{1}{n}\sum_{i=0}^{n} AE^N_{Tr,i} = \frac{1}{n}\sum_{i=0}^{n} \frac{|y_i - \hat{y}_i|}{y^{max} - y^{min}} 100\% \tag{5.3}$$

where $y$ and $\hat{y}$ denote the standardized target values and predictions respectively. Eq [5.2] refers to the $MARE$ applied on the transformed data. Eq [5.3] refers to the normalized $MAE$ applied on the transformed data, which is the highest when the opposite bound of the range is predicted.

The introduced MB Configuration produces much more accurate predictions, outperforming the inadequate SB Networks. The study conducted, comparing the two different approaches of the proposed idea, allows for some very interesting observations, as well as promising results.

The IMB Configuration relies on the separate adjustment of each convolutional branch, allowing for a fine-tuning of their (sequence of) processing units, depending on each view's needs. The multiple input images contain more encoded information, allowing for a better performance and higher accuracy, provided the model is properly trained. On the downside, the EASY software explores a 32-dimensional design space during the optimization of the model's architecture, introducing a significant

computational cost that should be taken into consideration when evaluating the IMB Configuration as a whole.

Fig. [5.22] and [5.23] presents the Distribution of $MARE_{Tr}$ and $MAE^{N}_{Tr}$ of the final, Elite IMB model, compared to the equivalent SB Model in both examined cases.

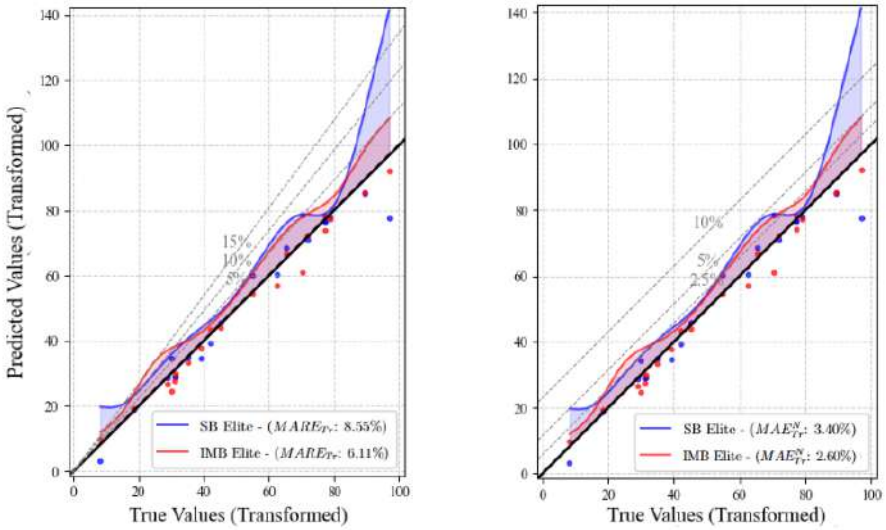



**Figure 5.22:** *Illustration of the (distribution of the) $ARE_{Tr}$ (left) and $AE^{N}_{Tr}$ (right) of the SB (blue) and IMB (red) Elites of the Surface Area prediction case.*

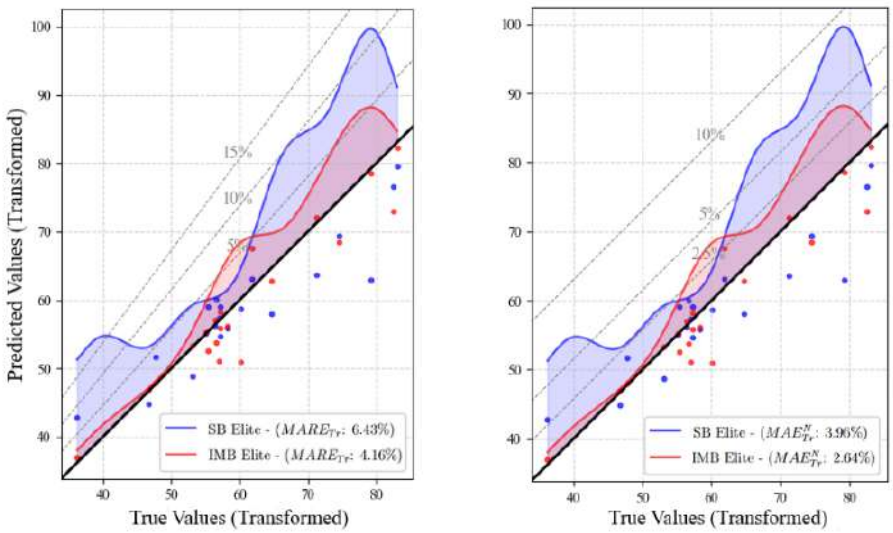



**Figure 5.23:** *Illustration of the (distribution of the) $ARE_{Tr}$ (left) and $AE^{N}_{Tr}$ (right) of the SB (blue) and IMB (red) Elites of the Drag Force prediction case.*

In the surface area case, the two networks perform similarly, providing accurate results and achieving small Errors across the entire range of the test dataset, excluding the bounds where the SB model performs with some inaccuracy. The IMB network presents a smoother, more even Distribution of Error and achieves a smaller $MARE$. The drag prediction case also highlights the Individualized-Branch model's superiority, which achieves a $MARE_{Tr}$ of 4.16% and a $MAE^{N}_{Tr}$ of 2.64%, and presents a significantly smoother error distribution.

The displayed results validate the IMB Configuration's ability to capture modifications in the depicted cars and interpret them accordingly, to provide reliable predictions over their geometric properties and aerodynamic behavior. The IMB model seems to accurately identify and translate the prominent changes in the depicted geometries, which are almost entirely responsible for most the drag's change, however it seemingly fails to capture smaller, more discrete fluctuations that affect it in a smaller scale. As a result, two issues persist; firstly, the error distribution is questionably smooth and follows a morphology seemingly uncorrelated with the sample density in the training dataset. Additionally, the overall performance of the model is not proportional to the large computational cost imposed by the fine-tuning process of its 32 Design Variables. As briefly mentioned earlier, although the IMB Configuration's design space potentially contains a better global solution, due to its high dimensionality, it also presents much higher complexity and a larger number of local optimal solutions, posing a risk of insufficient tuning and selection of a suboptimal composition. Since the goal of this study is to propose a surrogate model to avoid excessive exploration and substitute computationally costly methods, the IMB network's performance does not outweigh the cost needed to construct it.

Implementation of the SMB Configuration successfully addressed both issues in a simultaneous manner. Reduction of the dimensionality of the design space from 32 to 20 dramatically dropped the computational cost of the network's optimization by nearly 60%, and evidently prevented EASY from sticking at local minima. Moreover, the introduced forced restriction likely acts as a regularization mechanism, constraining the model's generalization ability and preventing it from overfitting on the (already small) training dataset samples. Additionally, the shared depth of the convolutional branches ensures the same level of abstraction for each view, crucial for the proper interpretation of the concatenation product by the dense part of the model.

Fig. [5.24] and [5.25] display the SMB Model outperforming the IMB one, both in the Surface Area case and the Drag prediction case.

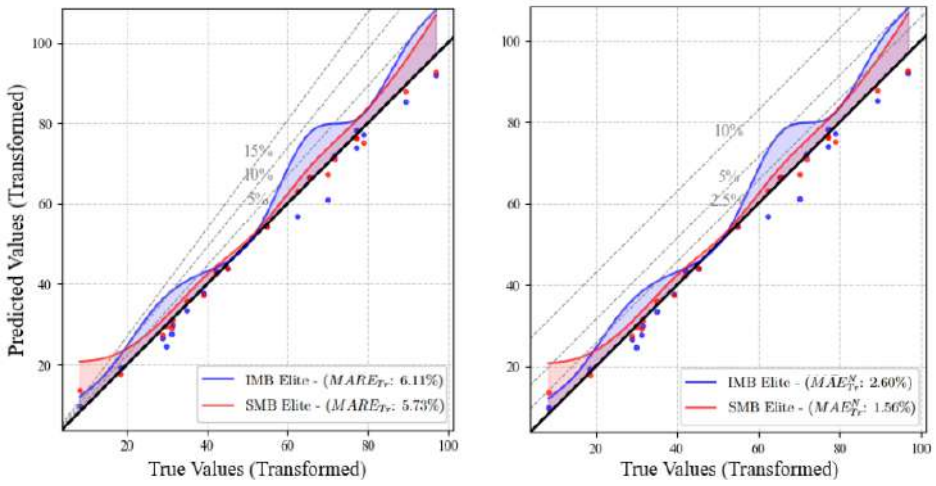



**Figure 5.24:** *Illustration of the (distribution of the) $ARE_{Tr}$ (left) and $AE^N_{Tr}$ (right) of the IMB (blue) and SMB (red) Elites of the Surface Area prediction case.*

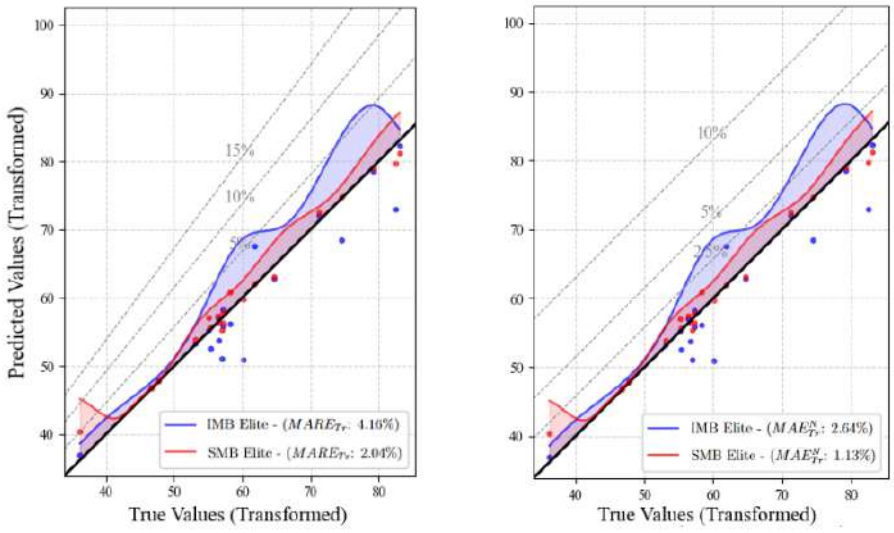



**Figure 5.25:** *Illustration of the (distribution of the) $ARE_{Tr}$ (left) and $AE^N_{Tr}$ (right) of the IMB (blue) and SMB (red) Elites of the Drag Force prediction case.*

In Fig. [5.24], concerning the surface area case, the SMB model presents a really smooth distribution of error across most of the dataset, particularly the inner regions. The $MAE^N_{Tr}$ only becomes greater than 5% at the first test sample's prediction, which belongs to a region of low sample concentration in the dataset. The rest of the distribution curve presents a remarkable smoothness, showcasing a defi-

nite improvement from the IMB Configuration, as far as surface area prediction is concerned.

In the case of drag prediction, the SMB network is still superior, as depicted in Fig. [5.25]. The Errors take values smaller than 5% for the $MARE_{Tr}$ and 2.5% for the $MAE^N_{Tr}$ across the entire dataset (again, excluding the first prediction); interestingly, here the SMB model performs better than in the surface area case, which is, however, justified by the "positioning" of the test samples in the working dataset.

| | **Surface Area (original domain)** | **Drag Force (original domain)** |
|---|---|---|
| SB | $MARE : 0.0104\%$ , $MRE : -0.0037\%$ | $MARE : 0.1497\%$ , $MRE : -0.0684\%$ |
| IMB | $MARE : 0.008\%$ , $MRE : -0.006\%$ | $MARE : 0.1008\%$ , $MRE : -0.0632\%$ |
| SMB | $MARE : 0.0048\%$ , $MRE : -0.0021\%$ | $MARE : 0.0453\%$ , $MRE : 0.0072\%$ |
| | **Design Space Dimentionality** | **Overall Cost** |
| SB | 20 design variables | 1 TU |
| IMB | 32 design variables | $\sim$ 12 TU |
| SMB | 20 design varibales | $\sim$ 5.5 TU |

**Table 5.8:** *Summary of the presented configurations' performances, cost and efficiency wise.*

## 5.7 Conclusions

The studies of Application II demonstrated an undeniable superiority of the Shared Multi-Branch Configuration in the context of this Thesis; despite the complexity of the task, it manages to provide accurate predictions with little fluctuations in the error distribution, while not requiring excessive computational resources to fine tune (approximately 40 hours on 2 $\times$ GeForce RTX3060). The promising results presented above indicate an increased generalization ability and a guaranteed level of predictive accuracy, provided a properly-calibrated cooperation of SE-Blocks and Regularizers. Thus, the SMB structure is selected and used in the subsequent Application III.

# Chapter 6
# Application III - Automobile's Drag Force (3 Morphing Boxes)

## 6.1 Introduction

Application III concerns an implementation of the studies conducted earlier in the Thesis, to construct an ML model that produces high-precision predictions of different automobiles' aerodynamic drag values. This section's car samples present more subtle geometry fluctuations. The following study aims to prove the presented SMB configuration's efficiency, accuracy and generalization ability, provided a proper tuning of the model's parameters and hyperparameters, proposing it as a promising surrogate alternative to costly CFD methods in early stages of design, where extreme precision is not yet necessary.

## 6.2 LHS-based Dataset Generation

Contrary to Application II, this application's dataset consists of a tot al of 366 variations of the DrivAer car model. These variations are constructed by employing the FFD technique with 3 morphing boxes partially embedding the baseline geoometry. The boxes, positioned as depicted in Fig. [6.1], contain a total of 270 CPs (each) arranged in a $6 \times 5 \times 9$ grid. CPs are categorized in $yRows$ ($xz$ plane) and $xRows$ ($yz$ plane) (as depicted in Fig. [6.1]) and are displaced with the LHS method; $yRows$ are allowed to move by $\pm 0.07$ and $\pm 0.033$ in the $-x$ and $-z$ directions respectively, while $xRows$ move by $\pm 0.052$ and $\pm 0.03$ in the $-x$ and $-z$ directions (Absolute Values).

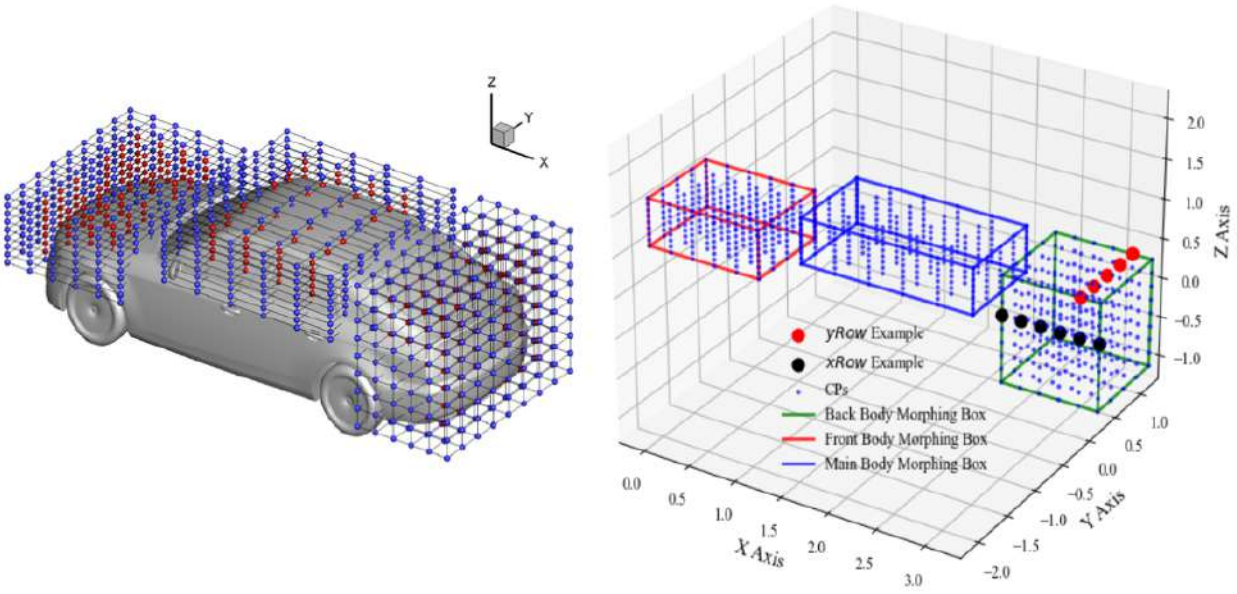



**Figure 6.1:** *Illustration of the Morphing boxes defined to develop the dataset of Application III. (left) The baseline DrivAer geometry and the CPs of the three morphing boxes; blue CPs are fixed while red CPs are allowed to move as described in the previous paragraph. (right) Simplistic illustration of the three control grids, exemplifying the categorization of the CPs in yRows and xRows.*

A total of $N_{db,III} = 366$ modified cars are generated ($\sim$ 185 hours, excluding all complications and re-runs). Images of the geometries are taken and preprocessed in a manner identical to Application II. The same flow as in the previous Application (summarized in Table [5.1]) is solved using the PUMA CFD Solver.

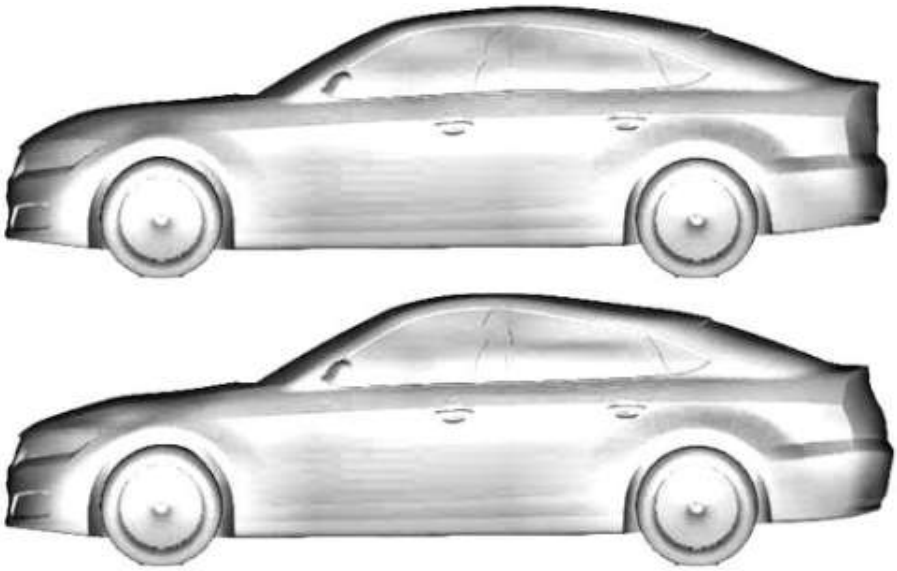


**Figure 6.2:** *Two samples of the SMB model's training dataset (side view). The samples are fed into the model in this form.*

Fig. [6.2] depicts two samples of the training dataset from the side view, displaying the subtle modifications applied to the baseline DrivAer model. In contrast to Application II, four views of the cars are exploited; the front, side, rear and top view, each with a Convolutional branch uniquely associated with it. The images are fed to the model in tensors of fixed size for each view. Specifically, the tensor sizes are $[312 \times 224]$ for the front and rear view, $[752 \times 240]$ for the side view and $[712 \times 312]$ for the top view.

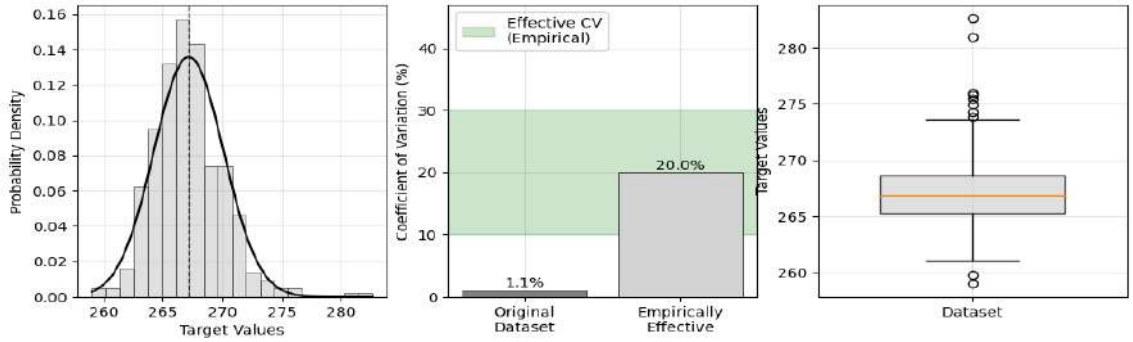



**Figure 6.3:** *Distributional summary of the original output dataset (left) The dataset's distribution compared to the corresponding normal curve. (center) Comparison of the dataset's coefficient of variation CV against an empirically effective CV range.(right) The dataset's boxplot.*

As depicted in Fig. [6.3], the working dataset once again poses challenges to the training process naturally; it presents a mean value of $\mu = 267.1737$ and a variance of $\sigma^2 = 8.6197$ ($CV = 1.10\%$). Unlike Application II, it is highly skewed (skewness $= 0.9183$) and non-normal according to the Shapiro-Wilk test.

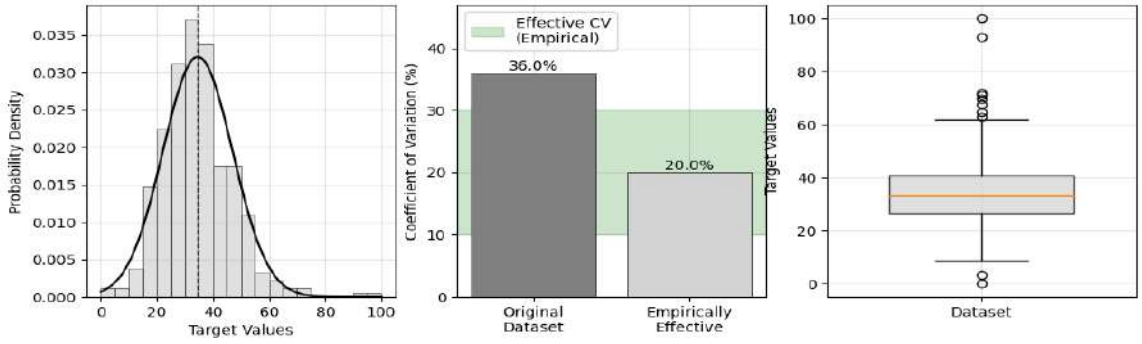



**Figure 6.4:** *Distributional summary of the transformed output dataset (left) The dataset's distribution compared to the corresponding normal curve. (center) Comparison of the dataset's coefficient of variation CV against an empirically effective CV range.(right) The dataset's boxplot.*

To address the original dataset's narrow dynamic range and extremely low variance, the linear transformation defined in Eq. (5.1) is once again applied. While

this operation successfully expands the range and improves numerical conditioning, the quality of the transformed dataset remains questionable. Specifically, the resulting coefficient of variation is approximately 36%, which exceeds the established preferable range. Although high CV values can indicate excessive variability and potential noise (increased risk of overfitting), 36% is still considered acceptable. The affine nature of the operation results in the preservation of the distribution's shape; the skewness remains unchanged, and the dataset continues to exhibit significant asymmetry and deviation from normality. These characteristics, while not ideal, are manageable through architectural choices within the model.

Rather than employing a more complex non-linear transformation—which could introduce interpretational complexity— a decision is made to retain this simple linear scaling. To regulate the effects of the dataset's idiosyncrasies, greater emphasis will be placed later on on the architectural components of the models, as well as the (hyper)parameters of the training process. A statistical summary of the dataset, pre- and post- transformation, is provided in Table [6.1].

| **Metric** | **Original Data** | **Transformed Data** |
|---|---|---|
| Sample Size ($n$) | 366 | 366 |
| Mean ($\mu$) | 267.1737 | 34.5432 |
| Standard Deviation ($\sigma$) | 2.9359 | 12.4346 |
| Variance ($\sigma^2$) | 8.6197 | 154.6198 |
| Coefficient of Variation (CV) | 1.10% | 36.00% |
| Skewness | 0.9183 | 0.9183 |
| Kurtosis | 2.9111 | 2.9111 |
| Range | 23.6109 | 100.00 |

**Table 6.1:** *Statistics of the output dataset before and after transformation.*

## 6.3 SMB Model Implementation

### 6.3.1 Drag Force Prediction - MAE Loss

The SMB configuration derived from the studies of Application II is used. Initially, the Elite model of Table [5.6] is trained on the current dataset. During training, the MAE of the predictions is monitored to guide the trainable parameters' adjustment. The model performed poorly, failing to both capture and translate patterns in the input data. Regardless of the test sample, the model predicted the test set's mean value $\mu$.

Enhancements are deemed necessary in order to increase the representational capacity and, ultimately, improve the predictive accuracy. Thus, a study is conducted to examine the architectural components and associated hyperparameters of the current model. The modifications derived from this analysis are summarized below.

- **Integration of SE-Blocks after every two Convolutional Layers:** The geometric variations in the current application are more subtle compared to those encountered in the previous Application and the selected SMB Elite evidently failed to capture them. SE-Blocks are now incorporated after every two Convolutional layers. This integration proves to enhance the network's interpretive capabilities.
- **Max-Pooling after every four Convolutional Layers:** Frequent use of Max-Pooling operations can lead to loss of important information due to downsampling, especially in this case. The study suggested the integration of these layers after every four Convolutional layers, which evidently balances abstraction and information preservation. However, this adjustment leads to an increased requirement of computational resources for training (and fine-tuning later on).

  These first two teps form the updated empirical rule for integrating SE Blocks and MaxPooling blocks in the developed networks. An application of this rule on a branch of 7 convolutional layers is displayed in Fig. [6.5].
- **Training Hyperparameter Optimization:** A batch size of 22 and a training duration of 200 epochs are defined, helping achieve a stable and accurate convergence.

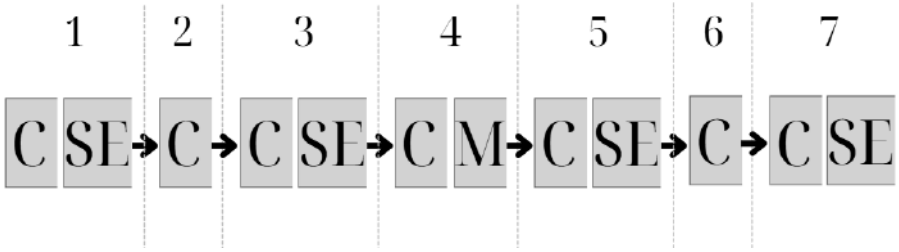



**Figure 6.5:** *Application of the presented modifications on a convolutional branch of 7 convolutional layers. Here, C denotes a 2D Convolutional Layer, M denotes a Max Pooling Layer and SE denotes a Squeeze-and-Excitation Block.*

- **Reduced SE-Block Reduction Ratio:** Lowering the SE-Block reduction ratio to $r = 11$ improves the model's performance. This is attributed to a finer detail in the recalibration process, allowing for more expressive representations. However, this imposes an increase in computational demand (as suggested in Eq (2.5) ), especially given the more frequent deployment of SE-Blocks.
- **Dropout Regularization in Dense Layers:** To aid generalization, Dropout is applied in every two dense layers with a dropout rate of 0.07. This technique proves effective in improving test dataset accuracy.

Application of these architectural revisions leads to the far-better performance of Fig. [6.6]. The updated SMB network is capable of identifying the subtle geometric modifications of the car depictions, but fails to properly interpret them.

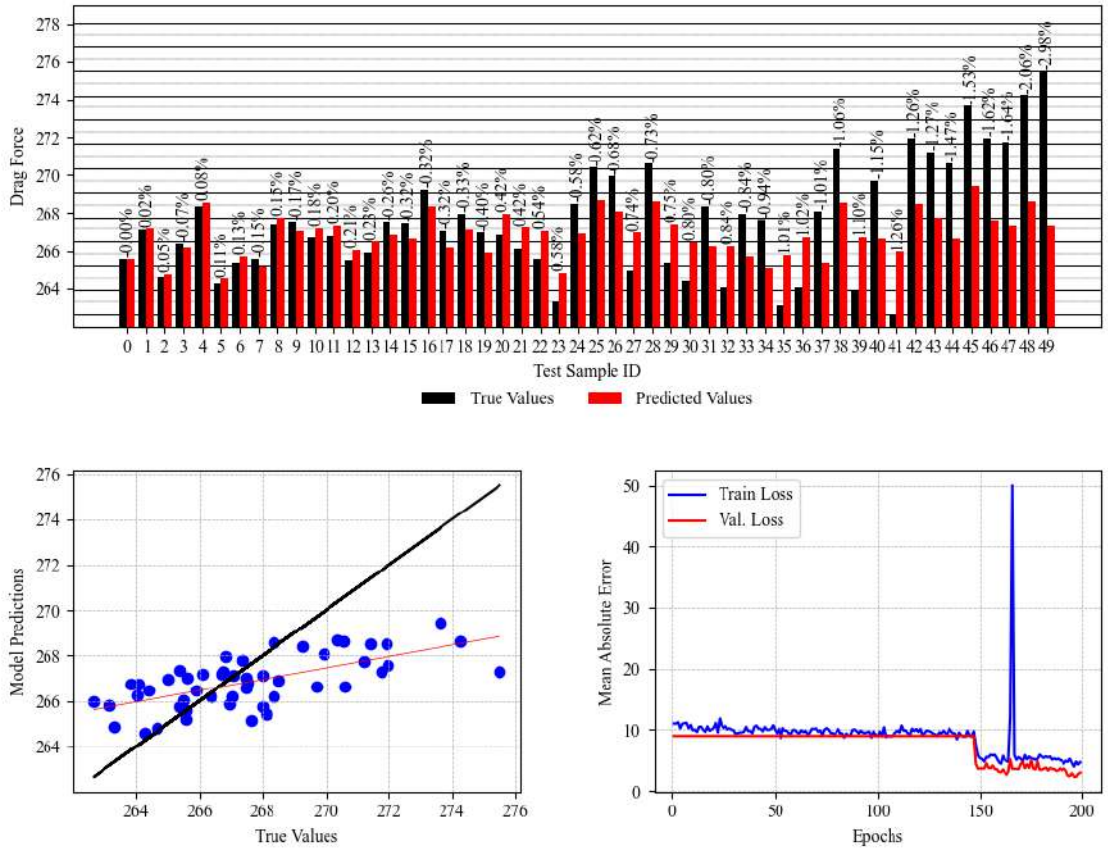



**Figure 6.6:** *Performance of the modified Elite Application II SMB model evaluated on this Application's dataset (top) Bar plot comparing the target Drag Force values to the predictions of the model, displaying the Relative Error on each sample's prediction (left) Regression plot of the model, illustrating the agreement of its predictions with the target Drag Force values. (right) Convergence of the MAE of the model over the training epochs.*

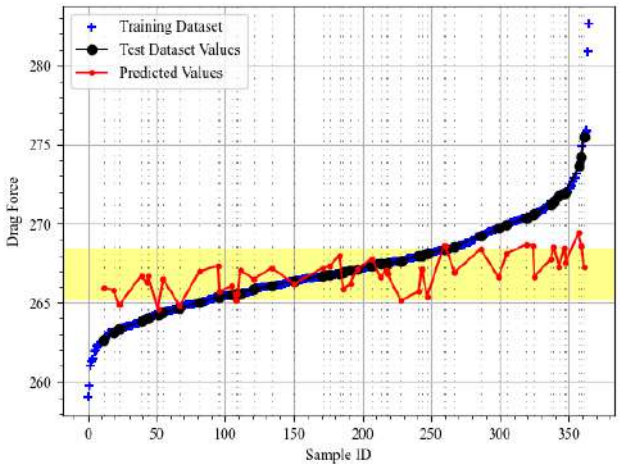



**Figure 6.7:** *The sorted working dataset. Blue crosses illustrate the training samples, while black dots illustrate the test samples and red dots the corresponding predictions of the modified SMB model. The highlighted region represents the $Q_1Q_3$ range.*

### 6.3.2 Bias-Variance Tradeoff and Loss modifications

The observed concentration of predictions within the interquartile range in Fig. [6.7] exemplifies a fundamental challenge in machine learning known as the *bias-variance tradeoff*. When the working dataset is of limited output variability, neural networks tend to develop a systematic bias toward the central tendency of the target distribution, effectively minimizing prediction error by converging on safe, middle-ground estimates [36]. This behavior is demonstrated by the Elite of Application II, which predicts the mean value $\mu$ of the output dataset on all samples. This phenomenon reduces variance in predictions and introduces an extremely problematic bias that constrains the model. To overcome this challenge, one approach would be to further enhance the model's complexity and increase the resources expended in fine-tuning, aiming to improve the network's interpretational capabilities. However that would ultimately lead to a dramatic computational drawback with an uncertain outcome [5], [39].

A different approach is followed, according to which more attention is paid on the statistical-wise agreement of the predictions with the target values. To implement this, a custom Loss function is formulated.

$$cLoss = \alpha \cdot \left|\sigma^2(y) - \sigma^2(\hat{y})\right| + (1-\alpha) \cdot MAE(y, \hat{y})$$

$$= a \cdot \left| \frac{1}{N}\sum_{j=1}^{N}(y_j - \mu_y)^2 - \frac{1}{N}\sum_{j=1}^{N}(\hat{y}_j - \mu_{\hat{y}})^2 \right| + (1-a) \cdot \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \qquad (6.1)$$

where $\mu_y$ and $\mu_{\hat{y}}$ are the mean values of the target value and prediction sets respectively, $N$ is the test dataset size and $\alpha$ is a coefficient defining the balance between the two terms.

Essentially, this custom Loss function penalizes not only the absolute difference of the predictions to the target values, but also the absolute difference of the variances of the two sets, thus introducing attention to the distributional properties of the error and explicitly adressing the bias-variance imbalance. The underlying idea is to push the model to first escape the highlighted region of Fig. [6.7] by producing predictions across the entire range of the dataset.

With an established (controlled) diversity in the predictions, ensured by the first term of *cLoss*, the quality of the model's performance is now a matter of proper training and evaluation by the second term in Eq [6.1]. The *cLoss* is now applied on the transformed output values and monitored during training. The batch size is increased to 40 to ensure a more representative variance in each batch.

### 6.3.3 Drag Force Prediction - Custom Loss

The coefficient $\alpha$ is initially set to 0.15. EASY is employed to fine-tune the updated model. After a total of 200 evaluations, with a parent-offpsing ratio of 3 and a population size of 30, the first elite performs as displayed in Fig. [6.8].

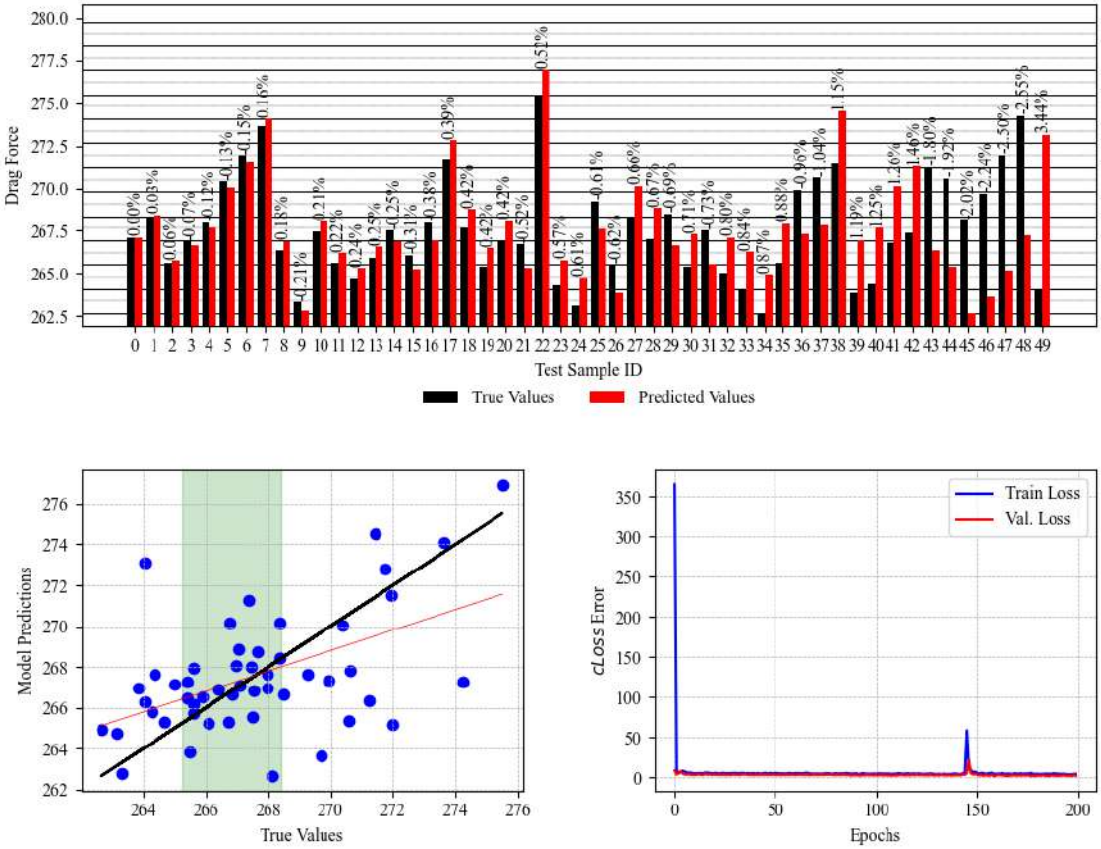



**Figure 6.8:** *Performance of the proposed SMB model with the cLoss coefficient $\alpha = 0.15$ (top) Bar plot comparing the target Drag Force values to the predictions of the model, displaying the Relative Error on each sample's prediction (left) Regression plot of the model, illustrating the agreement of its predictions with the target Drag Force values. The highlighted region represents the $Q_1Q_3$ range. (right) Convergence of cLoss (Train Loss) and MAE (Val. Loss) of the model over the training epochs.*

At first glance, this performance raises concerns about the use of the *cLoss* function. The high sparsity in the regression plot imitates randomness, and the errors associated with many samples take exceedingly high values. However, the predictions successfully range across the entire dataset. Specifically, the test and prediction sets have a variance of 158.4796 and 153.8926 respectively (or 8.8348 and 8.5791 in the original domain). Moreover, the inaccuracy is the most significant in regions outside the highlighted area containing the 50% of the dataset. Overall, employing the custom loss function addressed the lack of diversity in the predictions.

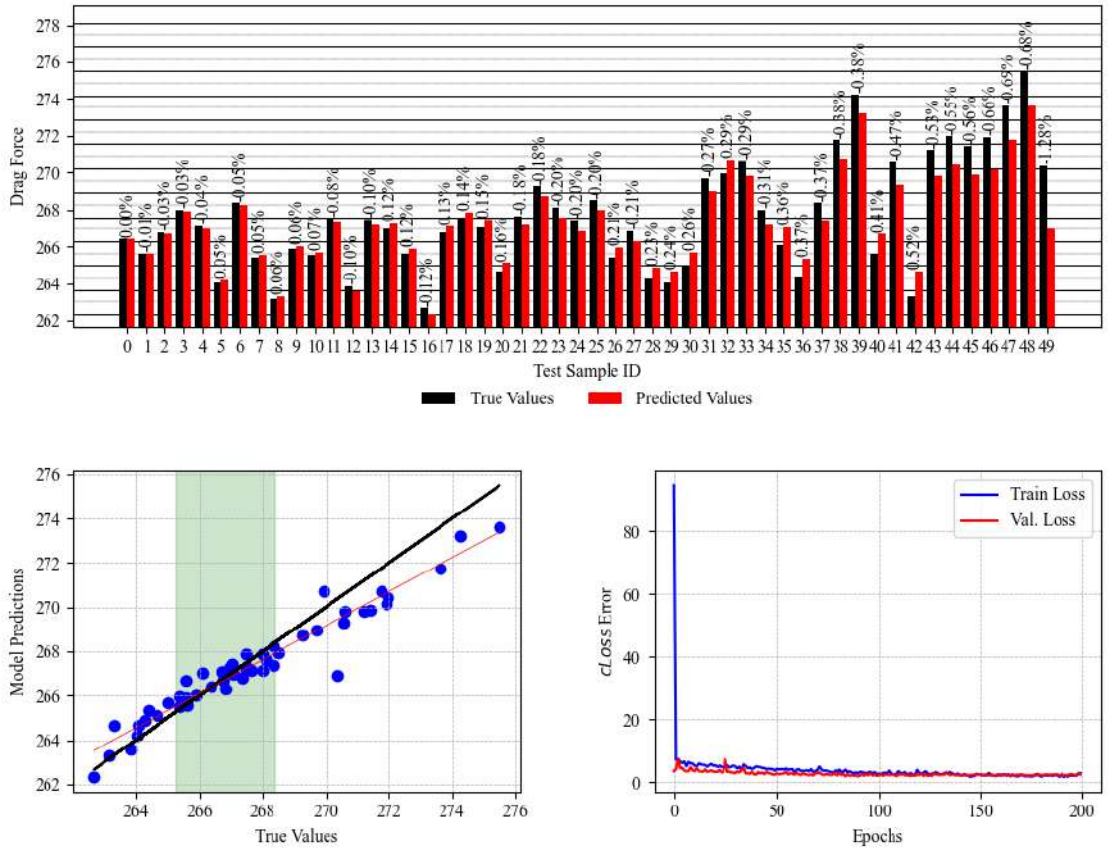



**Figure 6.9:** *Performance of the proposed SMB model with the cLoss coefficient* $\alpha = 0.05$ *(top) Bar plot comparing the target Drag Force values to the predictions of the model, displaying the Relative Error on each sample's prediction (left) Regression plot of the model, illustrating the agreement of its predictions with the target Drag Force values. The highlighted region represents the* $Q_1Q_3$ *range. (right) Convergence of cLoss (Train Loss) and MAE (Val. Loss) of the model over the training epochs.*

The coefficient is redefined to 0.05. Exploration of the design space with the same EA setup proposed a new model composition. Its performance, depicted in Fig. [6.9], presents a substantial improvement both in terms of variance agreement and isolated sample accuracy, particularly within the highlighted area. The test and prediction sets present a significant difference in variance, with $\sigma^2_{true} = 158.4796$

and $\sigma^2_{pred} = 100.1103$ respectively (8.8348 and 5.5809 in the original domain). These results highlight the sensitivity of the training process to the coefficient $\alpha$.

Further investigation within a trial-and-error process led to the definition of $\alpha = 0.08$. A computationally heavy fine-tuning process, with 800 EASY evaluations with the same general setup described earlier, proposed the model summarized in Table [6.2] as the Elite SMD implementation for the purpose of this Application. The test set has variance of 158.4796 and the prediction set 152.2746 (8.8348 and 8.4889 in the original domain). The predictions present $MARE = 0.1734\%$ ($MRE = 0.105\%$). Their regression line deviates by 0.95° from the identity line, and the distribution of error is moderately even across the range of the dataset (no bias is observed). The Elite's performance is graphically presented in Fig. [6.10] and [6.10].

| **Parameter / Metric** | **Value** | **Parameter / Metric** | **Value** |
|---|---|---|---|
| Number of CNN Layers | 5 | Number of DNN Layers | 4 |
| CNN L1 filter size (pow. of 2) | 5 | DNN L1 neurons (pow. of 2) | 7 |
| CNN L2 filter size (pow. of 2) | 6 | DNN L2 neurons (pow. of 2) | 10 |
| CNN L3 filter size (pow. of 2) | 3 | DNN L3 neurons (pow. of 2) | 5 |
| CNN L4 filter size (pow. of 2) | 4 | DNN L4 neurons (pow. of 2) | 6 |
| CNN L5 filter size (pow. of 2) | 4 | DNN L5 neurons (pow. of 2) | – |
| CNN L6 filter size (pow. of 2) | – | DNN L6 neurons (pow. of 2) | – |
| CNN L7 filter size (pow. of 2) | – | DNN L7 neurons (pow. of 2) | – |
| CNN L8 filter size (pow. of 2) | – | DNN L8 neurons (pow. of 2) | – |
| act. function CNN layers | ReLU | act. function DNN layers | ReLU |
| kernel size (constant) | (3, 3) | batch size | 40 |
| strides (constant) | (1, 1) | epochs | 300 |
| pool size (constant) | (2, 2) | *cLoss* coefficient $\alpha$ | 0.08 |
| MARE | 0.1734 % | MRE | 0.104 % |
| Test set variance $\sigma^2_{true}$ | 158.4796 | Prediction set variance $\sigma^2_{pred}$ | 152.2746 |
| $\lvert\sigma^2(y) - \sigma^2(\hat{y})\rvert$ | 6.205 | | |

**Table 6.2:** *Summary of the composition and performance of the selected SMB model after the exploration of the design space by EASY and evaluation.*

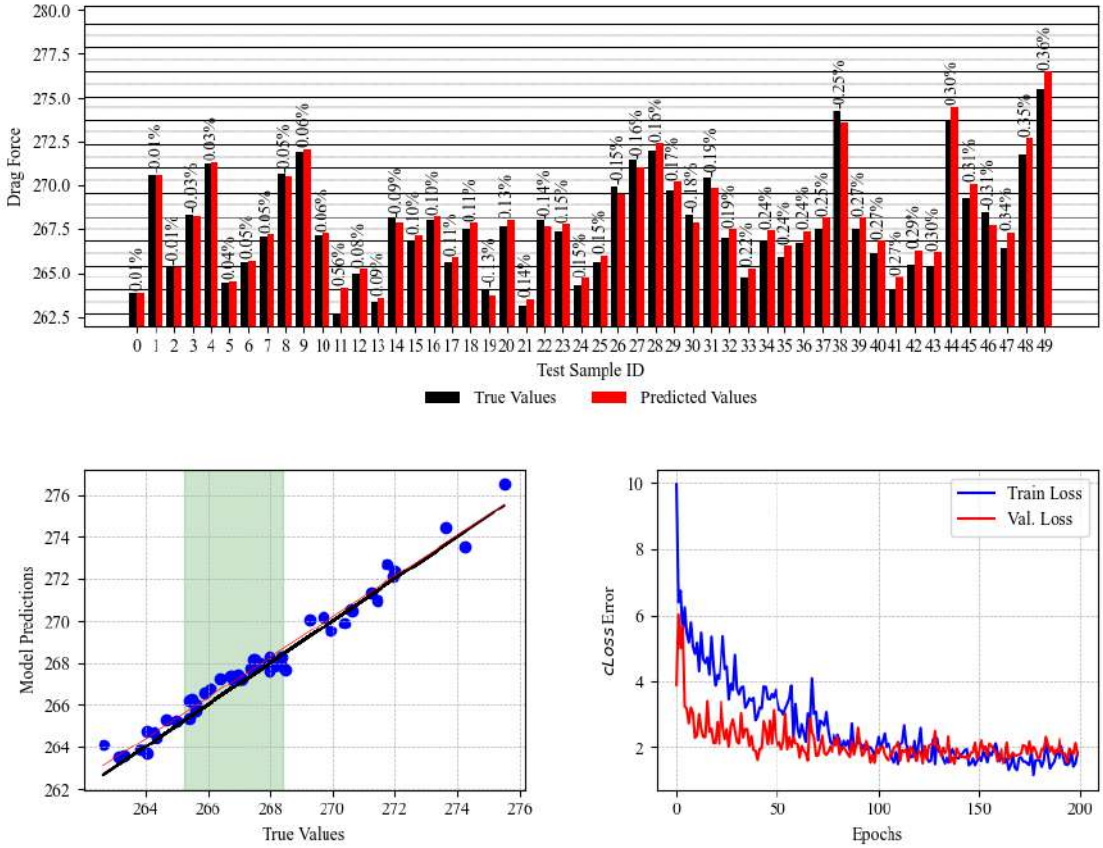



**Figure 6.10:** *Performance of the selected SMB model with the cLoss coefficient $\alpha = 0.08$. (top) Bar plot comparing the target Drag Force values to the predictions of the model, displaying the Relative Error on each sample's prediction (MARE: 0.1734%). (left) Regression plot of the model, illustrating the agreement of its predictions with the target Drag Force values. The highlighted region represents the $Q_1Q_3$ range. (right) Convergence of cLoss (Train Loss) and MAE (Val. Loss) of the model over the training epochs.*

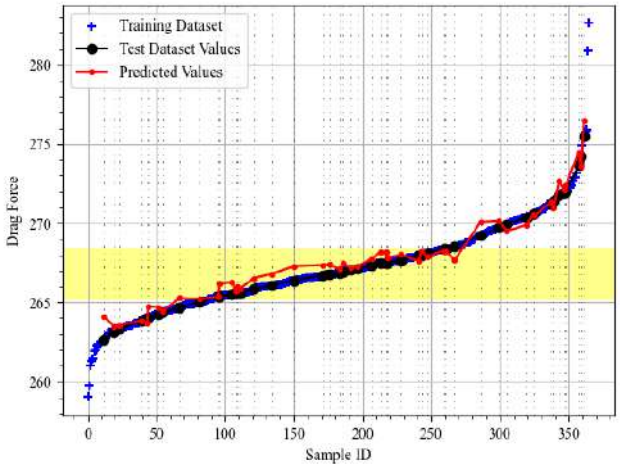



**Figure 6.11:** *The sorted working dataset. Blue crosses illustrate the training samples, while black dots illustrate the test samples and red dots the corresponding predictions of the modified SMB model. The highlighted region represents the $Q_1Q_3$ range.*

## 6.4 Overview and Conclusions

The studies conducted in Application III highlight the strength of the SMB configuration as a powerful NN architecture, capable of addressing complex aerodynamic problems, such as the prediction of the aerodynamic Drag Force of cars from their two-dimensional representations. Provided a statistically-driven pre-processing of the dataset and a generally cheap fine-tuning regarding the architectural components of the model and their parameters, as well as the training hyperparameters, the SMD models manage to identify subtle differences in the depicted geometries and decode the highly non-linear phenomena governing the problem, effectively producing predictions of great precision.

The selected SMB Elite manages to predict the Drag Force target values with accuracy across the entirety of the dataset. Employment of the *cLoss* effectively addressed the bias-variance imbalance and allowed the predictions to escape the interquartile region, introducing controlled diversity to the prediction set. Proper balance of the error distribution term and the MAE term of Eq [6.1], achieved by adjusting the $\alpha$ coefficient, allows for an overall robust performance.

However, significant computational resources are necessary to fine-tune the additional trainable parameters of the present SMB implementation, compared to that of Application II. Specifically, the exploration conducted by EASY to propose the Elite took approximately 72 hours on 2 $\times$ GeForce RTX3060.

# Chapter 7
# Conclusion

## 7.1 Overview

This Thesis demonstrates how CNNs can be integrated as local data-driven surrogate models within the early stages of automotive design, addressing the computational bottlenecks of conventional workflows that rely on 3D modeling, meshing, and flow simulations using expensive high-fidelity models. By leveraging 2D sketch-like representations of car designs (typically stored in industrial archives), the proposed methodology confirms that cooperation of SE Blocks with regularizers, and a moderately cheap fine tuning process concerning the model's architectural properties, composition and component parameters, allows for the development of case-dependent cost-efficient Convolutional models that accurately predict the designs' Drag Force values. These models can allow rapid real-time evaluation of modifications on a baseline automobile geometry, both in terms of aerodynamic criteria and aesthetics, guiding design iterations toward viable configurations while conserving computational resources for high-fidelity optimization in later stages.

The work is structured into three sequential applications of ascending complexity, each targeting distinct challenges in the development of the models.

**Application I: Airfoil Aerodynamic Prediction**

Application I concerned the development of an advanced convolutional configuration that achieves superior cost-effectiveness in 2D aerodynamic-governed phenomena. The conducted studies highlighted the importance of Squeeze-and-Excitation Blocks (SE-Blocks) both as standalone components and in conjunction with Regularization Techniques.

**Results:**

- $C_L$: MARE = 1.4271%, $C_D$: MARE = 6.8854%, Area: MARE = 4.5501%.

- Importance of SE-Blocks and Regularization to enhance CNN interpretational capabilities and decode non-linear aerodynamic phenomena
- Development of a robust backbone CNN architecture
- Restriction of the EASY exploration domain for subsequent Applications

**Application II: Single Morphing Box Automotive Design**

The backbone convolutional architecture of Application I was adapted for the prediction of properties of 3D car geometries. Three distinct approaches were examined, differing either on the shape of the input data or the architectural symmetry of the network.

- Single-Branch Model (SB): Consists of a single convolutional branch (similar to that of Application I). Accepts a single R34 view image.
- Multi-Branch (MB): Consists of multiple parallel convolutional branches, the outputs of which are concatenated and passed through a series of fully connected layers. Accepts as input multiple orthogonal views of the cars, each processed by a branch uniquely associated with it.
  - Individualized Multi-Branch (IMB): Each branch is adjusted separately during the fine-tuning process, resulting in different branch composition and an architectural asymmetry of the model.
  - Shared Multi-Branch (SMB): All convolutional branches share the same composition at all times, both in architectural components and their corresponding parameters.

The presented configurations were evaluated on the cars' Surface Areas and Drag Forces.

**Results:**

- **SMB Superiority in Drag Force Prediction:** Achieves $MARE_Tr$: 2.04%, $MAE^N_{Tr}$ : 1.13% , outperforming SB (6.43%, 3.96%) and IMB (4.16%, 2.64%).
- **SMB Superiority in Surface Area Prediction:** Achieves $MARE_Tr$: 5.73%, $MAE^N_{Tr}$ : 1.56% , outperforming SB (8.55%, 3.40%) and IMB (6.11%, 2.60%).
- **SMB Superiority in Distribution of Error:** Achieves a more evenly distributed Error across the range of the dataset.
- **Cost-Efficiency of the SMB Configuration:** The overall costs for the

development and fine-tuning of the examined models are approximated: SB =1TU, IMB ∼12TU, SMB ∼5.5TU.

A thorough comparison concerning the distributions of the error of the models' predictions justified the selection of the SMB configuration as the backbone configuration for Application III.

**Application III: Complex Automotive Geometries with Multiple Morphing Boxes**

The methodology was extended to automobiles presenting subtle modifications at their frontal, top and rear regions. Initially, the SMB model failed to produce accurate results, concentrating its predictions within the interquartile range of the dataset, indicating that the model is strongly biased. To counteract this bias-variance imbalance, a statistically driven modification was applied to the loss function of the training process, which allowed for the accurate predictions of the geometries' Drag Force values.

**Results:**

- **Highlights importance of custom Loss:** Application of *cLoss* allows for the accurate prediction of the Drag Force Values.
- $MARE = 0.1734\%, \quad MRE = 0.103\%$
- $|\sigma^2(y) - \sigma^2(\hat{y})| = 6.205$

## 7.2 The case-dependent SMB model

The developed SMB configuration, implemented in Applications II and III, is a CNN comprising the first (convolutional) half, associated with feature extraction, and the second (deep) half, associated with interpreting the extracted features. In the context of non-linear aerodynamic problems regarding complex 3D automobile designs, it manages to provide accurate precisions of the aerodynamic Drag Force values.

It is a multiple input - single output model; it accepts (herein three -in Application II- and four -in Application III-) fix-sized images and estimates the value of a single scalar parameter/metric of the examined geometry. The input images corre-

spond to different, orthogonal views of the geometry, each processed by a separate convolutional branch, uniquely associated with it. The branches' outputs are later concatenated and passed through a sequence of fully connected layers which leads to the single neuron of the output layer. It is graphically presented in Fig. [7.1].

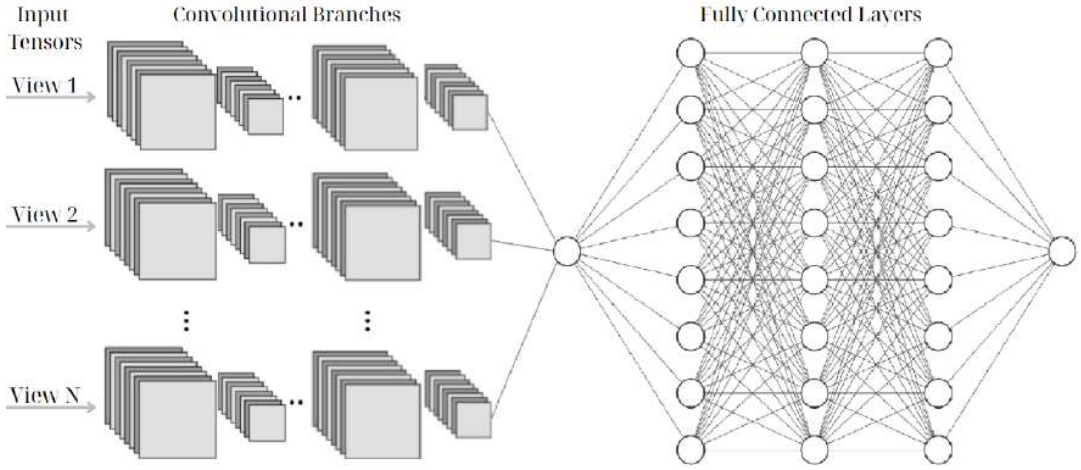



**Figure 7.1:** *Illustration of the general architecture of the SMB configuration.*

The model presents total architectural symmetry, as all convolutional branches feature the exact same sequence of components and parameters. The branches comprise convolutional layers, MaxPooling layers and, most importantly, SE Blocks, which critically enhance the networks interpretational capabilities and predictive accuracy. Regularization is applied both in the CNN and DNN parts of the network. Evolutionary tuning of the working components and application of a statistically-focused custom loss function fully unlocks this configuration's capabilities.

## 7.3 Conclusions

This Thesis suggests that CNNs can be used as local surrogate models in the preliminary phases of the car design process, aiming to bypass the costly steps of modeling, meshing and high-fidelity flow simulation in the conventional procedure. Specifically, it indicates that the developed networks can provide reliable predictions of automobiles' aerodynamic Drag Force values solely from their two-dimensional sketch-like representations. In the automotive industry, the presented methodology can allow for the development of a cheap, real-time evaluation tool for stylists, to

examine proposed designs (and design modifications) w.r.t. aerodynamic criteria in addition to aesthetics, thereby facilitating more informed design space exploration and optimization decisions during the conceptual design phase.

The applications of this Thesis collectively form a moderately cheap algorithmic process that successfully leads to the development of case-dependent CNN configurations that successfully decode the phenomena governing the working problem -in this case non-linear 3D aerodynamic applications- and produce high precision predictions. This algorithmic process can essentially be simplified into three distinct steps; the development of an effective and computationally efficient architecture for a simplified version of the target problem, the extension of this architecture to the full-scale problem, and the application of targeted modifications to ensure optimal performance in the working case.

Following this structure, Application I concerns the construction of a baseline configuration, while simultaneously aiming to implicitly exclude unimportant regions of the EA design space for the subsequent, more computationally-heavy Applications. Herein, it concerns 2D aerodynamic flows around isolated airfoils. Application II refers to the adaptive extension of the previously established backbone CNN architecture for aerodynamic problems concerning 3D geometries, specifically automobiles. In this work, three distinct configurations are examined, differing either in their input's shape or their architectural symmetry. EA-driven fine tuning and post-processing allow for the selection of a superior configuration that is best-suited for the working case, herein the SMB configuration, comprising three parallel Convolutional branches, including SE-Blocks and employing regularization techniques. Lastly, Application III highlights the importance of statistically driven pre-processing of the dataset to address potential inherent challenges. In this Thesis, a statistically-driven modification was applied to the training's Loss Function to mitigate the strong prediction bias, thereby eliminating the fundamental and common challenge of bias-variance tradeoff and produce high-precision predictions. Herein, when evaluated on a challenging set of 50 unseen automobile geometries, the SMB configuration produced predictions of their Drag Force values with a *MARE* of 0.1734%

Computational requirements for implementing these models include the cost of the dataset generation, the cost of training the model, and the cost of fine-tuning its parameters and composition. The conducted studies showcase the ability to develop robust and cost-efficient CNNs without the need for excessive sampling, model complexity (training cost) or fine-tuning. Specifically, referring to the development of the SMB Elite of Application III, the total cost is detailed (on 2 $\times$ GeForce

RTX3060) $\sim$ 185 hours for dataset generation and $\sim$ 72 hours for fine tuning (cost of training of each candidate included).

While these computational requirements appear substantial, practical implementation in industrial automotive applications typically eliminates the dataset generation phase, as manufacturers maintain extensive archives of sketches and corresponding aerodynamic performance data from conceptual design studies and existing vehicle configurations. Consequently, the primary computational investment for industrial deployment focuses on the fine-tuning phase, with cost-effectiveness determined by the frequency of the surrogate model's utilization and the associated reduction in conventional CFD simulation requirements. Indicatively, in Application III, the fifty test samples' Drag Force values were estimated at essentially zero computational cost, whereas employment of PUMA would require 25 hours in total. To balance the 72 hours of fine-tuning, utilization of the SMB for an additional 94 designs would be required.

## 7.4 Future Work Proposals

Based on the implementations presented in this Thesis, the following future work proposals are proposed:

- Firstly, additional types of networks (such as U-Nets) or network configurations can be implemented to examine whether it is possible to further improve predictive accuracy, concerning the automobiles' drag prediction from their 2D sketch-like representations. More sophisticated studies can also be conducted to replace the empirical rules employed in this Thesis' Applications II and III, although attention must be given to the balance between the computational drawbacks that this optimization would impose and the potential profits of the developed surrogate model.
- Next, the algorithmic procedure developed in this Thesis can be adjusted to be included in an automated optimization loop; the objective function being the drag force, or the downforce-to-drag ratio, and the design variables being the coordinates of the CPs that form the NURBS Moprhing Boxes enclosing regions of the original car geometry. In theory, this can allow for a shape-

optimization of the automobile within a predefined region of the design space, defined by the training dataset's bounds.

- Lastly, an interesting approach would be to extend the working dataset, by including variations of different baseline car geometries, and including a classification part of the model, identifying the baseline geometry the input sample corresponds to. Such an implementation would critically reduce the case-dependency of the presented models, and potentially allow for a greater generalization ability, which could, in theory, allow for predictions outside each baseline geometry's training dataset's bounds.

# Bibliography


[1] Asouti, V., Trompoukis, X., Kampolis, I., Giannakoglou, K.: Unsteady CFD computations using vertex–centered finite volumes for unstructured grids on Graphics Processing Units. International Journal for Numerical Methods in Fluids **67**(2), 232–246 (May 2011)

[2] Asouti, V., Trompoukis, X., Kampolis, I., Giannakoglou, K.: Unsteady cfd computations using vertex-centered finite volumes for unstructured grids on graphics processing units. International Journal for Numerical Methods in Fluids **67**(2), 232–246 (2011). https://doi.org/10.1002/fld.2342

[3] Aultman, M., Auza-Gutierrez, R., Disotell, K., Duan, L.: Effects of wheel rotation on long-period wake dynamics of the drivaer fastback model. Fluids **7**(1), 19 (2022). https://doi.org/10.3390/fluids7010019, `https://doi.org/10.3390/fluids7010019`

[4] Barthelmäs, P., Rosnitschek, T., Tremmel, S., Rieg, F.: Impact of hpc and automated cfd simulation processes on virtual product development—a case study. Applied Sciences **11**(14), 6552 (2021). https://doi.org/10.3390/app11146552, `https://doi.org/10.3390/app11146552`

[5] Belkin, M., Hsu, D., Ma, S., Mandal, S.: Reconciling modern machine learning and the bias-variance trade-off. PNAS **116**(32), 15849–15854 (2019)

[6] Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)

[7] Botchkarev, A.: Performance metrics (error measures) in machine learning regression, forecasting and prognostics: Properties and typology. arXiv preprint arXiv:1809.03006 (2018)

[8] Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Review **60**(2), 223–311 (2018)

[9] Chai, T.L., Draxler, R.R.: Root-mean-square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. Geoscientific Model Development **7**(3), 1247–1250 (2014)

[10] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. In: Proceedings of the 24th International Conference on Neural Information Processing Systems (NeurIPS). p. 257–265 (2011)

[11] Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Natural Computing Series, Springer, 2nd edn. (2003)

[12] Gain, J., Lab, C.: Enhancing spatial deformation for virtual sculpting (09 2000)

[13] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS). pp. 249–256 (2010)

[14] Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier neural networks (2011)

[15] Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)

[16] Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2 edn. (2009)

[17] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition pp. 770–778 (2016)

[18] He, X., Xue, F., Ren, X., You, Y.: Large-scale deep learning optimizations: A comprehensive survey. arXiv preprint arXiv:2111.00856 (2021)

[19] He, Y., Zhang, X., Sun, J.: Rethinking the value of batch size in stochastic gradient descent. arXiv preprint arXiv:1905.04694 (2019)

[20] Heft, A.I., Indinger, T., Adams, N.A.: Introduction of a new realistic generic car model for aerodynamic investigations. Sae technical paper 2012-01-0168, SAE International (2012). https://doi.org/10.4271/2012-01-0168

[21] Hornik, K.: Approximation capabilities of multilayer feedforward networks. Neural Networks **4**(2), 251–257 (1991)

[22] Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks **2** (1989). https://doi.org/10.1016/0893-6080(89)90020-8

[23] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7132–7141 (2018)

[24] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **42**(8), 2011–2023 (2019)

[25] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[26] Kontou, M.: The Continuous Adjoint Method with Consistent Discretization Schemes for Transitional Flows and the Use of Deep Neural Networks in Shape Optimization in Fluid Mechanics. Ph.D. thesis, National Technical University of Athens (2023)

[27] Kyriacou, S., Giannakoglou, K.X.: Evolutionary algorithm-based design optimization methods in turbomachinery – updates to easy. Technical Presentation, NTUA (2012), describes the integration of knowledge-based design, specialized operators, and PCA-driven metamodeling within EASY

[28] Lagaris, I.E., Likas, A., Fotiadis, D.I.: Artificial neural networks for solving ordinary and partial differential equations. IEEE Transactions on Neural Networks **9**(5) (1998). https://doi.org/10.1109/72.712178

[29] Larsson, R., Gustavsson, E., Wikander, J.: Shape modeling and design system using free-form deformation and nurbs. Computer-Aided Design and Applications **3**(1-4), 77–85 (2006)

[30] LeCun, Y., Bottou, L., Orr, G.B., Müller, K.R.: Efficient backprop pp. 9–50 (1998)

[31] Lee, J.H., Ko, Y.D., Yun, I.: Comparison of latin hypercube sampling and simple random sampling applied to neural network modeling of hfo2 thin film fabrication. Transactions on Electrical and Electronic Materials **7**(4), 210–214 (2006)

[32] McKay, M., Beckman, R., Conover, W.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979)

[33] Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)

[34] Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge, MA (2012)

[35] Prechelt, L.: Early stopping—but when? pp. 55–69 (1998)

[36] Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434 (2015)

[37] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, chap. 8, pp. 318–362. MIT Press (1986). https://doi.org/10.7551/mitpress/5236.003.0026

[38] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson, Hoboken, NJ, 4th edn. (2020)

[39] Schwartz, R., Stanovsky, G., Swayamdipta, S., Dodge, J., Smith, N.A.: Measure and improve robustness in nlp models: A survey. In: NAACL (2020)

[40] Science, T.D.: Performing uncertainty analysis in three steps: A hands-on guide. towardsdatascience.com/performing-uncertainty-analysis-in-three-steps-a-hands-on-guide-9110b120987e/ (2023)

[41] Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. ACM SIGGRAPH Computer Graphics **20**(4), 151–160 (1986)

[42] Shapiro, S.S., Wilk, M.B.: An analysis of variance test for normality (complete samples). Biometrika **52**(3/4), 591–611 (1965)

[43] Smith, S.L., Le, Q.V.: Don't decay the learning rate, increase the batch size. arXiv preprint arXiv:1711.00489 (2018)

[44] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 2nd edn. (2018), `http://incompleteideas.net/book/RLbook2020.pdf`

[45] Tieleman, T., Hinton, G.: Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude. `http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf` (2012), coursera: Neural Networks for Machine Learning

[46] Trompoukis, X., Tsiakas, K., Asouti, V., Giannakoglou, K.: Continuous adjoint-based shape optimization of a turbomachinery stage using a 3d volumetric parameterization. International Journal for Numerical Methods in Fluids (2023). https://doi.org/10.1002/fld.5187

[47] Wang, Q., Wu, B., Zhu, P., Li, P., Zuo, W., Hu, Q.: Eca-net: Efficient channel attention for deep convolutional neural networks. CVPR (2020)

[48] Willmott, C.J., Matsuura, K.: Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. Climate research **30**(1), 79–82 (2005)

[49] Xu, X., Chen, H., Zhang, C., Duan, Y., Wang, G.: Data-driven sensitivity analysis of the influence of geometric parameterized variables on flow fields under different design spaces. Aerospace **11**(12), 984 (2024). https://doi.org/10.3390/aerospace11120984

[50] Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. Communications of the ACM **64**(3), 107–115 (2021)

[51] Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **67**(2), 301–320 (2005)

**Εθνικό Μετσόβιο Πολυτεχνείο**
**Σχολή Μηχχανολόγων Μηχανικών**
**Τομέας Ρευστών**
**Μονάδα Παράλληλης υπολογιστικής Ρευστοδυναμικής & Βελτιστοποίησης**

# Συνελικτικά Νευρωνικά Δίκτυα ως Μεταμοντέλα στα Πρώιμα Στάδια της Διαδικασίας Σχεδιασμού Αυτοκινήτων

Διπλωματική Εργασία - Εκτενής Περίληψη στην Ελληνική

**Φαλιάκος Βασίλειος**

Επιβλέπων: Κυριάκος Χ. Γιαννάκογλου, Καθηγητής ΕΜΠ

Αθήνα, 2025

# Εισαγωγή

Στόχος της Διπλωματικής αυτής Εργασίας είναι η εφαρμογή Συνελικτικών Νευρωνικών Δίκτύων (ΣΝΔ) ως τοπικά μεταμοντέλα στα πρώιμα στάδια της διαδικασίας σχεδιασμού αυτοκινήτων, με σκοπό την υποκατάσταση κοστοβόρων βημάτων της καθιερωμένης διαδικασίας, συγκεκριμένα την μοντελοποίηση, πλεγματοποίηση και προσομοίωση (επίλυση της ροής). Κατά την προσέγγιση που παρουσιάζεται, τα ΣΝΔ δέχονται διδιάστατες αναπαραστάσεις των αυτοκινήτων σε μορφή σκίτσου, όπως συνηθίζεται να παρουσιάζονται προτεινόμενα σχέδια στην βιομηχανία, και στοχεύει στην ακριβή πρόβλεψη της αεροδυναμικής τους αντίστασης. Ένα τέτοιο εργαλείο θα επιτρέπει στους σχεδιαστές να αξιολογούν ταχέως πιθανές τροποποιήσεις σε μια αρχική γεωμετρία, και να τις δέχονται ή απορρίπτουν με γνόμωνα τόσο την αισθητική όσο και τα εκάστοτε αεροδυναμικά κριτήρια. Ως εκ τούτου, η εφαρμογή τέτοιων δικτύων δύναται να περιορίσει τον σχεδιαστικό χώρο από τα νωρίς, επιτρέπωντας την διατήρηση υπολογιστικών πόρων για πιο υποσχόμενα σχέδια σε επόμενα στάδια του σχεδιασμού. Η διπλωματική αυτή ασχολείται με την ανάπτυξη προηγμένων ΣΝΔ που, με χαμηλό υπολογιστικό κόστος, καταφέρνουν να εκτιμούν την αεροδυναμική αντίσταση των απεικονιζόμενων αυτοκινήτων με μεγάλη ακρίβεια.

# Τεχνητή Νοημοσύνη και Μηχανική Μάθηση

Η Τεχνητή Νοημοσύνη (AI) περιλαμβάνει αλγορίθμους που προσομοιώνουν ανθρώπινες γνωστικές λειτουργίες, ενώ η Μηχανική Μάθηση (ML), ως υποπεδίο της, επιτρέπει σε υπολογιστικά μοντέλα να βελτιώνονται μέσω εμπειρίας. Η εποπτευόμενη μάθηση αποτελεί το κυρίαρχο πλαίσιο για προγνωστικά μοντέλα. Τα Τεχνητά Νευρωνικά Δίκτυα (ANNs) είναι ισχυροί προσεγγιστικοί αλγόριθμοι που έχουν αποδειχθεί ιδιαίτερα αποτελεσματικοί στην υποκατάσταση πολύπλοκων υπολογιστικών διαδικασιών.

Η παρούσα εργασία αξιοποιεί (CNNs) ως τοπικά μεταμοντέλα για την απευθείας πρόβλεψη αεροδυναμικών χαρακτηριστικών αυτοκινήτων από 2D σκίτσα. Στόχος είναι η μείωση του κόστους και του χρόνου του συμβατικού σχεδιασμού, που περιλαμβάνει μοντελοποίηση, πλεγματοποίηση και ΥΡΔ ανάλυση. Μέσω αξιοποίησης υφιστάμενων βάσεων δεδομένων, η προτεινόμενη προσέγγιση ενσωματώνει τις αρχές της AI/ML στη ροή εργασιών του σχεδιασμού, ιδίως σε πρώιμα στάδια, επιτρέποντας άμεση, ποσοτική αξιολόγηση εναλλακτικών γεωμετριών σε πραγματικό χρόνο.

## Διαδικασία Σχεδιασμού Μοντέλων

Η διαδικασία που ακολουθείται για την ανάπτυξη των μοντέλων αποδομείται σε τρεις βασικές μελέτες διαφορετικών στόχων και αυξανόμενης πολυπλοκότητας: Ι) την ανάπτυξη προηγμένων CNNs για 2Δ αεροδυναμικές εφαρμογές που, στην προκειμένη, αφορούν σε αεροτομές ΙΙ) την επέκταση της προκύπτουσας αρχιτεκτονικής σε 3Δ εφαρμογές που αφορούν σε παραπλήσια αυτοκίνητα και ΙΙΙ) την αξιολόγηση του υπερισχύοντος μοντέλου σε ένα απαιτητικό σύνολο δεδομένων αυτοκινήτων.

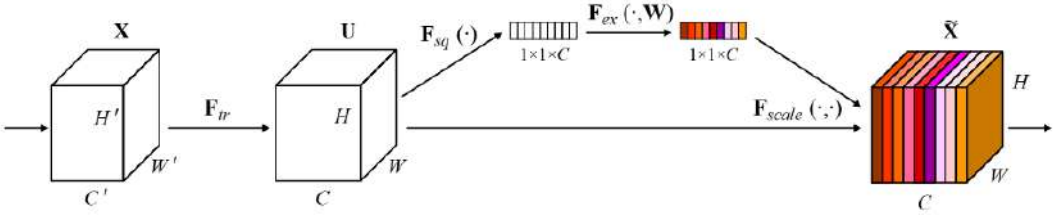
Στις τρεις μελέτες ακολουθείται με τον ίδιο, κατά βάση, τρόπο η διαδικασία που παρουσιάζεται αλγοριθμικά παρακάτω.

- Κατασκευή συνθετικού συνόλου δεδομένων για την εκπαίδευση και αξιολόγηση των δικτύων, μέσω εφαρμογής της μεθόδου FFD σε ένα αρχικό σχέδιο.
- (*Για τις μελέτες ΙΙ και ΙΙΙ:*) Αποτύπωση των γεωμετριών από διάφορες κάθετες μεταξύ τους όψεις, καθώς και την οπίσθια 3/4 όψη (R34), και επεξεργασία ώστε να αποκτήσουν μορφή που προσομοιάζει σκίτσο, για τους σκοπούς της Διπλωματικής.
- Επίλυση ροής γύρω από την εκάστοτε γεωμετρία με χρήση του λογισμικού ΥΡΔ PUMA. Τα αποτελέσματα των προσομοιώσεων απαρτίζουν το σύνολο δεδομένων εξόδου που τα μοντέλα καλούνται να προβλέψουν.

Ακόμα, γίνεται χρήση εξελικτικών αλγορίθμων, μέσω του λογισμικού EASY, για την προσαρμογή (και όχι παγκόσμια βελτιστοποίηση) αρχιτεκτονικών ιδιοτήτων και δομικών στοιχείων του εκάστοτε δικτύου με στόχο την ελαχιστοποίηση του σφάλματος των προβλέψεων.

Τέλος, παρουσιάζονται τα Μπλοκ συμπίεσης και διέγερσης (Μπλοκ Σ/Δ, SE Blocks) και η τεχνική κανονικοποίησης (Regularization), που συντέλεσαν στην ανάπτυξη εύστοχων και αποδοτικών δικτύων.

Τσ Μπλοκ Σ/Δ είναι δομικά στοιχεία που ενισχύουν την αναπαραστατική ικανότητα των ΣΝΔ, αναθέτωντας συντελεστές βάρους στα κανάλια της εξόδου ενός συνελικτικού επιπέδου, επιτρέποντας έτσι στο μοντέλο να δώσει έμφαση στα πιο σημαντικά κανάλια και να καταστείλει τα λιγότερο χρήσιμα. Τα παραπάνω γίνονται δυνατά μέσω τριών βασικών λειτουργιών: συμπίεση, διέγερση και επαναφορά σε κλίμακα. Το Σχήμα 1 απεικονίζει γραφικά την αρχή λειτουργίας των Μπλοκ Σ/Δ.




**Σχήμα 1:** *Γραφική απεικόνιση της αρχής λειτουργίας ενός Μπλοκ Σ/Δ.*

Η κανονικοποίηση είναι μια βασική τεχνική για την αποφυγή υπερπροσαρμογής και την ενίσχυση της ικανότητας γενίκευσης των μοντέλων σε νέα δεδομένα, προσθέτοντας έναν επιπρόσθετο όρο στην συνάρτηση σφάλματος που παρακολουθείται κατά την εκπαίδευση των δικτύων. Σε αυτή την Διπλωματική Εργασία, χρησιμοποιούνται οι τεχνικές L1 και L2. Η πρώτη προάγει την επιλογή χαρακτηριστικών μέσω της μηδενικής τιμής βαρών, ενώ η δεύτερη μειώνει την ευαισθησία του μοντέλου στο θόρυβο.

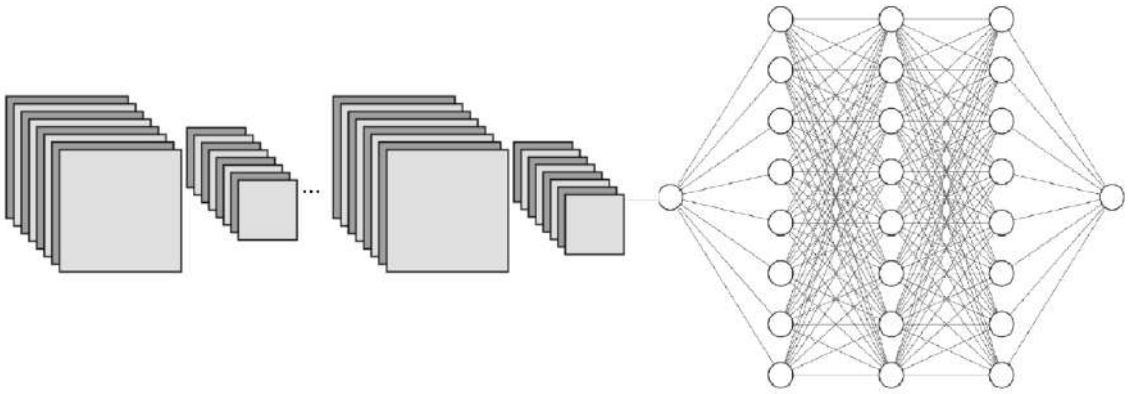
### Μελέτη Ι

Το σύνολο δεδομένων της πρώτης μελέτης προκύπτει από εφαρμογή της παραπάνω διαδικασίας στην αεροτομή NACA4318, και απεικονίζεται στο Σχήμα 2. Σκοπός είναι η πρόβλεψη του συντελεστή αεροδυναμικής άνωσης $C_L$, του συντελεστή αεροδυναμικής αντίστασης $C_D$ και της επιφάνειας των αεροτομών, σε κάποιες συνθήκες ροής.

**Σχήμα 2:** *(αριστερά) Το πλέγμα ελέγχου που περικλείει την αρχική γεωμετρία της μεμονωμένης αεροτομής NACA4318, και τα ΣΕ που το απαρτίζουν. (κέντρο) Τα περιγράμματα των διάφορων παραλλαγών της αρχικής αεροτομής που κατασκευάστηκαν. (δεξιά) Ένα τυχαίο δείγμα του συνόλου δεδομένων εισόδου, στην μορφή που θα εισέλθει στα νευρωνικά δίκτυα της παρούσας μελέτης.*

Το εργαζόμενο μοντέλο αρχικά χωρίζεται στο Συνελικτικό τμήμα (τμήμα ΣΝΔ), που ασχολείται με την αναγνώριση και απόσπαση μοτίβων στα δεδομένα εισόδου, και στο Βαθύ τμήμα (τμήμα ΒΝΔ), που ακολουθεί το Συνελικτικό και αρχολείται με την αποκωδικοποίηση και ερμηνεία των μοτίβων. Το τμήμα ΣΝΔ απαρτίζεται από αλληλουχία ζευγαριών συνελικτικών επιπέδων (Convolutional Layers) και επιπέδων μέγιστης υποδειγματοληψίας (MaxPooling layers). Το τμήμα ΒΝΔ απαρτίζεται μόνο από πυκνά

επίπεδα (dense layers) και καταλήγει σε έναν μοναδικό νευρώνα για την πρόβλεψη του εκάστοτε μονόμετρου μεγέθους. Η δομή του μοντέλου απεικονίζεται στο Σχήμα 3.



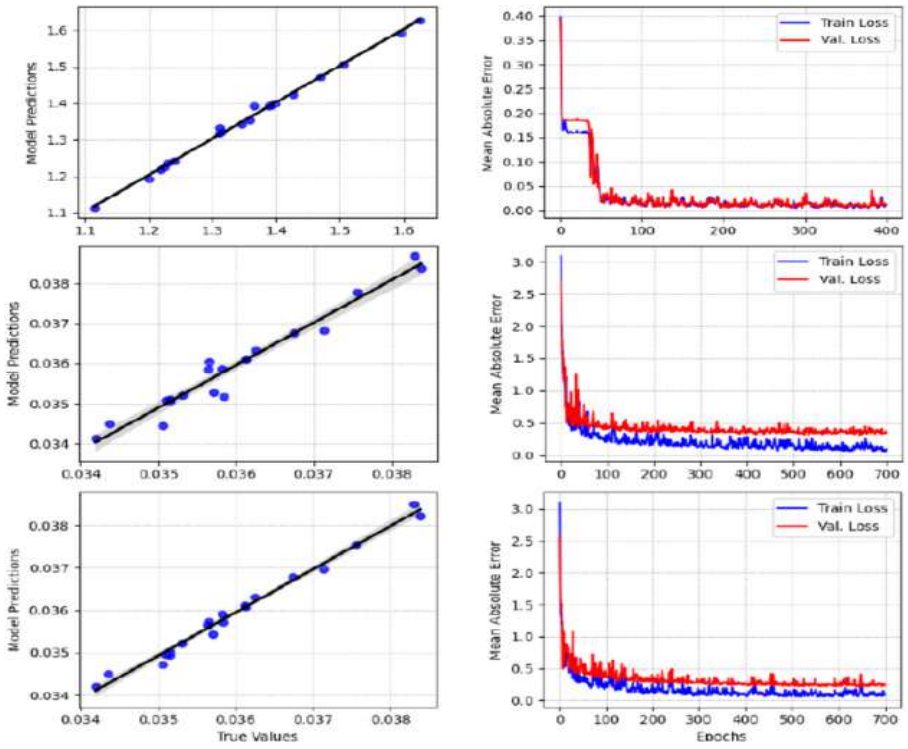
**Σχήμα 3:** *Η (απλουστευμένη) δομή του μονοκλαδικού μοντέλου που χρησιμοποιείται κατά την παρούσα μελέτη.*

Αρχικά, γίνεται χρήση του EASY για να παράξει διαμορφώσεις συμβατικών ΣΝΔ που σέβονται την παραπάνω αρχιτεκτονική, τα καλούνται να προβλέψουν και τα τρία μεγέθη ενδιαφέροντος. Γίνονται προσαρμογές στην αρχιτεκτονική και στα δομικά στοιχεία των μοντέλων, με σκοπό την ανάπτυξη προηγμένων δικτύων που αποδίδουν καλύτερα. Συγκεριμένα, προστίθενται Μπλοκ Σ/Δ ανά δύο συνελικτικά επίπεδα, και εφαρμόζονται τεχνικές κανονικοποίησης. Τα μοντέλα που σχηματίζονται για καθεμία από τις τρείς περιπτώσεις ($C_L, C_D$,, *επιφάνεια*) συγκρίνονται με τα αντίστοιχα συμβατικά.

Στην περίπτωση του συντελεστή άνωσης, οι προσαρμογές επέφεραν περίπου 30% αύξηση στην ακρίβεια των προβλέψεων με μόνο $\sim 0.52\%$ παραπάνω υπολογιστικό κόστος. Το προηγμένο δίκτυο πετυχαίνει Μέσο Απόλυτο Σχετικό Σφάλμα (ΜΑΣΣ) 0.403% ή 1.4271% στα κανονικοποιημένα δεδομένα στο εύρος 0-100. Τα αποτελέσματα του προκύπτοντος δικτύου απεικονίζονται στο Σχήμα 4.

Στην περίπτωση του συντελεστή αεροδυναμικής αντίστασης, επιτεύχθη μερική αύξηση της ακρίβειας με επιβάρυνση $\sim 0.66\%$ στο υπολογιστικό κόστος Το ΜΑΣΣ παίρνει τιμή 0.523% ή 6.8854% στο κανονικοποιημένο 0-100 πλαίσιο. Τα αποτελέσματα απεικονίζονται στο Σχήμα 4.

Τέλος, σχετικά με την πρόβλεψη της επιφάνειας, το αναπτυχθέν μοντέλο πετυχαίνει ΜΑΣΣ $= 0.3353\%$ (ή 4.5501% στα κανονικοποιημένα δεδομένα στο εύρος 0-100), όπως φαίνεται στο Σχήμα 4.



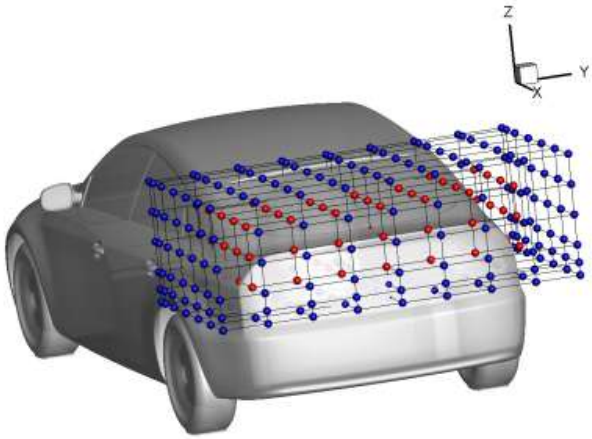

**Σχήμα 4:** *Οι αποδόσεις των ανεπτυχθέντων μοντέλων για τις περιπτώσεις (πάνω) του* $C_L$ ,(κέντρο) του $C_D$ και (κάτω) της επιφάνειας.

Η πρακτική αξία της Μελέτης Ι, πέραν της επαλήθευσης ότι τα Μπλοκ Σ/Δ και οι τεχνικές κανονικοποίησης βελτιώνουν σημαντικά την αποδοτικότητα των δικτύων, έγκειται στην αναγνώριση των σημαντικών παραμέτρων σχεδιασμού που εξερευνούν οι ΕΑ.

Συνεπώς, με το πέρας της πρώτης Μελέτης έχουμε καταφέρει αφενός να αναπτύξουμε μια αρχιτεκτονική CNNs που υπερισχύει των αντίστοιχων συμβατικών στα αεροδυναμικά προβλήματα που εξετάζονται, αφετέρου να περιορίσουμε τις διαστάσεις του χώρου εξερεύνησης των ΕΑ, μειώνοντας το κόστος της διερεύνησης προσαρμογών, ενόψει των (υπολογιστικά) απαιτητικών μελετών που ακολουθούν.

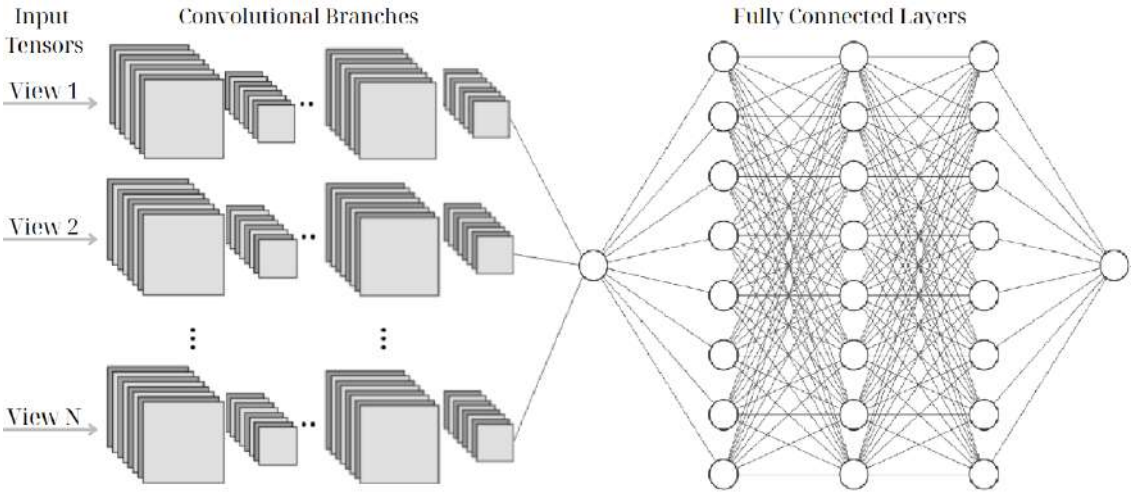
### Μελέτη ΙΙ

Η δεύτερη μελέτη εξετάζει προσαρμογές της αναπτυχθείσας αρχιτεκτνικής για χρήση σε τριδιάστατα προβλήματα που αφορούν την αεροδυναμική αντίσταση αυτοκινήτων. Ως αρχική γεωμετρία χρησιμοποιείται η fastback εκδοχή του δημόσιου μοντέλου αυτοκινήτου DrivAer, που παραμετροποιείται με πλέγμα ελέγχου στο οπίσθιο μέρος με 210 κόμβους ελέγχου, όπως φαίνεται στο Σχήμα [5].




**Σχήμα 5:** *Το πλέγμα ελέγχου που περικλείει το οπίσθιο μέρος του μοντέλου ΔριΑερ, μέσω του οποίου κατασκευάζονται 100 παραλλαγές του.*

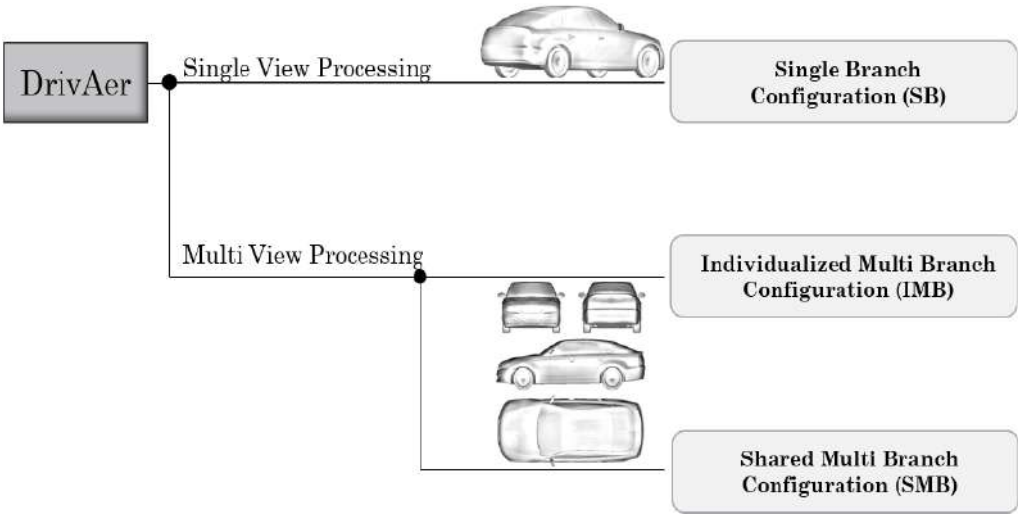
Οι τρείς εξεταζόμενες προσεγγίσεις διαφέρουν είτε ως προς το σχήμα της εισόδου των μοντέλων, είτε ως προς την συμμετρία του, και απεικονίζονται στο Σχήμα 7. Κατά την διαμόρφωση Μοναδικού (συνελικτικού) Κλάδου (Single Branch - SB), το μοντέλο δέχεται μοναδική εικόνα, που απεικονίζει το αυτοκίνητο απο την οπίσθια 3/4 όψη. Ως εκ τούτου, η αρχιτεκτονική παραμένει κατά βάση ίδια με αυτή της Μελέτης I.




**Σχήμα 6:** *Η (απλουστευμένη) δομή του πολυκλαδικού μοντέλου που εξετάζεται σε αυτή την μελέτη.*

Κατά την Πολυκλαδική προσέγγιση (Multi Branch - MB) του Σχήματος 6, το μοντέλο δέχεται εικόνες που απεικονίζουν το αυτοκίνητο από πολλαπλές, κάθετες μεταξύ τους όψεις, συγκεκριμένα εδώ την πρόσοψη, πλάγια όψη και οπίσθια όψη. Εξετάζονται δύο υποπερπτώσεις: η Εξατομικευμένη (IMB) και η Ενοποιημένη (SMB). Κατά την IMB, οι συνελικτικοί κλάδοι υπόκεινται σε διαφορετικές προσαρμογές κατά την δερεύνηση με τους ΕΑ και αποκτούν διαφορετικές διαμορφώσεις, συνεπώς το μοντέλο δεν παρουσιάζει συμμετρία. Κατά την SMB, όλοι οι κλάδοι απαρτίζονται από ακριβώς τα ίδια

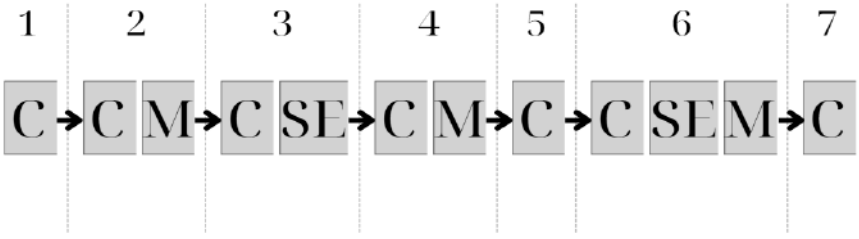
δομικά στοιχεία με τις ίδιες παραμέτρους, με το δίκτυο να είναι απόλυτα συμμετρικό.




**Σχήμα 7:** *Διάγραμμα που απεικονίζει τις τρείς εξεταζόμενες προσεγγίσεις αρχιτεκτονικής μοντέλου στην παρούσα μελέτη.*

Μετά από συνοπτική στατιστική αξιολόγηση του συνόλου δεδομένων, αυτό κανονικοποιείται στο εύρος 0-100, ώστε να αντιμετωπισθεί η πολύ μικρή διασπορά που αρχικά παρουσίαζε. Τα εξεταζόμενα μοντέλα εκπαιδεύοντια με αυτά τα (μετασχηματισμένα) δεδομένα.

Μέσα από πειραματικές διαδικασίες δοκιμών, γίνεται στους συνελικτικούς κλάδους εφαρμογή του εμπειρικού κανόνα του Σχήματος 8, ο οποίος προτείνει την προσθήκη επιπέδον μέγιστης υποδειγματοληψίας ανά δύο συνελικτικά επίπεδα, και προσθήκη Μπλοκ Σ/Δ ανά τρία.




**Σχήμα 8:** *Ο εμπειρικός κανόνας που εφαρμόσθηκε στους συνελικτικούς κλάδους, για την εισαγωγή επιπέδων μέγιστης υποδειγματοληψίας και Μπλοκ Σ/Δ.*

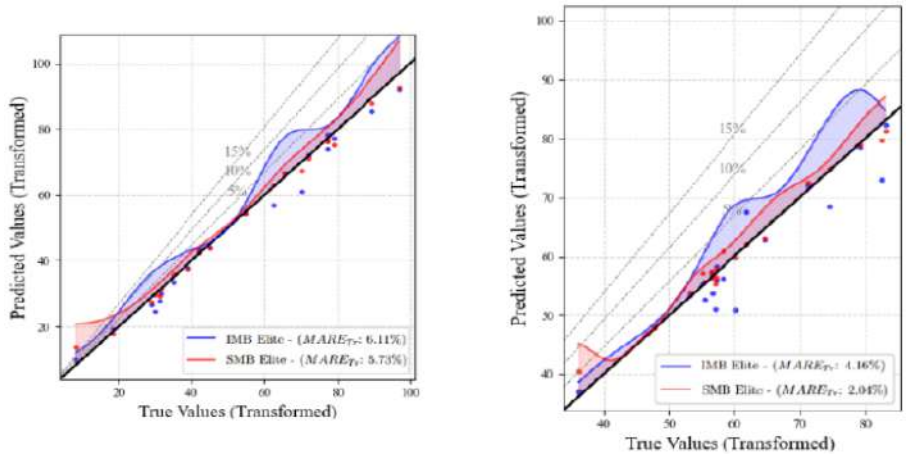
Μετά από μελέτες και προσαρμογές μέσω του λογισμικού EASY, προκύπτουν τα ακόλουθα αποτελέσματα (στα κανονικοποιημένα δεδομένα στο εύρος 0-100).

- **Μονοκλαδικό Μοντέλο:** 20 διάστατος παραμετρικός χώρος

- Πρόβλεψη Επιφάνειας: ΜΑΣΣ = 8.55%, Ανομοιόμορφη κατανομή σφάλματος
- Πρόβλεψη Αεροδυναμικής Αντίστασης: ΜΑΣΣ = 6.43%, Έντονη απόκλιση προβλέψεων από τους στόχους σε όλο το εύρος του συνόλου δεδομένων

- **Εξατομικευμένο Πολυκλαδικό Μοντέλο:** 32 διάστατος παραμετρικός χώρος
  - Πρόβλεψη Επιφάνειας: ΜΑΣΣ = 6.11%, Σχετικά ομοιόμορφη κατανομή σφάλματος
  - Πρόβλεψη Αεροδυναμικής Αντίστασης: ΜΑΣΣ = 4.16%, Ανομοιόμορφη κατανομή σφάλματος
- **Ενοποιημένο Πολυκλαδικό Μοντέλο:** 20 διάστατος παραμετρικός χώρος
  - Πρόβλεψη Επιφάνειας: ΜΑΣΣ = 5.73%, Πολύ ομοιόμορφη κατανομή σφάλματος
  - Πρόβλεψη Αεροδυναμικής Αντίστασης: ΜΑΣΣ = 2.04%, Ομοιόμορφη κατανομή σφάλματος

Οι μελέτες δείχνουν την αναμφισβήτητη υπεροχή της Πολυκλαδικής Διαμόρφωσης, συγκεκριμένα της SMB, τόσο ως προς την ακρίβεια των αποτελεσμάτων, όσο και την απαίτηση υπολογιστικών πόρων για την διερεύνηση από τον EASY, λόγω των διαστάσεων του παραμετρικού χώρου. Το Σχήμα 9 απεικονίζει την κατανομή του ΜΑΣΣ των δύο πολυκλαδικών διαμορφώσεων στα κανονικοποιημένα δεδομένα.




**Σχήμα 9:** *Οι κατανομές του ΜΑΣΣ των δύο πολυκλαδικών μοντέλων στα κανονικοποιημένα δεδομένα, στην περίπτωση (αριστερά) της πρόβλεψης επιφάνειας και (δεξιά) της πρόβλεψης της αεροδυναμικής αντίστασης.*

### Μελέτη III

Η τρίτη και τελευταία μελέτη εξετάζει την αποδοτικότητα της Κοινοποιημένης Πολυκλαδικής διαμόρφωσης σε ένα απαιτητικό σύνολο αυτοκινήτων. Εδώ, τα αυτοκίνητα παραμορφώνονται μέσω τριών πλεγμάτων ελέγχου, τοποθετημένα στο εμπρόσθιο, κεντικό και οπίσθιο τμήμα τους, και οι παραμορφώσεις είναι πιο διακριτικές από ότι στην προηγούμενη μελέτη. Δημιουργούνται $N_{db,III} = 366$ παραλλαγές του DrivAer στο σύνολο. Στατιστική μελέτη δείχνει ότι η κατανομή των δεδομένων εξόδου είναι πολύ πυκνή και παρουσιάζει εξαιρετικά μικρή διασπορά. Κανονικοποίηση τους στο εύρος 0-100 βελτιώνει εν μέρει την κατανομή, αλλά δεν αντιμετωπίζει τις περισσότερες ιδιομορφίες της.

Το επικρατέστερο μοντέλο της προηγηθείσας μελέτης (με τις ίδιες παραμέτρους) αδυνατεί να παράξει ακριβείς προβλέψεις. Αρχικά εφαρμόζονται τροποποιήσεις στο μοντέλο, με σκοπό την αύξηση της πολυπλοκότητας, και ακολούθως, ευστοχίας του, θυσιάζοντας υπολογιστική αποδοτικότητα. Το ανανεωμένο μοντέλο, παρότι αναγνωρίζει διαφορές στα απεικονιζόμενα αυτοκίνητα, αδυνατεί να τις ερμηνεύσει, και συγκεντρώνει τις προβλέψεις του στο εύρος ενδοτεταρτημορίου του συνόλου δεδομένων, που περιέχει το 50% των δεδομένων.

Το φαινόμενο αυτό αναδεικνύει μια θεμελιώδη πρόκληση στην μηχανική μάθηση που ονομάζεται αντιπαράθεση προκατάληψης-διασποράς (bias-variance tradeoff): όταν τα δεδομένα εξόδου παρουσιάζουν περιορισμένη διακύμανση, τα νευρωνικά δίκτυα αναπτύσσουν μια προκατάληψη προς την μέση τιμή της κατανομής, ελαχιστοποιώντας έτσι το σφάλμα.

Για να αντιμετωπισθεί το παραπάνω, προς αποφυγή παιρεταίρω αύξησης της πολυπλοκότητας, που θα καθιστούσε την ανάπτυξη και χρήση του δικτύου ως μεταμοντέλο υπολογιστικά ασύμφορη, επιλέγεται μια διαφορετική προσέγγιση, κατά την οποία τροποποιείται η συνάρτηση σφάλματος που παρακολουθείται κατά τη διαδικασία εκπαίδευσης. Συγκεκριμένα, αυτή επαναπροσδιορίζεται ως εξής.

$$cLoss = \alpha \cdot \left|\sigma^2(y) - \sigma^2(\hat{y})\right| + (1-\alpha) \cdot MAE(y, \hat{y})$$

$$= a \cdot \left| \frac{1}{N}\sum_{j=1}^{N}(y_j - \mu_y)^2 - \frac{1}{N}\sum_{j=1}^{N}(\hat{y}_j - \mu_{\hat{y}})^2 \right| + (1-a) \cdot \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

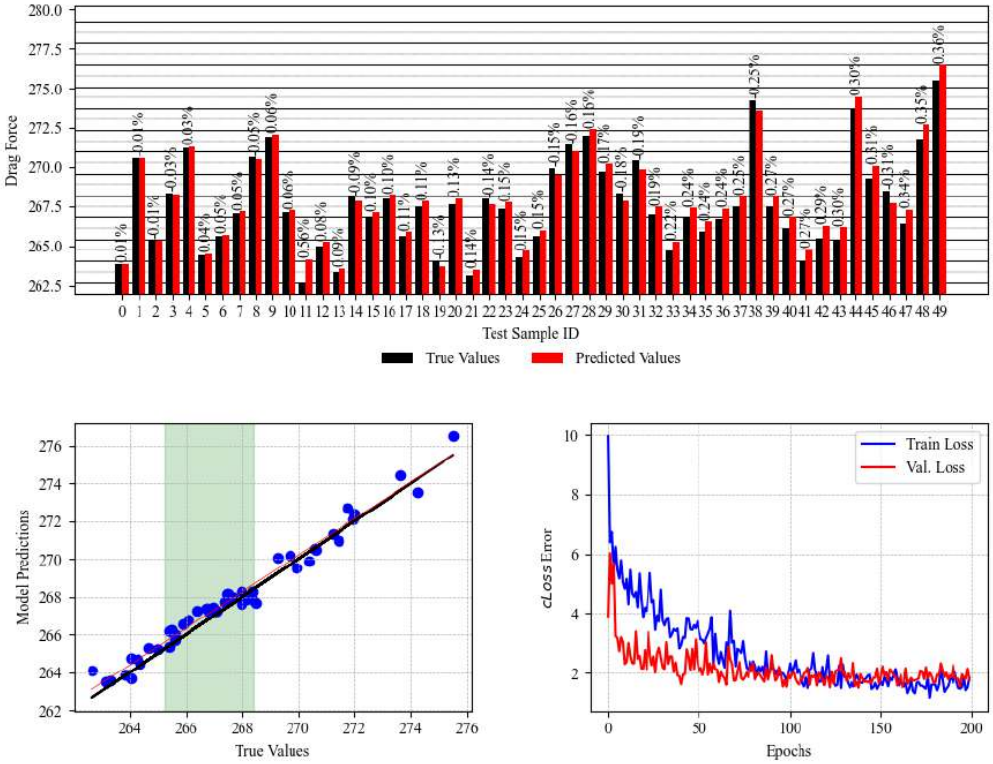
Στην ουσία, προστίθεται ένας όρος που ποσοτικοποιεί την απόκλιση της διασποράς των προβλέψεων με αυτή των δεδομένων στόχου. Η ισορροπία μεταξύ του όρου του

ΜΑΣ και της απόλυτης διαφοράς των διασπορών ρυθμίζεται μέσω του συντελεστή $\alpha$.

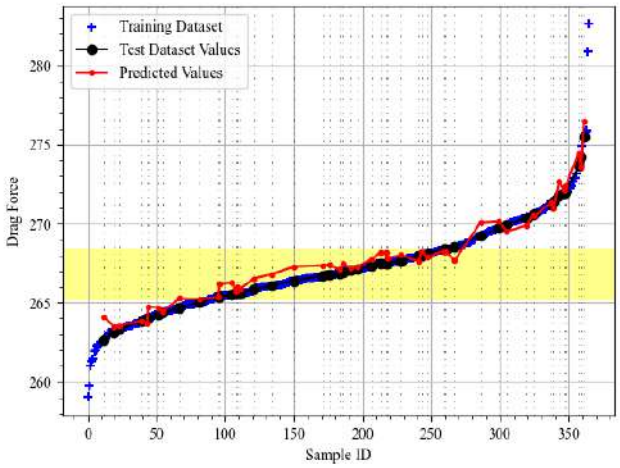
Η τροποποίηση αυτή επιφέρει μια ελεγχόμενη διασπορά στις προβλέψεις του μοντέλου. Μετά από ρύθμιση του συντελεστή $\alpha$ και προσαρμογης των δομικών στοιχείων μεσω ΕΑ, προκύπτει το Ενοποιημένο Πολυκλαδικό μοντέλο του Πίνακα 1.1, που καταφέρνει να προβλέψει την αεροδυναμική αντίσταση με εξαιρετική ακρίβεια, όπως φαίνεται στα Σχήματα 10 και 11.

| **Parameter / Metric** | **Value** | **Parameter / Metric** | **Value** |
|---|---|---|---|
| Number of CNN Layers | 5 | Number of DNN Layers | 4 |
| CNN L1 filter size (pow. of 2) | 5 | DNN L1 neurons (pow. of 2) | 7 |
| CNN L2 filter size (pow. of 2) | 6 | DNN L2 neurons (pow. of 2) | 10 |
| CNN L3 filter size (pow. of 2) | 3 | DNN L3 neurons (pow. of 2) | 5 |
| CNN L4 filter size (pow. of 2) | 4 | DNN L4 neurons (pow. of 2) | 6 |
| CNN L5 filter size (pow. of 2) | 4 | DNN L5 neurons (pow. of 2) | – |
| CNN L6 filter size (pow. of 2) | – | DNN L6 neurons (pow. of 2) | – |
| CNN L7 filter size (pow. of 2) | – | DNN L7 neurons (pow. of 2) | – |
| CNN L8 filter size (pow. of 2) | – | DNN L8 neurons (pow. of 2) | – |
| act. function CNN layers | ReLU | act. function DNN layers | ReLU |
| kernel size (constant) | (3, 3) | batch size | 32 |
| strides (constant) | (1, 1) | epochs | 300 |
| pool size (constant) | (2, 2) | *cLoss* coefficient $\alpha$ | 0.08 |
| MARE | 0.1734 % | MRE | 0.104 % |
| Test set variance $\sigma^2_{true}$ | 158.4796 | Prediction set variance $\sigma^2_{pred}$ | 152.2746 |
| $\lvert\sigma^2(y) - \sigma^2(\hat{y})\rvert$ | 6.205 | | |

**Table 1.1:** *Τα χαρακτηριστικά του τελικού Ενοποιημένου Πολυκλαδικού Μοντέλου, μετά απο εξερεύνηση του χώρου σχεδιασμού από το λογισμικό EASY.*




**Σχήμα 10:** *Επίδοση του επιλεγμένου μοντέλου με τον συντελεστή $\alpha = 0.08$. (πάνω) Ραβδογράφημα που συγκρίνει τις τιμές στόχου της αεροδυναμικής αντίστασης με τις προβλέψεις του μοντέλου, εμφανίζοντας το Σχετικό Σφάλμα σε κάθε πρόβλεψη (MARE: 0,1734%). (αριστερά) Διάγραμμα παλινδρόμησης του μοντέλου, που απεικονίζει τη συμφωνία των προβλέψεών του με τις τιμές στόχου της αεροδυναμικής αντίστασης. Η τονισμένη περιοχή αντιπροσωπεύει το εύρος $Q_1Q_3$. (δεξιά) Σύγκλιση του cLoss (Train Loss) και του MAE (Val. Loss) του μοντέλου κατά τη διάρκεια των εποχών εκπαίδευσης.*




**Σχήμα 11:** *Το ταξινομημένο σύνολο δεδομένων. Οι μπλε σταυροί απεικονίζουν τα δείγματα εκπαίδευσης, ενώ οι μαύρες κουκκίδες τα δείγματα αξιολόγησης και οι κόκκινες κουκκίδες τις αντίστοιχες προβλέψεις του επιλεγμένου μοντέλου. Η τονισμένη περιοχή αντιπροσωπεύει το εύρος $Q_1Q_3$.*

## Συμπεράσματα

Τα αποτελέσματα από τις προηγηθείσες μελέτες αποδεικνύουν ότι, με κατάλληλες προσαρμογές και σχετικά φθηνή διερεύνηση μέσω EA, είναι δυνατή η κατασκευή CNNs που να προβλέπουν με ακρίβεια την αεροδυναμική αντίσταση αυτοκινήτων κατευθείαν από διδιάστατες αναπαραστάσεις τους σε μορφή σκίτσων. Είναι, επομένως, δυνατή η εκπαίδευση τους με δεδομένα που ποϋπάρχουν σε εταιρίες της αυτοκινητοβιομηχανίας, και η χρήση τους ως τοπικά μεταμοντέλα στην διαδικασία σχεδιασμού αυτοκινήτων, υποκαθιστώντας κοστοβόρα βήματα της συμβατικής διαδικασίας που ακολουθείται. Εφαρμογή τους δύναται να επιφέρει τεράστια πτώση στο υπολογιστικό κόστος και ασήμαντες (για τα πρώιμα στάδια) επιπτώσεις στην ακρίβεια των αποτελεσμάτων.