POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Master's Degree Thesis

The Explainable AI lens: revealing the turbulence dynamics

Supervisors

Prof. Gioacchino CAFIERO

Prof. Gaetano IUSO

Enrico AMICO

Candidate

Lorenzo MATTEUCCI

October 2024

Abstract

This study explores the potential of Explainable Artificial Intelligence (XAI) algorithms, with a particular focus on the SHapley Additive Explanations algorithm (SHAP), for providing novel insights into the complex dynamics of turbulence. Turbulence is a highly non-linear and chaotic phenomenon, which presents significant challenges in terms of prediction and modelling. It is often the case that traditional methods are unable to adequately capture the intricate details of turbulent flows, thereby necessitating the exploration of advanced computational techniques.

In this study, we utilise convolutional neural networks (CNNs) in conjunction with the SHAP algorithm to analyse and predict turbulent flow patterns in an axisymmetric jet. The CNNs are trained on experimental data with the objective of identifying and learning the underlying patterns of turbulence. The objective of integrating the SHAP algorithm is to enhance the interpretability of the neural network outputs, thereby facilitating a more comprehensive understanding of the physical phenomena that govern turbulence.

The findings of the study demonstrate that the SHAP algorithm is an effective method for identifying the most influential features that contribute to the neural network's predictions. This provides valuable insights into the mechanics of turbulent flows. This enhanced interpretability not only facilitates the validation of the model's predictions but also contributes to the field of fluid dynamics by uncovering so far obscured aspects of turbulence.

Acknowledgements

I would like to begin by acknowledging my supervisors, Prof. Gioacchino Cafiero and Prof. Gaetano Iuso for giving me the opportunity to work in my field of interest in such a stimulating environment.

I would like to thank Ph.D. student Enrico Amico, who supported me throughout the project and encouraged me to achieve the best results. I am grateful to Enrico for sharing his passion for scientific research with me. It was an honour to work side by side with him.

I would also like to acknowledge my family and friends who supported me and believed in me throughout this important goal of my life.

Table of Contents

List of Tables

List of Figures

Acronyms

AI

Artificial intelligence

ANN

Artificial neural network

cGAN

Conditional Generative Adversarial Network

CNN

Convolutional neural network

GAN

Generative Adversarial Network

GDM

Gradient descent with momentum

GSD

Stochastic gradient descent

LIME

Local interpretable model/agnostic explanations

LSTM

Long short-term memeory

NN

Neural network

PDF

Probability density fucntion

PIV

Particle image velocimetry

RMS

Root mean square

RMSprop

Root mean square propagation

SHAP

SHapley Additive exPlanations

XAI

Explainable artificial intelligence

Chapter 1

Introduction

The objective of this study is to invetigate the potential of algorithms for Explainable Artificial Intelligence (XAI), with a particular focus on the SHAP algorithm, to provide novel insights into turbulence dynamics.

The turbulence dynamics is an open problem of very large scientific interest this is because of the complexity of fluid flow. Turbulence is highly non-linear phenomenon. This means that small changes in the initial conditions can lead to drastically different outcomes, making it difficult to predict and model accurately. Turbulent flows involve a wide range of spatial and temporal scales. Capturing all these scales simultaneously in a single simulation is computationally expensive and often impractical. The Navier-Stokes equations are the fundamental governing equations for fluid flow, including turbulence. However, solving these equations analytically for turbulent flows is not possible.

The complexity of turbulence necessitates the use of advanced mathematical tools, computational methods, and experimental techniques, so that new approach are costantly being developed to tackle turbulence.

Despite the challenges, turbulence remains a fascinating and intellectually stimulating field for mathematicians, physicists, engineers, and other scientists and unraveling its mysteries promises significant advances in various scientific and engineering disciplines.

A classical example of turbulent flow is the axisymmetric jet that is a fluid flow developing in the absence of walls. The absence of a wall is the defining characteristic of the jet, which is inherently turbulent. This is in contrast to other canonical flows developing in the presence of walls, such as those in a channel or on a flat sheet, which may not be turbulent.

In recent years, many resercher are using the machine learning for investigate aerodyanmics problems because of their advantages respect to the traditional CFD. In particular many authors propose convolutional neural networks (CNN) for the turbulent flow prediction.

These neural networks are *image-to-image translation networks*, which means that the neural network takes images as input and generates an output image. In such instances, the image may be represented by the velocity field of fluid motion at a specific instant, t, while the output image may contain the velocity field at the subsequent instant, $t + \Delta t$.

K. Portal-Porras, U. Fernandez-Gamiz et. al. [1] proposed the development of a neural networks to obtain the flow field in the wake of a 2D cylinder. They have developed a CNN with U-Net architecture that take as input the free stream velocity, the lift coefficient at time t and at time $t - 1$ and a function called $SDF(x, y)$ that is equal to zero where in the contour of the cylinder, is less than zero inside the cylinder geometry an greater than zero outside the cylinder. At the output of the network we have the velocity field u, v at the istant t and the pressure filed at the same istant. The results indicated that the training phase had been highly effective, with an accuracy level of approximately 5% error on the lift coefficient and Strouhal number compared to the traditional CFD with a reduction of 192.4 times the required computational time.

A significant shortcoming of these neural networks is their lack of interpretability, which hinders the ability to comprehend the underlying rationale behind their outputs. In other words the problem is that whan we give an input at a pretrained model it will give us an output but the model is a black box so we can't se and we can't understand in any way why the model generate that specific output. This phenomenon is particularly evident in the context of increasingly complex and deep neural networks, which render them unreliable. This is because it is often not possible to utilise the output of such models in applications that require a high degree of reliability, precisely because it is not possible to estimate the level of reliability of the output.

Consequently, recent research in the field of Artificial Intelligence (AI) has focused on addressing these issues. The field of explainable artificial intelligence (XAI) has emerged as a key area of study in this regard, aiming to develop new methods for enhancing the interpretability of AI models.

Following the training of a convolutional neural network for the prediction of a turbulent flow, the objective is to apply an XAI algorithm to the trained model in order to ascertain whether these algorithms can provide insights into the underlying physics of fluid dynamics.

In this instance, the Kernel-SHAP algorithm will be employed, which is a game-theoretic algorithm formulated by Shapley.[2] This algorithm can be used to calculate the distribution of winnings among a coalition of players based on their contributions to the game. The objective of our study is to identify a turbulent structure and to understand its impact on the prediction of turbulent flow performed by the implemented neural network.

Applying this algorithm to the CNN that predicts the flow of an axisymmetric

jet means finding turbulent structures in the predicted flow and calculating the relevance of these structures. Subsequently, the objective is to conduct a statistical analysis of these structures, encompassing both the physical attributes of the turbulent structures and their Shapley values. This analysis aims to ascertain whether insights can be gleaned about turbulence dynamics through such an approach.

Chapter 2

How a CNN work

As previously indicated in the introduction, convolutional neural networks (CNNs) are employed to predict turbulent flow. CNNs represent a specific type of neural network designed for image classification. Subsequently, they have been utilized to generate images from another image, a process known as image-to-image translation.

Artificial neural networks are computational models inspired by the way the human brain works. The brain is made up of billions of neurons, nerve cells that transmit electrical signals to each other through connections called synapses. Each neuron receives inputs from other neurons and produces an output based on an activation function. The brain is able to learn from experience and perform cognitive tasks because of synaptic plasticity, or the ability to change the strength of connections based on signals received.

Artificial neural networks consist of computational units called nodes or artificial neurons, which are connected to each other by synaptic weights. Each node receives inputs from other nodes or external sources and produces an output based on an activation function. Artificial neural networks can learn from data and perform classification, regression, generation, etc. tasks by updating synaptic weights based on an error function.

Artificial neural networks can be organised in different ways, depending on the number of layers, the type of connections and the activation function. Some of the most common architectures are feedforward networks, recurrent networks, convolutional networks and generative adversarial networks. Each of these architectures has specific characteristics and applications in the field of artificial intelligence.

CNNs are analogous to traditional Artificial Neural Networks (ANNs): each neuron receives an input and performs a mathematical operation. The main difference between ANNs and CNNs is that the latter are used to encode images because, due to their structure, classical ANNs require higher computational complexity to compute image data, making them very expensive to use.

2.1 A look at the human brain

The nervous system of humans can be represented by a block diagram consisting of three main subsystems: receptors, neural network, and effectors. The receptors receive stimuli from the external world and convert them into electrical impulses. These electrical signals are then transmitted to the neural network, which processes the information from the receptors and produces another electrical signal. This signal is then transmitted to the effectors, which produce a response in our body.

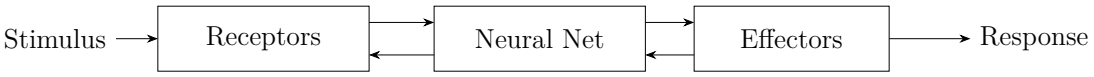

Figure 2.1: Block diagram representation of human nervous system [3]

The neural network of our nervous system is represented by the brain, that could be described as composed from neurons that are the structural components of the brain and the synapses that are the interconnection between neurons. Traditionally it's used to describe the neural organization assuming that synapses is a simple connection that can impose the excitation or inhibition state of neurons, but not both states.

The brain develops by adapting to environmental stimuli through the plasticity of synaptic connections. In the adult brain plasticity may be accounted for by two mechanism:

- creation of new synaptic connection between neurons;
- modification of existing synapses.

Axons (the transmission lines) and dendrites (receptive zones) can be distinguished because of their morphological differences: axons have a smoother surface and are longer than dendrites, which have an irregular surface. Neurons can vary in shape and size depending on their location in the brain.

As previously discussed, neurons encode their outputs through a series of brief electrical pulses known as *action potentials* or *spikes*. These originate near the cell body of neurons and propagate across individual neurons with a constant velocity and amplitude.

Axons is very long and thin and they are characterized by an electrical resistance and capacitance, so the axon can be modeled as a RC transmission line. The analysis of this propagation mechanism reveals that when a voltage is applied at one end of the axon, it decay exponentially with the distance.

As will be demonstrated, artificial neural networks are accurately represented by emulating the structural characteristics of the human brain. The receptors are conceptualised as input layers, which are responsible for receiving inputs and

transforming them into computable signals for neurons. The neural net is defined as the collective of all neurons that are interconnected in a manner analogous to that observed in the artificial neural network. Finally, the effectors, which are responsible for generating a response in the body, constitute the output layer of the artificial neural network.

Additionally, the training process, which in the context of artificial neural networks refers to the identification of an optimal neuron weight value to achieve a desired output, is analogous to the human brain's analogous process. As previously discussed, during the learning of a new concept or process, the plasticity of synapses enables their modification, thus enabling the acquisition of new knowledge or skills.

2.2 Artificial neural networks

As seen in the introduction, artificial neural networks are computational objects inspired by the human brain, and their development began in the mid-20th century. In 1943, W.S. McCulloch and Walter Pitts proposed a single neuron model capable of computing simple Boolean functions.

Later, in 1958, the mathematician John von Neumann, in his work "The Computer and the Brain" [4], examined the models proposed up to that point and highlighted their lack of accuracy in performing complex operations.

In the same year, the psychologist Frank Roselblatt proposed the implementation of a neural network consisting of a single neuron, together with input and output nodes, which he called *perceptrons*. The neuron receives input from the input nodes via axons containing weights. An activation function characterises the neuron that processes the input and returns an output:

$$y = f\left(b + \sum_{i=1}^{n} w_i \cdot x_i\right) \tag{2.1}$$

where:

- f is the activation function;
- b is the bias, that is an external parameter of artificial neuron;
- w_i is the i-th weigth of the neural network;
- n is the total number of the input.

The activation function defines the output of the neuron. There are several types of activation function, the most common are:

- Rectified Linear Unit: $ReLU(x) = max(0, x)$;

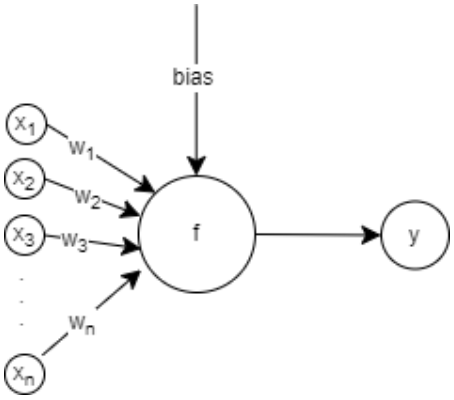

Figure 2.2: Perceptron scheme

- Hyperbolic Tangent: $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$;
- Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$.

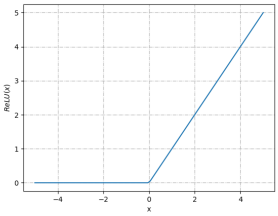

Figure 2.3: Rectified Linear Unit activation function

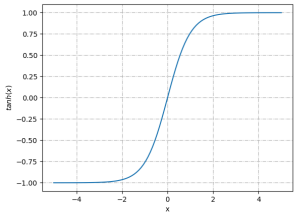

Figure 2.4: Hyperbolic tangent activation function

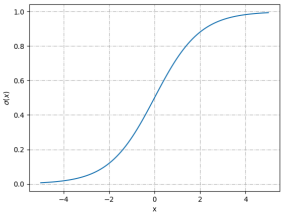

Figure 2.5: Sigmoid activation function

2.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of ANN where the input layer is an image, so the architecture of this NN has been specified to process images.

If we want the NN to deal with normal images, we should use hidden layers with their neurons organised in three dimensions, the first two dimensions being the height and width of the image and the last and width of the image, and the last dimension is the number of layers in which the colours are discretized. For example a full HD image with 1920x1080 pixels will be a tensor of 1920x1080x3 elements. These networks can work also with "fake image", that means matrix with

numbers that dont't represent properly an image, but it don't know becaue what is important for the network is the form of the input. Also exists CNNs that can take as input tensor of four dimensios, that is a simple extension of the concept of CNNs for images, but it is very useful in fluid dynamic applications because we often deal with three-dimensional motion field, overall with turbulent flow that are always 3D.

One of the first application of CNNs was for the image classification, that means the neural network take as input an image and elaborate it to have as output a descriprion of what the image represent. For example Y. LeCun, B. Boser et. al. in 1989 [5] proposed a CNN that was able to recognise handwritten characters. This was then used to automate the reading of bank cheques.was used to automatize the reading of bank cheques.

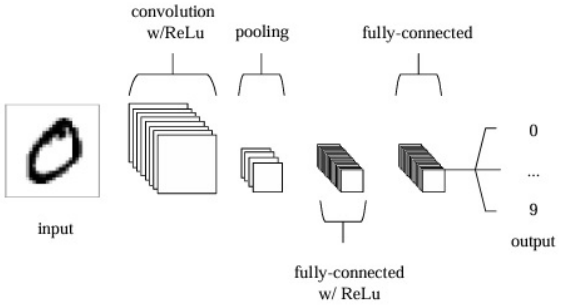

Figure 2.6: Simple scheme of a CNN architecture [6]

Usually, CNNs a re composed of four types of layer: the input layer, convolutional layers, pooling layers, and fully connected layers, as shown in Figure 2.6 [6].

The input layer is responsible for receiving the input data and transforming it into a matrix of numbers that the computer can process. This matrix is then transferred to the convolutional layers.

Convolutional layers apply filters, called kernel, at the input data. Convolutional is a mathematical operation defined as:

$$(f \star g)(t) = \int_{-\infty}^{+\infty} f(t)g(t-\tau)d\tau \tag{2.2}$$

where f represent the input function and g the kernel function. For each convolutional layer, an activation function is applied to the output. Convolutional layers have many parameters to set such as the kernel dimension that difine the number o learnable parameter of the layer, or the padding that is uswful in order to regulate the dimensions of the output and so one. The aforementioned settings have the potential to impact the efficiency of the network, with the specific choice of settings depending on the nature of the work to be performed by the network.

Following the convolutional layers, there are pooling layers, which can rapidly reduce the output dimension, thereby reducing the computational cost required to obtain the desired output size. The final layers are fully connected layers, which link all the artificial neurons of the preceding layers in order to obtain the final output of the NN.

This is the standard architecture of convolutional neural networks. However, depending on the purpose of the network's operation, it can be modified and integrated with the desired output. As seen in chapter 1 we need a CNN that takes as input an instantaneous velocity field and elaborates it to predict the velocity field at the subsequent instant. For these goals the more ertinent architecture are the U-Net CNN, such as proposed from Hou Yuqing and Li Hui et. al. [7] for the prediction of flow around a submarine.

The U-Net architecture is composed by two branches: the firts perform the down-samplig of the input image, and the last one perform the upscaling of the output to obtain an image of the desired size (usually the same size of the input image). The term "U-Net" is derived from the shape of its block diagram, which is represented by a U-shaped structure. This is illustrated in Figure 2.7.

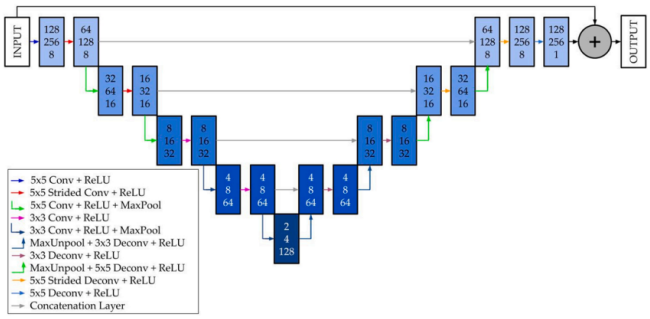

Figure 2.7: Scheme of a CNN with U-Net atchitecture [1]

This architecture is particularly suitable for the prediction of temporal data this is because of we used this type of CNN. In fact these architecture allow to "remember" data from the peceeding information thank so the presence of skip connection, that are link of the informations of different layers of the networks represented by the line the links parallel blocks of the scheme in Figure 2.7.

The training of these networks is frequently challenging due to their high sensitivity to input data noise and the vanishing gradient problem. In fact, we typically train these networks with experimental data (PIV images) or CFD data. Both types of data may be affected by noise due to errors resulting from measurement or calculation errors, so we introduce in the networks an intrinsic error which effects all outputs produced.

The phenomenon of a vanishing gradient occurs when the gradient is too small

so that the adjustment of neuron's weights is negligible, so that the network wont learn anymore and this don't allow the network to learn long terms information. To circumvent this issue, it is imperative to exercise caution when selecting the activation function, modify the learning rate during the training phase, use the skip connection in order to permit the gradient transfer between the network's layers. But the most efficient strategy to avoid the vanishing gradient is to add Long Short-Term Memory (LSTM) layers. The structure of these layers enables the network to recall the most significant long-term dependencies, which is why integrating these layers into CNNs for the prediction of turbulent flows can significantly enhance performance[1].

2.4 Residual Net

As will be demonstrated subsequently, a parametric study of the U-architecture revealed that the structure exhibiting optimal functionality was the simplest, thus comprising only a single Down block. Consequently, a convolutional neural network with a residual network (ResNet) architecture was implemented, as it exhibited a structural similarity to the U-Net with a single Down block. This was done to assess the efficiency of the CNN in predicting the jet flow.

ResNets were developed with the objective of circumventing the conventional issues associated with CNNs, namely vanishing and exploding gradients. As illustrated in the case of the U-Net, the utilisation of skip connections represents a means of addressing these challenges. The architecture is constituted by a specific number of residual blocks. The number of convolutions performed by the neural network is contingent upon the dimensions of the residual block. Furthermore, each convolutional layer is connected to the subsequent layer in a skip structure as shown in Figure 2.8 where two subsequential residual block are schematized.

As previously stated, the structure of these neural networks is notably simple. They all begin with an initial convolution, which increases the number of filters in the image without reducing the original image dimensions. This differs from the approach taken with the U-Net, where the number of filters was reduced. Following this, we have the residual blocks, which perform a specific number of convolutions with constant number of channels, so the depth don't increase.

2.5 Learning process

The training process of a neural network is the modification of the neuron's weights and bias with the objective of minimising a loss function that measure the distance between the desired output with the real output of the NN.

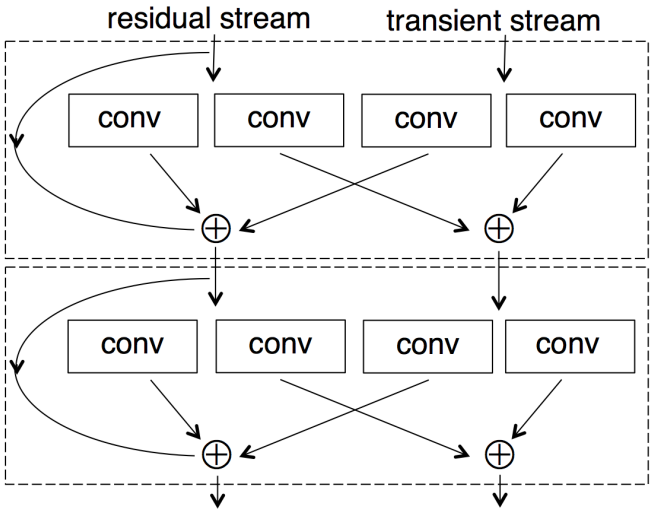

Figure 2.8: Structure of two consecutive residual blocks [8]

Training algorithms can be divided into numerous categories. The first of these is the distinction between supervised and unsupervised learning algorithms.

The batch size during the training of a CNN is a significant hyperparameter that directly influences the model's learning process. It determines the number of training samples that will be processed before the model's internal parameters are updated. A smaller batch size can lead to faster training iterations, as less computational memory is required, and it can also help the model to generalize better by providing a more regularized approach. However, smaller batches may result in a less accurate estimation of the gradient, which can affect the convergence of the model. On the other hand, larger batch sizes offer a more accurate estimate of the gradient direction but can be computationally expensive and may lead to overfitting if not managed correctly. Therefore, the choice of batch size is a balance between computational efficiency and the quality of the gradient estimation, which ultimately impacts the model's performance on new, unseen data. It's a critical decision that can affect the speed of training, the convergence behavior, and the overall success of the CNN's training process.

Supervised training algorithms for CNNs represent a fundamental component of contemporary deep learning, particularly in contexts involving image recognition and classification. The process commences with the provision of a labelled dataset, wherein each input (frequently an image) is associated with a correct output label. The CNN learns to map the input data to the correct labels through a series of layers that extract and transform features from the data. The initial layer, the convolutional layer, employs a variety of filters to process the input, thereby generating feature maps that emphasise particular aspects of the data.

Subsequently, the feature maps undergo downsampling through the incorporation of pooling layers, which serve to reduce the dimensionality and computational complexity of the data while maintaining the integrity of the information.

Subsequently, the features undergo further refinement and abstraction in the subsequent layers, with fully connected layers at the network's conclusion making decisions based on the abstracted features. The training process comprises two key stages: forward propagation, whereby input data is processed by the network to generate an output, and backpropagation, whereby the network adjusts its weights and biases to minimise the discrepancy between its output and the correct labels. This adjustment is achieved through the use of optimisation algorithms, such as stochastic gradient descent, which iteratively updates the network's parameters with the aim of reducing the loss function, which serves as a measure of the network's performance on the training data.

Regularisation techniques, such as dropout and weight decay, are frequently utilised to avert overfitting, thereby ensuring that the network exhibits effective generalisation capabilities with respect to novel, unobserved data. Furthermore, contemporary CNNs may employ techniques such as batch normalization to enhance stability of the learning process and accelerate convergence rates. Once trained, CNNs are capable of performing tasks such as image classification with high accuracy. Furthermore, they can be fine-tuned with additional data or adapted to new tasks through transfer learning.

In contrast, unsupervised learning pertains to unlabeled data. The algorithm must be capable of discerning patterns and relationships within the data without any external guidance or correction. This is analogous to self-directed learning, whereby the learner is required to identify structure within the information presented to them independently. Typical tasks for unsupervised learning algorithms include clustering and association. The objective in these cases is to group similar data points together or to identify rules that describe large portions of the data.

The decision between supervised and unsupervised learning is contingent upon the nature of the problem at hand and the type of data available. Supervised learning is frequently employed when the intended result is known and the objective is for the model to learn to predict or classify new data based on past examples. The utility of unsupervised learning lies in its capacity to facilitate the exploration of data, thereby uncovering hidden structures and patterns.

In practice, these two types of learning can also be combined in what is known as semi-supervised learning, whereby a small amount of labelled data is used in conjunction with a larger set of unlabeled data. Alternatively, reinforcement learning may be employed, whereby an agent learns to make decisions by receiving rewards or penalties.

Each method has its own particular strengths and limitations. Supervised learning can achieve high accuracy when a substantial amount of labelled data

is available; however, obtaining such data can be costly and time-consuming. Unsupervised learning can be applied to unlabeled data, which is often more readily available, but it may yield less precise results due to the lack of clear guidance during training. In summary, supervised learning algorithms learn from examples with known outcomes to make predictions or classifications, while unsupervised learning algorithms explore data to find patterns without prior knowledge of the results. The choice between them is dictated by the specific requirements and constraints of the machine learning task at hand.

The choice of the learning algorithm and loss function are often connected because some algoritmh perform better with a certain loss function.

The RMSprop algorithm Root Mean Square Propagation (RMSprop) is an optimisation algorithm employed in the training of NNs. The RMSprop algorithm is based on the Stochastic Gradient Descent (GSD) algorithm, but with certain modifications that enhance the stability and convergence velocity of the algorithm. The first step of this algorithm is to compute the loss function's gradient, it give an indication on the direction in wich the NN's parameters should be update to reduce the error. Then, based on the RMS value of the gradients of the previous training's epoch, it estimate the local gradient for each parameter. The Equation 2.3 is the formula for compute new value of the $j-th$ weight at the $t+1-th$ epoch.

$$\theta_{t+1}^{j} = \theta_{t}^{j} - \frac{\gamma}{\sqrt{v_t^j + \varepsilon}} \left(\frac{\partial E_j}{\partial w_j} \right)_t \tag{2.3}$$

dove

- γ: is the learning rate;
- ε: is a small value that is added to prevent the denominator from being zero.

v_t^j is:

$$v_t^j = \gamma v_{t-1}^j + (1-\alpha) \left(\frac{\partial E_j}{\partial w_j} \right)_t^2 \tag{2.4}$$

where α is a factor that indicate a fraction of the gradient decay.

This solution uses the partial derivative of the loss function in the j-th weight to correct the step size of the gradient descent. The algorithm 1 shows the operation of the RMSprop algorithm implemented in the PyTorch library [9].

As we noted before, the advantages of this algorithm are the stability and the convergence velocity. In general this emprove the algorithm's efficiency, but it's heavly influenced by the learning rate γ choice.

Another algorithm largely used for the weights optimization of this NN's is the Adaptive Moment Estimation (Adam) algorithm. It combines the advantages of two optimizations algorithm: the Gradient Discent with Momentum (GDM) and

Algorithm 1 RMSprop optimizer

procedure RMSprop($\alpha, \gamma, \theta_0, f, \lambda, \mu$)
  ▷ Initializes parameters
  $v_0 \leftarrow 0$
  $b_0 \leftarrow 0$
  $g_0^{avg} \leftarrow 0$
  for `t=1 to ...` **do**
    $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$
    if $\lambda \neq 0$ **then**
      $g_t \leftarrow g_t + \lambda\theta_{t-1}$
    end if
    $v_t \leftarrow \alpha v_{t-1} + (1-\alpha) g_t^2$
    $\tilde{v}_t \leftarrow v_t$
    if centered **then**
      $g_t^{avg} \leftarrow g_t^{avg}\alpha + (1-\alpha)g_t$
      $\tilde{v}_t \leftarrow \tilde{v}_t - (g_t^{avg})^2$
    end if
    if $\mu > 0$ **then**
      $b_T \leftarrow \mu b_{t-1} + \frac{g_T}{\left(\sqrt{\tilde{v}_t}+\varepsilon\right)}$
      $\theta_t \leftarrow \theta_{t-1} - \gamma b_t$
    else
      $\theta_t \leftarrow \theta_{t-1} - \frac{\gamma g_t}{\left(\sqrt{\tilde{v}_t}+\varepsilon\right)}$
    end if
  end for
  return θ_t
end procedure

the RMSprop. The GDM algorithm, using momentum μ to accelerate the learning process. The momentum is a parameter that takes into account the gradient of the previous iterations to damp the oscillations and thus drive the parameters to a more stable minimum. The algorithm update the exponential moving averages m_t and v_t, respectively of the gradient and the sqared gradient using the parameters β_1 and β_2 that control the exponential decay. This averages are initized to zero, so that in the first training's epoch, are nearest to zero . Based on the moving averages, the weights are updated with the following formula:

$$\theta_t = \theta_{t-1} - \gamma \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{2.5}$$

The advantages of this algorithm are:

- **Efficiency:** Adam algorithm converge faster than other algorithm such as GDM or SGD, so the training process require fewer iteratons to minimize the loss function;

- **Stability:** Adam algorithm is less prone to oscillation of the gradient than other algorithms;

- **Scalability:** Adam algorithm is well suited to models also with much many parameters;

- **Ease of use:** Adam algorithm require only few parameter to set, i.e. the damping parameters and the initial learning rate.

2.5.1 Weights initialization

Weights initialization before the training process begin, could be very important, in fact an appropriate inizialization of the neural network's parameter can enormously reduce the necessary training's epoch to reach the convergence, otherwise if the weights has random value the network may have difficulty learning, or may converge to a non optimal solution and begin to oscillate around it without converge to it. local minimum. So an appropriate weight initialization garantees us that the NN's training start from a favourable point to reach fast and accurate minimum value and also can help to avoid the vanishing gradient. The conventional approach is to initialize parameters with a standard distribution, fixing the standard deviation. However, in certain applications, this initialization hinders convergence, and the deeper the model, the more challenging it is to achieve convergence.

To avoid these problems, several weight initialisation methods have been developed in the history of AI, such as the Xavier initialisation or the Xe initialisation, each with its own peculiarities depending on the activation function used.

The selection of the initialisation method is contingent upon the intended application of the neural network and the activation function of each layer. For instance, He et al. [10] have demonstrated that the Xavier initialisation is not optimal solution and begin to oscillate around it without converge to it. when utilising the ReLU activation function.

In this work we use the Kaiming initialization proposed in the reference [10]. This was already implemented in PyTorch environment [9] in the function named *kaiming_normal_*. This function assign to the neural network's weights tensor a value from a normal distribution with standard deviation:

$$std = \frac{gain}{\sqrt{fan\_mode}} \tag{2.6}$$

where fan_mode define if we want a constant variance of the weights in the forward process (with $fan\_mode = fan\_in$) or in the bckward process (with $fan\_mode = fan\_out$).

In this work we initialize the weights of convolutional layers with the Kaiming initialization, meanwhile for the BatchNormalization layers the weights has been intialized with a simple normal distribution with constant variance and the bias parameters has been initialized to a constant null value.

2.5.2 Setting up the learning rate

In the context of machine learning, the value of the learning rate represents a hyper-parameter of fundamental relevance for the training efficiency. In fact, it determines the rate of change of the model's parameters during the training process, significantly influencing the convergence and the quality of the model.

Setting the learning rate too high could lead to the phenomenon of overshoot, which occurs when the network overtakes the optimal solution and begins to oscillate around it without converging to it. Conversely, if the learning rate is too low, the gradient may decay slowly and inefficiently, prolonging the model's training times and preventing it from reaching an optimal solution.

To avoid these problems scheduler has been implemented. Scheduler are function that update te value of the learning rate during the training process. This technique consists of dynamically varying the value of the learning rate according to various criteria, such as the number of epochs completed, the error reduction rate or the size of the gradient. This adaptive approach balances the learning rate with the accuracy of the model, optimising the training process.

In conclusion, the learning rate plays a key role in the training of neural networks, influencing both the speed and quality of learning. By ensuring that the learning rate is correctly set and that adaptation techniques are employed, it is possible to obtain optimal models in a reasonable time.

For these reason, during the training preocess, scheduler are setted up to vary the learning rate according to optimize the learning process of the neural network.

The choise the scheduler function adopted is driven by the behaviour of the model during the training. In this work we have tested two type of scheduler, the *CosineAnnilingLR* e il *CyclicLR* both implemented in the PyTorch environment [9].

The CosineAnnilingLR brings down the learning rate as the cosine in a certain number of epochs:

$$\begin{cases} \eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left[1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right] & T_{cur} \neq (2k+1)T_{max} \\ \eta_{t+1} = \eta_t + \frac{1}{2}(\eta_{max} - \eta_{min})\left[1 + \cos\left(\frac{1}{T_{max}}\pi\right)\right] & T_{cur} = (2k+1)T_{max} \end{cases} \tag{2.7}$$

where:

- η_{max} is the initial value of the learning rate;
- T_{max} is the max number of iterations of the scheduler;
- η_{min} is the minimum value of the learning rate to be achieved;
- T_{cur} is the number of epoch after the last warm restart.

In Figure 2.9 is reported the trend of this scheduler.

The last scheduler tested is the CyclicLR scheduler, this causes the learning rate to fall linearly for a number of epochs, after which it rises again. Its trend is showed in Figure 2.10.

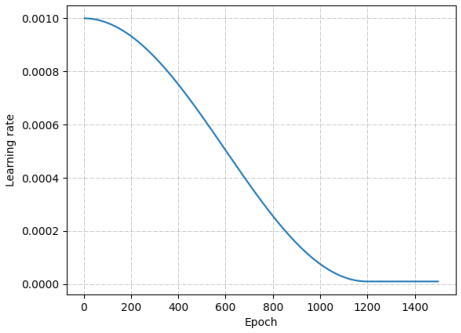

Figure 2.9: Trend of the CosineAnnilingLR scheduler

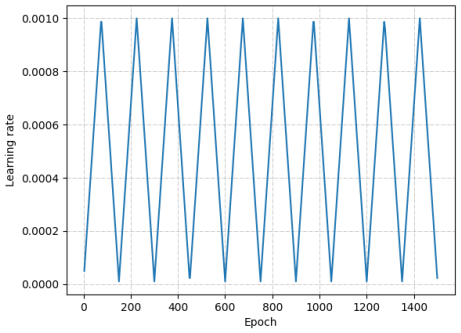

Figure 2.10: Trend of the CyclicLR scheduler

2.6 Conditional Generative Adversarial Networks

The proposed neural network is part of the *image-to-image translation* networks, i.e. generative networks that from one image generate a different one, in this case from the image of a velocity field of the flow of an axisymmetric jet generate the image of the same field at the next instant.

Classical CNNs during the training process decrease the value of the loss function, which is a function that provides a measure between the predicted and actual image. This learning process is automatic and for the learning process to be effective, it is essential to select the most appropriate loss function in order to tell the process which is the target to be minimised. For example, selecting the Euclidean distance between the generated and real pixels as the target to be minimised can lead the CNN to generate blurred images, this happens because the Euclidean distance is

minimised by averaging all possible outputs, it is often not easy to select the right target for the intended purpose and this requires some experience.[11]

Conditional Generative Adversarial Networks (cGANs), represent an advanced architecture of artificial neural networks used for the generation of realistic data from conditional data. Unlike standard GANs, cGANs integrate additional information into the generation process, resulting in more controlled and realistic results.

A cGAN is constituted of two neural networks that are in conflict with one another:

- **Generative network:** A generative network is a CNN that has been trained to learn the conditional distribution of data and to generate new data that is similar to the real data, but with specific characteristics that are determined by the input conditions. The structures of the new data are perfectly equivalent to those seen previously.

- **Discriminator network:** The discriminator network is also a CNN trained to distinguish between data that is real and data that has been generated by the generator. The objective of the discriminator is to correctly classify the data as either real or generated.

CGANs offer significant advantages over standard GANs. Firstly, they offer greater control over generation because the conditioning information allows data with specific desired characteristics to be generated, thereby improving the controllability of the generation process. Secondly, the results are more realistic than those generated by standard GANs. Indeed, as illustrated by U. Demir and G. Unal [12], these networks are capable of generating images with greater definition, which could potentially yield more precise velocity field values in the case study.

The discriminator network receives as input the images generated by the generator that have been identified as 'fake' and the real images. The objective of the discriminator is to ascertain which images have been generated and which are the real images. In doing so, the discriminator attempts to deceive the system, thereby generating images that are increasingly similar to the real images.

In mathematical terms the loss function can be espressed as:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}\left[logD(x, y)\right] + \mathbb{E}_{x,z}\left[log(1 - D(x, G(x, z)))\right] \tag{2.8}$$

where x are the input data, y are the generated data and z is the Gaussian noise.

A recent study has demonstrated that combining the traditional GAN objective with the L1 loss function enhances the visual quality of the generated images while simultaneously reducing the occurrence of blurring. So the final objective is expressed as:

$$G^* = arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \tag{2.9}$$

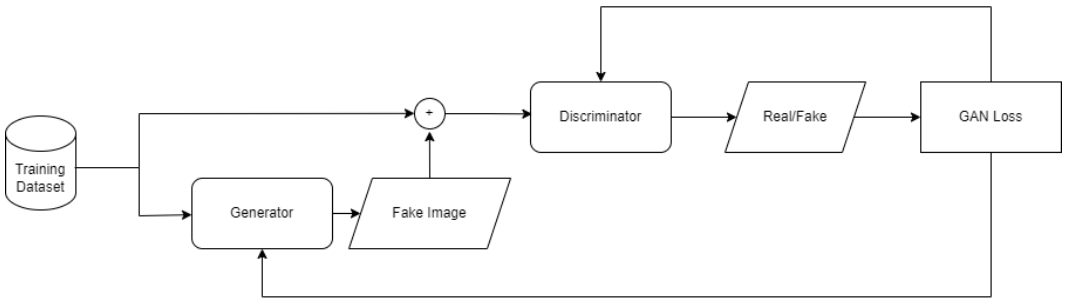

Figure 2.11: Flow chart of the raining diagram of a cGAN network [13]

The L1 loss function represents a criterion that quantifies the mean absolute error between each element in the predicted images and the corresponding elements in the real images.

The elimination of Gaussian noise from the data enables the network to learn a mapping from x to y, thereby facilitating the generation of less stochastic results.

Algorithm 2 Adam optimizer algorithm.

procedure ADAM($\gamma, \beta_1, \beta_2, f, \theta_0$, *amsgrad*, *maximize*)
  ▷ α is the stepsize
  ▷ $\beta_1, \beta_2 \in [0, 1)$ are the exponential decay rates for the moment estimates
  ▷ $f(\theta)$ is the objective function to optimize
  ▷ θ_0 is the initial vector of parameters which will be optimized
  ▷ Initialization
  $m_0 \leftarrow 0$ ▷ First moment estimate vector set to 0
  $v_0 \leftarrow 0$ ▷ Second moment estimate vector set to 0
  $t \leftarrow 0$ ▷ Timestep set to 0
  ▷ Execution
  for `t=1 to ...` **do**
    ▷ Gradients are computed w.r.t the parameters to optimize
    ▷ using the value of the objective function
    ▷ at the previous timestep
    if maximize **then**:
      $g_t \leftarrow -\nabla_\theta f_t(\theta_{t-1})$
    else:
      $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
    end if
    if $\lambda \neq 0$ **then**
      $g_t \leftarrow g_t + \lambda\theta_{t-1}$
    end if
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$
    ▷ Bias-correction of estimates
    $\hat{m}_t \leftarrow \dfrac{m_t}{1 - \beta_1^t}$
    $\hat{v}_t \leftarrow \dfrac{v_t}{1 - \beta_2^t}$
    if amsgrad **then**:
      $\hat{v}_t{}^{max} \leftarrow max(\hat{v}_t{}^{max}, \hat{v}_t)$
      $\theta_t \leftarrow \theta_{t-1} - \gamma \cdot \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$
    else:
      $\theta_t \leftarrow \theta_{t-1} - \gamma \cdot \dfrac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$
    end if
  end for
  return θ_t ▷ Optimized parameters are returned
end procedure

Chapter 3

Axisymmetric jet flow

A jet is defined as a flow generated when a fluid exits an orifice into another ambient medium, where it is assumed that the jet is made by the same fluid that exits the orifice. If the orifice has a circular geometry, the jet is axisymmetric. The jet falls into the family of free shear flows, which are inhomogeneous flows whose main velocity gradient develops in the absence of walls. These flows have been much studied because they involve many of the natural and engineered processes, such as combustion or contrails.

As is the case with all turbulent flows, the field of motion of the jet is three-dimensional. However, for the sake of simplicity, this work considers only two dimensions: the streamwise direction and the radial one.As all the fluid dynamics problems, the physics of axisymmetric jet is well described by the Navier-Stokes equations that derived from the conservation law of mass, momentum and anergy:

$$\begin{cases} \dfrac{\partial \rho}{\partial t} + \nabla \cdot \left(\rho \vec{V}\right) = 0 \\ \dfrac{\partial \vec{V}}{\partial t} + \left(\vec{V} \cdot \nabla\right) \vec{V} = -\nabla p + \vec{F} + \mu \nabla^2 \vec{V} \\ \dfrac{\partial (\rho E)}{\partial t} + \nabla \cdot \left(E \rho \vec{V}\right) = -\nabla \cdot \left(p \vec{V}\right) + \mu \nabla \cdot \bar{\tau} + \rho \vec{g} \cdot \vec{V} - \nabla \cdot \vec{q} \end{cases} \tag{3.1}$$

The motion field of the jet is characterised by the presence of four mean regions, as schematically illustrated in Figure 3.1:

- **Potential core:** the potential core is a conical region of the motion field that is characterised by costant velocity and equal to the fluid's velocity at the exit of the nozzle U_∞;
- **Mixing region:** this region envelops the potential core and extends across the entire width of the potential core. In this regions there is a velocity decay due to the viscous effects;

- **Transition region:** in the transition region, there are no points at which the velocity is equal to the free-stream velocity, U_∞. This is because the viscous effects on each point of the motion field result in a parabolic trend in the velocity profile $u(r)$.
- **Self-similar region:** from a certain distance x_2 until the total exhaustion of the flow, if we normalise the velocity profiles by considering an appropriate length, we observe a perfect equality between these. The normalization that show the similarity of vleocity profile is:

$$\frac{u(x,r)}{U_{max}(x)} = f\left(\frac{r}{r_{u=0.5U_{max}(x)}}\right) \tag{3.2}$$

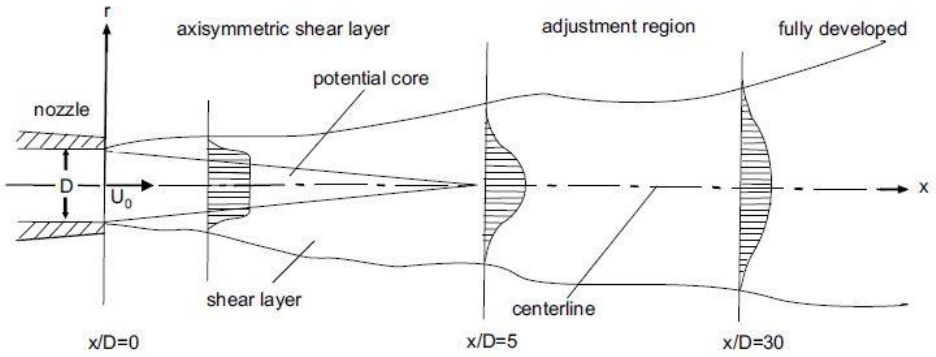

Figure 3.1: Scheme of the jet flow's regions [15]

The velocity on the axis of the jet, $U(x,0)$, is observed to decline in accordance with an exponential law, showed in Equation 3.3:

$$\frac{U_{max}(x)}{U_0} = \frac{k}{(x/D)^n} \tag{3.3}$$

Thus far, the motion field has been observed in time-averaged space. However, turbulence is characterised by rapid fluctuations in time of the physical quantities, which necessitates an extension of the temporal perspective to fully comprehend its dynamics.

The study of turbulence dynamics has a history of some 160 years, beginning with the first investigations by Hermann von Helmholtz in 1858. Despite this long history, turbulence dynamics remains a subject of great scientific interest, with many open problems still to be solved due to its inherent complexity.

Turbulence is a chaotic motion of fluid in the motion field, this is because turbulence is a state of continuous instability, so the fluid organises itself into vortices to seek stability. This causes large temporal variations in the physical quantities describing the fluid motion, and therefore turbulent motion is very difficult to predict because the variation is arbitrary.

Given the inherent variability of turbulent flows, their characterisation is typically conducted in terms of statistical quantities, which can be reproduced through experimental means through measurement methods characterised by a high frequency response, such as hot wire anemometry or optical techniques.

Entrainment in axisymmetric jet flow refers to the process by which a fluid jet, issuing from a nozzle into a quiescent or moving ambient fluid, incorporates surrounding fluid into its flow structure. This phenomenon is crucial for understanding how jets mix with their environment, which has implications for various applications, from industrial processes to environmental science. In an axisymmetric jet, the entrainment occurs as the jet's momentum causes it to spread radially, creating a shear layer where the jet fluid and ambient fluid interact. This interaction leads to the entrainment of ambient fluid, which is then mixed into the jet due to turbulence. The dynamics of this process are complex, involving factors such as the jet's velocity profile, the density difference between the jet and the ambient fluid, and the presence of any external turbulence. Recent studies have shown that external turbulence can significantly affect the entrainment process by altering the topology of the scalar turbulent/turbulent interface (TTI), which differs from the traditional turbulent/non-turbulent interface (TNTI). The TTI is characterized by a greater thickness and an increased function of the background turbulence intensity, which suggests that large-scale engulfment plays a more significant role in the entrainment process under these conditions. Additionally, the presence of ambient turbulence can lead to the formation of concentration 'holes' within the interfacial layer and detached jet patches, or 'islands,' indicating intense detrainment events. These phenomena contribute to the overall complexity of the entrainment process in axisymmetric jet flows and highlight the need for further research to fully understand the interplay between jet dynamics and ambient conditions. [16]

The Reynolds number plays a pivotal role in the entrainment process of jet flows, serving as a dimensionless parameter that characterizes the flow's regime. It is defined as the ratio of inertial forces to viscous forces and is used to predict the transition from laminar to turbulent flow. In the context of jet entrainment, a higher Reynolds number typically indicates a more turbulent flow regime, which is associated with increased mixing and entrainment due to the presence of larger and more energetic eddies. These eddies enhance the entrainment of ambient fluid into the jet, contributing to its growth and the mixing of substances within the flow. Studies have shown that as the Reynolds number increases, there is a decrease in the centerline axial velocity decay rate and reduced turbulence intensities, suggesting

a more efficient mixing process. The jet spread rate, however, is observed to be independent of the Reynolds number, indicating that factors other than the Reynolds number also influence the spread of the jet. Additionally, high Reynolds number jets experience an early transition to a turbulent regime, which can lead to a more rapid mixing and entrainment of the surrounding fluid. This early transition is characterized by an early potential-core collapse, which is the point where the maximum velocity within the jet decreases sharply due to the increased mixing with the ambient fluid. The dynamics of jet entrainment are complex and are influenced by a multitude of factors, including the initial velocity profile of the jet, the density difference between the jet and the ambient fluid, and the level of ambient turbulence. Understanding the role of the Reynolds number in jet entrainment is crucial for optimizing industrial processes such as fuel injection in combustion engines and for environmental applications like pollutant dispersion in the atmosphere. [15]

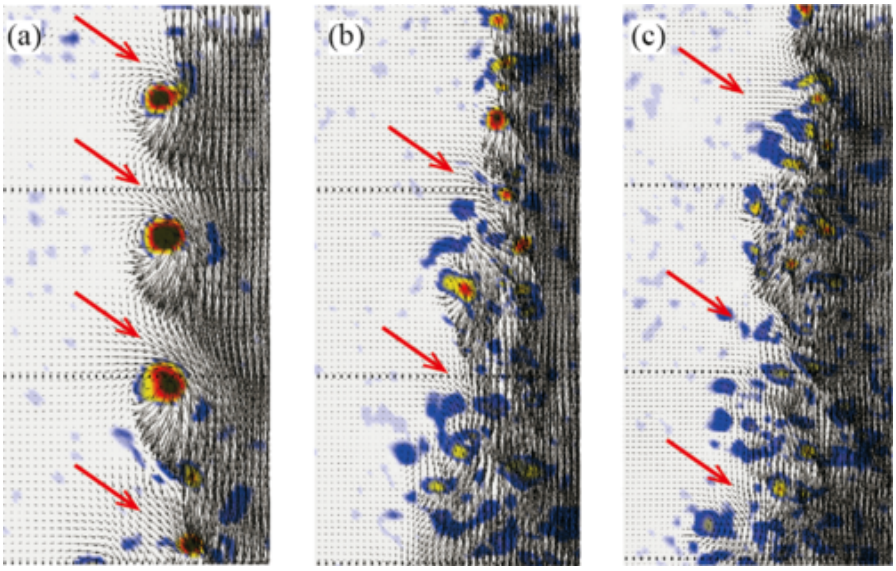

Figure 3.2: Close-up views of the shear layer of jets at different Reynolds number, red arrows highlight entrainment zones [17]

3.1 Momentum and energy

Applying the principle of momentum conservation to the jet flow:

$$M = 2\pi \int_0^{+\infty} \rho u^2 dr \tag{3.4}$$

The costant momentum along the axis of an axisymmetric jet flow is predominantly attributable to the absence of external forces acting upon the fluid. This implies that external pressures and forces that could disrupt the flow are absent, allowing the momentum to remain constant. Furthermore, the absence of dissipative forces, such as wall shear stress, is of great consequence. In numerous fluid dynamics scenarios, wall shear stress can markedly dissipate the momentum of the fluid by introducing frictional forces that impede the flow. However, in the case of an axisymmetric jet, the absence of such boundaries ensures that the momentum is conserved along the axis.

$$E = \frac{dE_c}{dt} = \frac{1}{2}Gu^2 = \pi \int_0^{+\infty} \rho u^3 r dr \tag{3.5}$$

In the potential core of an axisymmetric jet flow, the kinetic energy is maintained at a constant value due to the absence of viscous effects. This region is distinguished by an inviscid flow, wherein the fluid particles are observed to move without internal friction, thereby enabling the conservation of kinetic energy. The potential core is typically located in the vicinity of the jet's exit, where the velocity profile is uniform and the effects of viscosity are negligible.

Conversely, as the fluid moves away from the potential core into the shear layers and the outer regions of the jet, the effects of viscosity become increasingly evident. In these regions, the interaction between fluid layers of disparate velocities gives rise to the formation of shear stresses. The viscous forces exerted upon the fluid cause it to experience friction, which results in the dissipation of kinetic energy. The energy lost due to viscous dissipation is converted into thermal energy, which results in a reduction in the overall kinetic energy of the fluid as it moves further from the jet axis.

The dissipation function is defined as:

$$\varepsilon = \frac{\delta H}{\delta V} \tag{3.6}$$

where δH is the internal thermal energy and δV is the element's volume. In this case the dissipation function will be:

$$\varepsilon = \frac{1}{2}\nu \left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right)^2 \tag{3.7}$$

In the field of fluid dynamics, the dissipation function represents a conceptual framework that elucidates the manner in which the mechanical energy inherent to fluid flows is transformed into thermal energy as a consequence of viscous effects. This function is of particular significance in the study of turbulent flows, where the viscosity of the fluid results in a continuous conversion of flow kinetic energy into thermal energy, thereby reducing the flow's mechanical energy. In essence,

the dissipation function serves to quantify the rate of this energy conversion. It is a pivotal element in the energy equilibrium of a fluid system, influencing the behaviour of the flow in significant ways, particularly in engineering contexts where heat generation and energy efficiency are paramount considerations. An understanding of the dissipation function can assist engineers in the design of systems that either minimise energy loss due to dissipation or exploit the heat generated by the process.

3.2 Turbulence dynamics in axisymmetric jet flow

The most significant structures are those of a small scale, as they are typically defined as concentrated regions of high enstrophy with a lifetime exceeding the characteristic time scale of the flow. These structures are the most responsible for energy dissipation. In recent years, the growth of computational capabilities has enabled a comprehensive investigation of these structures.

These structures appear to share some universal features, such as the preferential alignment of the vorticity vector with the direction of the mean principal strain, and their average radius is about five times the Kolmogorov length scale. [18]

The Kelvin-Helmholtz (KH) mechanism has long been employed to describe coherent structures in both transitional and turbulent planar shear layers and jets. This mechanism is fundamental to the field of fluid dynamics and plays a crucial role in understanding the behaviour of various flow systems. It should be noted that the KH mechanism is not defined in isolation; rather, it is defined in the context of parallel or quasi-parallel laminar shear layers. In these contexts, the mechanism is well characterised by the spatial stability theory, in which the KH is identified as an unstable modal solution with an associated spatial growth rate.[19]

In the context of jets, the KH mechanism manifests in a distinct manner. The solution is typically a convective instability, whereby disturbances grow as they are convected downstream. This is in contrast to absolute instabilities, whereby disturbances grow in situ. As the jet propagates, the initially growing wave, which oscillates at a fixed frequency, will eventually become neutral and begin to decay. The transition from growth to decay is influenced by the spreading of the flow and the changing conditions within the jet.

The KH mechanism's capacity to elucidate these coherent structures is of paramount importance for forecasting and regulating flow behaviour in a plethora of engineering applications, including aerodynamics, meteorology, and even astrophysics. An understanding of this mechanism facilitates the design of more efficient systems and the mitigation of the adverse effects of turbulent flows.

The study of Tissot G. et. al [20] supports the presence of the Orr mechanism

in the downstream region of jets, where disturbances grow nonmodally in response to nonlinear forcing. The Orr mechanism is a transient energy growth mechanism proposed by Orr [21], who demonstrated that the perturbation energy of a given velocity field can grow transiently in time as the perturbation field, initially tilted upstream, is gradually tilted downstream by mean shear.[22]

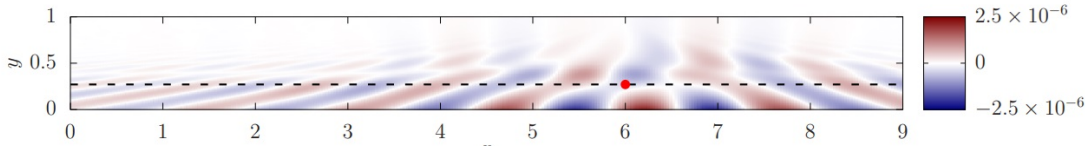

Figure 3.3: Spatial Orr solution $[Re(u)]$ with inflow calibrated to the jet wave packet response $\delta\tilde{q}$ at $x/D = 6$. The red point indicates the position where the conditions from the PSE (shear, mean flow velocity, inflow profile and local wave number) are used to construct the Couette flow approximation. The dashed line is the line where the PSE and the Orr model are compared.[20]

3.3 Large scale vortices

The presence of streamwise vortices gives rise to the formation of streaks through the lift-up mechanism, which constitutes a significant aspect of jet dynamics. The streamwise vortices are defined by their vorticity, which is a measure of the local rotation of the fluid. The interaction of these vortices with the surrounding fluid results in the generation of a secondary flow, which in turn drives the lift-up mechanism. Furthermore, the lift-up mechanism can be viewed as an outcome of the proliferation of specific flow instabilities. Such instabilities can extract energy from the mean flow and convert it into the kinetic energy of the streaks.

The process commences with the emergence of streamwise vortices. Such structures are characterised by a rotational motion aligned with the direction of the main flow. In a turbulent jet, these vortices can be generated by a variety of instabilities and interactions within the flow. As these vortices move through the fluid, they induce a secondary flow that is perpendicular to the main flow direction. This secondary flow lifts low-momentum fluid from the jet's core region and transports it outward towards the higher-momentum regions near the edges of the jet. The lift-up mechanism is crucial for the redistribution of energy within the flow. By lifting low-momentum fluid into high-momentum regions, the mechanism enhances mixing and promotes the transfer of kinetic energy from the mean flow to the turbulent structures.

Streaky structures in turbulent jets are characterized by their elongated, band-like shape. These bands can extend several diameters downstream from the nozzle

exit, making them prominent features in the flow. These structures consist of alternating regions of high and low momentum fluid. The high-momentum streaks are typically faster-moving fluid, while the low-momentum streaks are slower-moving fluid. The streaks are aligned with the direction of the main flow (streamwise direction). This alignment is a result of the lift-up mechanism, which stretches the fluid elements in the direction of the flow.

As discussed earlier, the lift-up mechanism is central to the formation of streaky structures. Streamwise vortices lift low-momentum fluid from the jet's core and transport it outward, creating the streaks. These structures play a significant role in the redistribution of kinetic energy within the jet. By transporting low-momentum fluid into high-momentum regions, they enhance mixing and energy transfer. The streaky structures are often associated with the growth of certain flow instabilities. These instabilities can extract energy from the mean flow and convert it into the kinetic energy of the streaks.

The streaky structures interact with smaller-scale turbulent eddies, influencing the overall turbulence dynamics. This interaction is crucial for understanding the mixing and spreading of the jet. Spectral analysis reveals that streaky structures dominate the low-frequency range of the flow. This indicates their large spatial extent and slow temporal evolution compared to smaller-scale turbulence. [23]

Chapter 4

Methodology

This chapter is dedicated to the comprehensive exploration of the methodologies that have been employed in our investigation of turbulence dynamics using Explainable AI (XAI) techniques. Our primary focus is on the application of Convolutional Neural Networks (CNNs) and the SHAP (SHapley Additive exPlanations) algorithm, both of which have been instrumental in analyzing and predicting turbulent flow patterns.

We begin this chapter by providing a detailed description of the dataset that forms the backbone of our research. This dataset comprises images of the motion field of an axisymmetric jet, which were meticulously obtained through the Particle Image Velocimetry (PIV) technique. The PIV technique is a powerful tool in fluid dynamics, allowing us to capture two- or three-dimensional velocity fields in fluid flows by taking advantage of the light scattered by tiny particles within the fluid.

Following the discussion on the dataset, we outline the structure and training process of the neural networks used in our study. We delve into the architecture of the CNNs, discussing the various layers and their functions. We also touch upon the importance of hyperparameter optimization in the training process, explaining how the right choice of hyperparameters can significantly improve the performance of the neural networks.

In the latter part of the chapter, we explore the role of the SHAP algorithm in our research. The SHAP algorithm is a game-theoretic approach to explain the output of any machine learning model. In our study, it provides interpretative insights into the predictions made by the neural networks. By doing so, it enhances our understanding of the underlying physical phenomena governing turbulence, making our research not just about prediction, but also about understanding.

This chapter serves as a bridge between the theoretical foundations laid out in the previous chapters and the results that will be presented in the upcoming chapters. It is our hope that the methodologies discussed herein will provide a clear understanding of the tools and techniques used in our research, thereby paving the

way for the interpretation of the results in the context of turbulence dynamics.

4.1 Dataset description

The dataset utilised for the training of the neural networks contains images of the motion field of an axisymmetric jet, obtained experimentally by the Particle Image Velocimetry (PIV) 2D2C technique. This technique involves measuring two velocity components on a plane of the motion field. Figure 4.1 schematises the experimental setup.

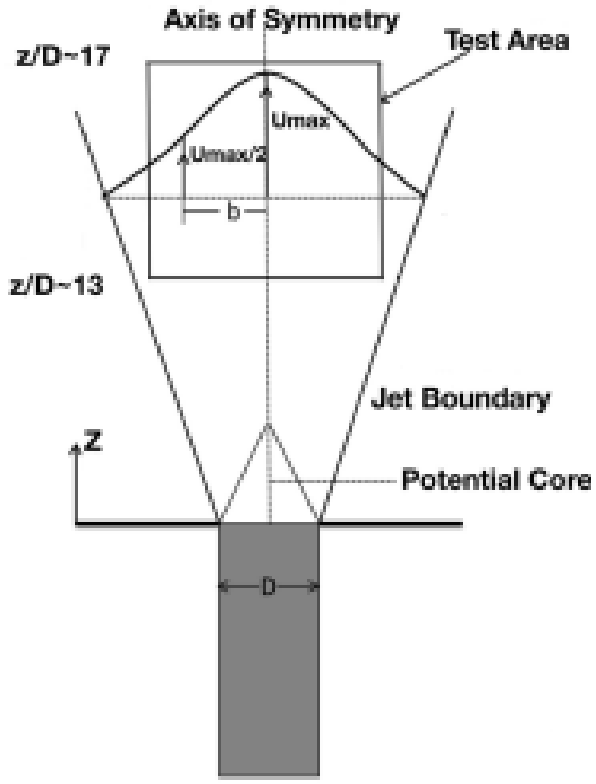

Figure 4.1: Experimetal setup of the PIV tecnique to obtain the dataset[24]

The dataset comprises 8451 instantaneous snapshots, each of which contains the w components, which represent the streamwise component of the velocity, and the u components, which represent the radial component.

The spatial mesh grid has 67×51 points spaced in both direction of $\Delta z = \Delta x = 0.268mm$. Meanwhile the time resolution is $\Delta t = 0.01ms$ so that the the turbulence temporal micro-scale is about seven times greater then the Δt. In Figure 4.2 there is the time-averaged velocity field.

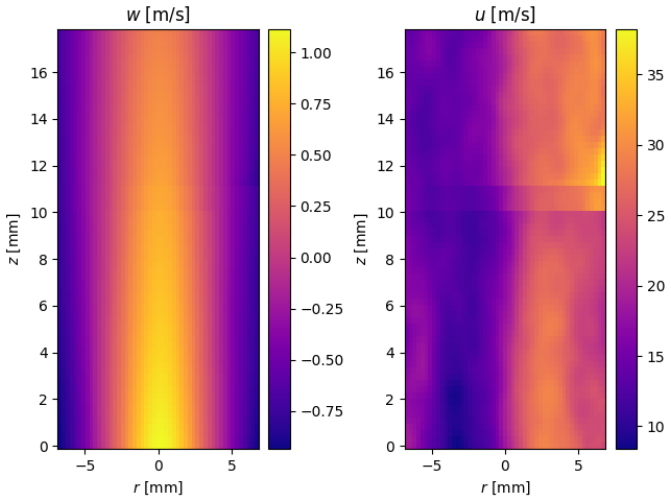

Figure 4.2: Time-averaged velocity field

In the case of bidimensional flow the vorticity field as only one component that is orthogonal to the velocity field:

$$\vec{\omega} = \nabla \times \vec{V} = \left(\frac{\partial w}{\partial x} - \frac{\partial u}{\partial z} \right) \hat{j} \tag{4.1}$$

Reynolds	21000
Inlet velocity	66 m/s
Taylor Micro-Scale	327 μm
Temporal scale	0.07 ms

Table 4.1: Jet data [25]

In order to facilitate the training process, the dataset has been normalised using the minmax method, as outlined in reference [26]. This has resulted in the generic dimensional physical quantity, denoted by the symbol x, becoming:

$$\phi = \frac{x - min(x)}{max(x) - min(x)} \tag{4.2}$$

where:

- x: are the values of velocity fluctuations;
- $min(x)$: is the minimum value of the velocity for each point;

- $max(x)$: is the maximum value of the velocity for each point.

In order to ensure that all datasets contain velocity values that have been normalised between 0 and 1.

It is essential to implement a standardisation process for the values in order to maintain data consistency and to prevent excessive high velocities from overwhelming lower fluctuations. This is crucial when algorithms necessitate a uniform distribution of the data or when employing methods that are sensitive to scale variation, such as clustering algorithms or gradient-based optimisation algorithms in the training of the CNN.

4.2 Neural Networks

As we have seen the CNN with U-Net architecture can be structured with many layers of deepness. How many layers the network has influence its work and its cost.

If the network has too much deep layers, we could encour to the overfitting phenomenon during the training, such as the neural network will learn too well the snapshots in the training dataset, meanwhile it will tend to decrease its capability to predict the flow field at a random istant. Also in this case wi will have too many parameters in the CNN so it will be very big and so will require high computation performance.

In the event that the network lacks an adequate number of parameters, it will be unable to accurately predict the motion field. So that when we construct the CNN we must do a parametrical analysis of the structure of the CNN.

So we have trained six CNN with U architecture, changing the deepness of the network and the number of convolution for each layer. In Figures A.1, A.2 and A.3 there are the schemes of the structure of the netowrks analyzed with different deepness layers, in order we have one, two and three deepness layers.

As we have seen in section 2.3 for each deepness layer of the network a certain number of convolutions are performed, so we have tested the network also changing this hyper-parameter to optimize the CNN workings.

The blocks of the neural network were constructed with reference to the methodology employed by D. Schmekel, F. Alcántara-Ávila et. al. [27] as shown in Figure 4.3

As illustrated, the convolutional layer is followed by batch normalization and the activation function. The batch ormalization is a mathematical operation defined as:

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \varepsilon}}\gamma + \beta \tag{4.3}$$

where:

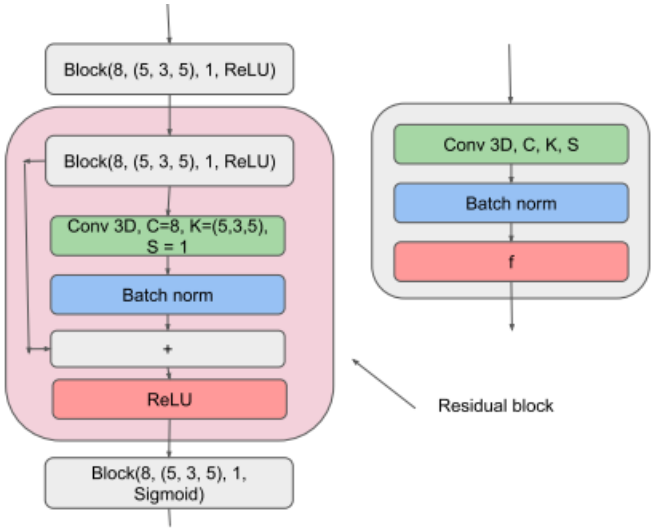

Figure 4.3: Diagram of the convolutional blocks [27]

- $E[x]$ is the averaged x;
- $Var[x]$ is the variance of x;
- ε is a very small value necessary for not having 0 in the denominator (default $\varepsilon = 1 \times 10^{-5}$ in PyTorch library);
- γ and β are learnable parameters that the model acquires during the training process.

The purpose of having batch normalisation layers is to reduce the training time and improve the performance of the model.

Subsequently, the activation function determines the output value of the neuron based on its input, introducing a non-linearity that allows the network to learn complex relationships between data. The selection of an activation function is contingent upon the nature of the problem to be solved and the attributes of the input and output data.

The implemented U-Net is composed of three main blocks. The first block performs the convolution in the straight branch of the U, followed by batch normalisation and the ReLU activation function. The block receives as input the number of channels of the input tensor, the number of channels at the output of that block and the number of convolutions to be performed. In Figure 4.4 is shown a functional scheme of this block.

The subsequent stage is to implement the down blocks, which are those blocks that perform the downsampling of the image, thus representing the descent part of

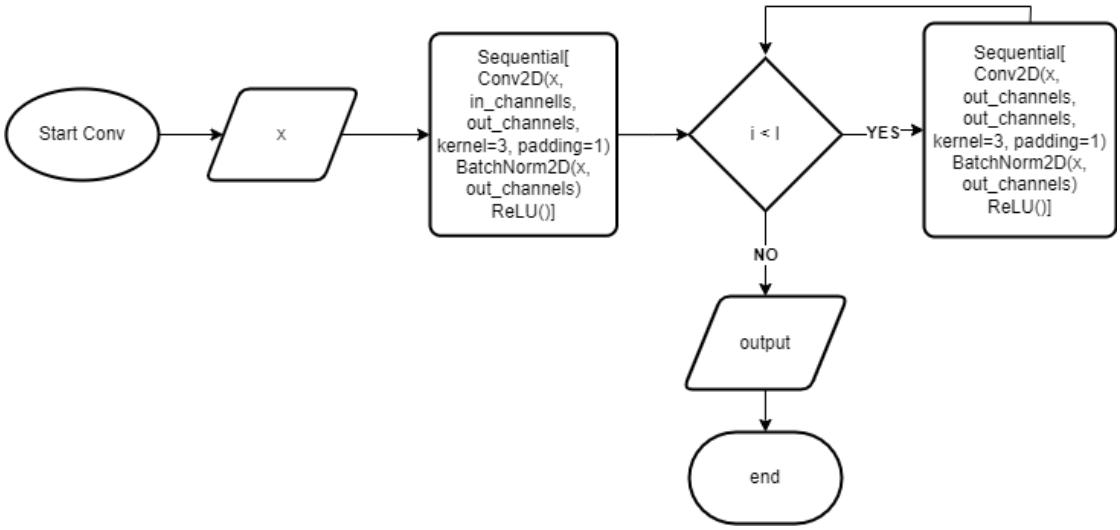

Figure 4.4: Functional diagram of the convolutional block

the U. The down function accepts as input the image tensor from the preceding block, along with the numbers of channels of the input and output tensors and the number of convolutions in the straight part of the U. The downsampling is performed by the function MaxPool2D, implemented in the Pytorch environment[9]. The kernel size was set to 2×2, resulting in a halving of the image size.

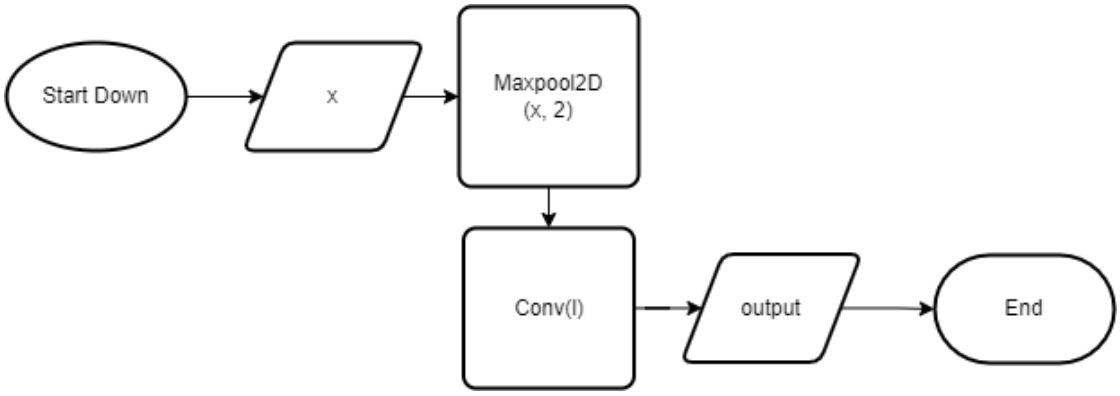

Figure 4.5: Functional schme of the downsampling block

Lastly is necessary to implement the block wich perform the upsampling of image to obtain as final output the desired image. The Upsampling function receives as input the output tensor from the previous block and the output tensor of the downsampling block at the corresponding depth level. Also receives as input the channel's number of the output and input tensors, and the number of convolution to

perform in the straight part of the U. A transposed convolution is then performed with a 2×2 kernel dimension in order to duplicate the image dimension, until the output size does not return to the original dimension.

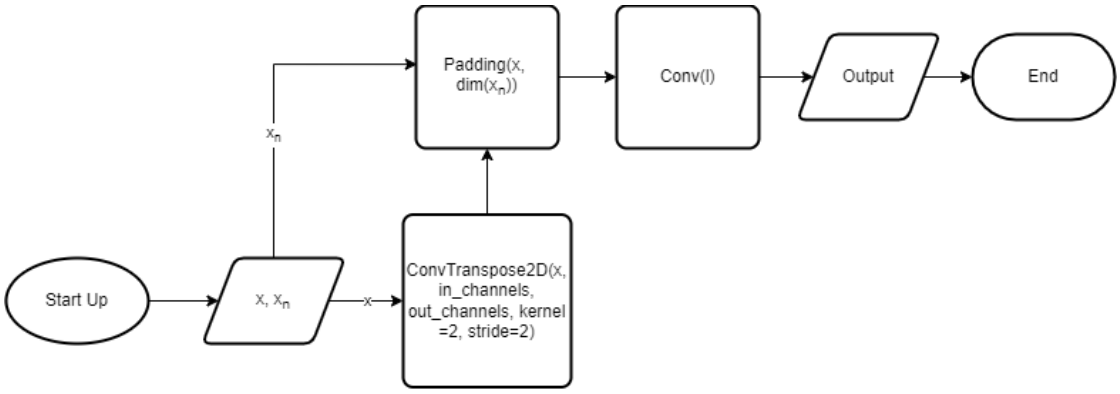

Figure 4.6: Functional schme of the upsampling block

As demonstrated below, the more efficient trained U-Net was the simplest, that is the U-Net with only one deepth layer. So we've traied to train also some Residual Networks (ResNet), illustreted in section 2.4, that is another type of architecture for convolutional neural network but simplest than U-Net.

As previously stated, the structure of these neural networks is notably simple. They all begin with an initial convolution, which increases the number of filters in the image without reducing the original image dimensions. This differs from the approach taken with the U-Net, where the number of filters was reduced. Following this, we have the residual blocks, which perform a specific number of convolutions (dependent on the size of the residual block) with skip connections, as illustrated in Figure 4.7.

4.2.1 Conditional generative adversarial netwok

The trained cGAN employs a U-Net with a single convolutional layer as its generator. This network was identified as the optimal choice for predicting jet flow, and the objective of the cGAN was to enhance the quality of the predicted velocity field, thereby obtaining more precise velocity values.

In this model, the discriminator was of the convolutional variety. The initial convolutional layer, with a kernel size of 2, was designed to capture local patterns in the input data. Subsequently, the model incorporated a 2D batch normalization layer. This batch normalization layer was instrumental in stabilizing and accelerating the training process by normalizing the activations of the previous layer. Finally, a LeakyReLU activation function was applied. The LeakyReLU activation function introduced a small slope for negative values, which mitigated the issue of

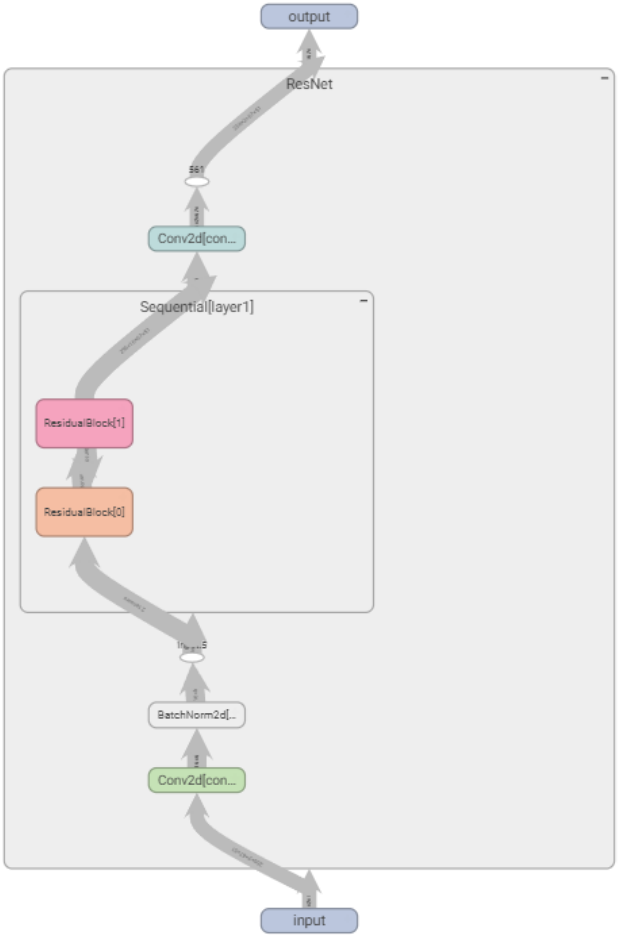

Figure 4.7: ResNet block diagram

dying neurons and ensured that the model could learn more effectively from the data.

4.3 SHAP algorithm

Neural networks, in particular deep neural networks, are often considered as "black boxes". This metaphor is attributed to several factors inherent to the structure of neural networks:

- **Complexity:** deep neural networks are composed of many interconnected artificial neurons organised in multiple layers. The intricate weave of these connections makes it difficult to trace the flow of information within the network and understand how it arrives at its decisions;

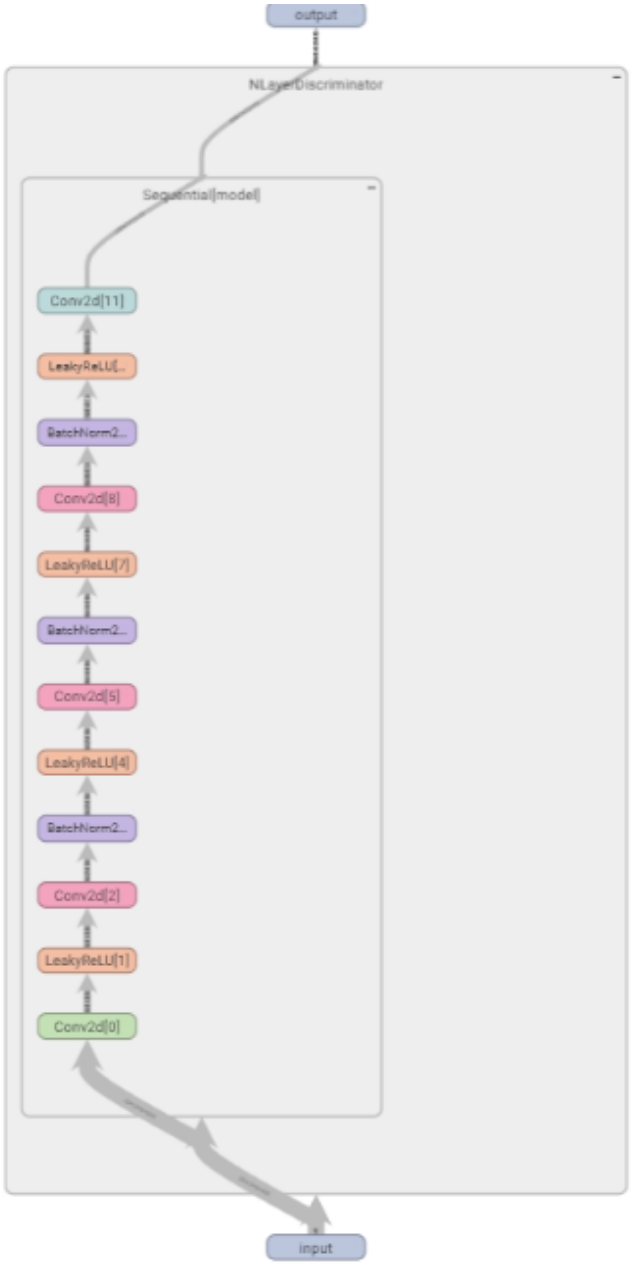

Figure 4.8: Discriminator block diagram

- **Lack of interpretability:** in contrast to traditional statistical models, neural networks do not provide explanations for their decisions. This means that it is not possible to ascertain with certainty the characteristics of the input that led the network to classify it in a certain category or generate a certain output. This makes verification and validation of the results challenging. Consequently,

the results provided by these systems are not as reliable as they could be;

- **Training data:** Neural networks learn from training data, which can be affected by distortions and errors of various kinds. If the data are not accurate or representative of reality, the neural network may learn incorrect patterns and generate unreliable results;
- **Lack of control:** once a neural network has been trained, it is challenging to modify its behaviour. The parameters of the network are optimised during the learning process, and subsequent alterations to these parameters can have unpredictable effects that are difficult to control;
- **Debugging problems:** the identification of the source of an error in a neural network is often a challenging endeavour. The intricate nature of the network makes it challenging to identify the source of the error and to devise an effective corrective measure.

Nevertheless, research on interpretable artificial intelligence is developing novel techniques to enhance the interpretability and reliability of neural networks, thereby overcoming the current limitations of neural networks. The SHAP (SHapley Additive exPlanations) algorithm represents a tool for explainable artificial intelligence.

The technique in question is that of interpreting the decisions made by a model by assigning each input variable an importance value (Shapley value), which indicates the extent to which that variable contributed to the final result. This method can thus enhance confidence in the model by facilitating an understanding of how the model arrived at its result, or by improving the efficiency of the model.

The Shapley value is a concept from cooperative game theory, named after the American mathematician Lloyd Shapley. It provides a fair distribution of the total gains (or costs) among the players in a coalition based on their individual contributions. This method is particularly useful in situations where the collective effort of a group leads to a certain outcome, and it is necessary to determine how to fairly allocate the resulting benefits or costs among the participants.

To calculate the Shapley value, we consider all possible permutations of the players and determine the marginal contribution of each player to every possible coalition they can join. The Shapley value for each player is then the average of these marginal contributions across all permutations. This ensures that each player is rewarded in proportion to their contribution to the overall success of the coalition.

$$\phi(i, v) = \frac{1}{|N|!} \sum_{\pi \in \Pi_N} v[B(\pi, i) \cup \{i\}] - v[B(\pi, i)] \tag{4.4}$$

where:

- $\phi(i, v)$ is the reward due to the i-th player;

- v is the characteristic function;
- Π_N is the set of all permutations of the elements N;
- $B(\pi, i)$ is the set of players preceding the i-th player in the considered permutation.

This formula takes into account the different ways players can contribute to various coalitions, ensuring a fair and equitable distribution of the total value generated by the coalition.

The Shapley value has applications in various fields, including economics, political science, and network theory. It is used to solve problems related to cost-sharing, resource allocation, and voting power, among others. By providing a systematic and fair method for distributing gains, the Shapley value helps to ensure cooperation and stability within groups.

An analogue formulation for the Equation 4.4 could be:

$$\phi(i, v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!\,(|N| - |S| - 1)!}{|N|!} \left[v\left(S \cup \{i\}\right) - v\left(S\right)\right] \tag{4.5}$$

where:

- N is the set of coalition participants;
- S is the set of permutations that don't contain i;
- $\left[v\left(S \cup \{i\}\right) - v\left(S\right)\right]$ is the term of the marginal contributions of the $i - th$ partecipant;
- $\frac{|S|!(|N|-|S|-1)!}{|N|!}$ is the probability that the $i - th$ partecipant will be part of the coalition S.

This formulation is beneficial in facilitating a more comprehensive comprehension of the utilisation of the Shap algorithm for XAI.

The objective of applying this algorithm to neural networks for the prediction of turbulent flows is to identify the physical phenomena that govern the fluid motion under analysis and to assign weights to them. The network will also return the importance that each of the physical phenomena considered had in the result generated by the network[28].

The SHAP value of each event, designated as Φ_e, is calculated by summing the contributions of each turbulent structure that is related to the e-th event. The greater the SHAP value attributed to a given phenomenon, the more influential it is in the reconstruction of the turbulent flow.

In the case study, with the Equation 4.5 as a point of reference, the function v will be taken to represent the neural network, and the term $\left[v\left(S \cup \{e\}\right) - v\left(S\right)\right]$

will be understood to represent the contribution of the event in question. The subset S of events under consideration will be contrasted with the total set of events N.

The precise calculation of SHAP values is not immediately apparent; therefore, the typical approach is to approximate them with the kernel-SHAP method, which integrates SHAP values with the Local Interpretable Model-agnostic Explanations (LIME) method:

$$\mathcal{L}(f, g, \pi_x) = \sum_{q' \in \mathcal{Q}} \left[f\left(h_x(q')\right) - g(q') \right]^2 \pi_x(q') \tag{4.6}$$

$$\pi_x(q') = \frac{|Q| - 1}{\binom{|Q|}{|q'|} |q'| (|Q| - |q'|)} \tag{4.7}$$

where $|q'|$ is the number of structures, h_x represents a mask function that tranforms the binary space of q' into the space of the input of the model. The LIME equation has been resolved by means of a linear regression, with the resulting error being $(f - g)^2 \mathcal{O}(10^{-7})$.

The LIME method involves the interpretation of individual model predictions based on the approximation of the model in question at a local level, situated in proximity to the given prediction. It has been demonstrated by Riberio et. al. [29] that this method can provide an explanation for generative models with a precision rate exceeding 90%.

In the work of A. Cremades, S. Hoyas, R. Deshpande et. al. [28] neural network was constructed to predict flow in a channel, taking into account the phenomena of sweep, inward interaction, ejection and outward interaction. In this work, XAI has been used to quantify the importance of these phenomena, in particular it has been seen that in flows in a channel with fully developed turbulence, the most important structures governing the fluid motion governing the fluid motion are ejection and sweep, this coincides with the current knowledge of the physics of flow, therefore one can consider the the result of the U-net. So on the bases of this result our purpose is to exploit the capabilities of this algorithm to capture information on the physics of turbulent flows, in this case we use a dataset of PIV images of an axisymmetric jet flow.

In order to compute the Shap values, the Kernel-SHAP method, implemented in the SHAP library [30] , was employed. Kernel SHAP combines Shapley values from game theory with linear regression to compute feature importance, ensuring local accuracy, missingness, and consistency.

The method circumvents the heuristic choices inherent to LIME through the utilisation of a bespoke loss function, a weighted kernel, and a regularisation term, thereby facilitating the recovery of Shapley values.

4.3.1 Application of the Kernel-SHAP algorithm

The first step to apply the kernel-SHAP algorithm to study the importance of turbulent structure in the prediction of the motion field is to identify them. In this study, we have identified the turbulent structures in each photogram by searching the vorticity field for zones with a value greater than a specified threshold. It is important to note that the choice of threshold value can influence the results obtained, as different structures may be identified if the value is altered.

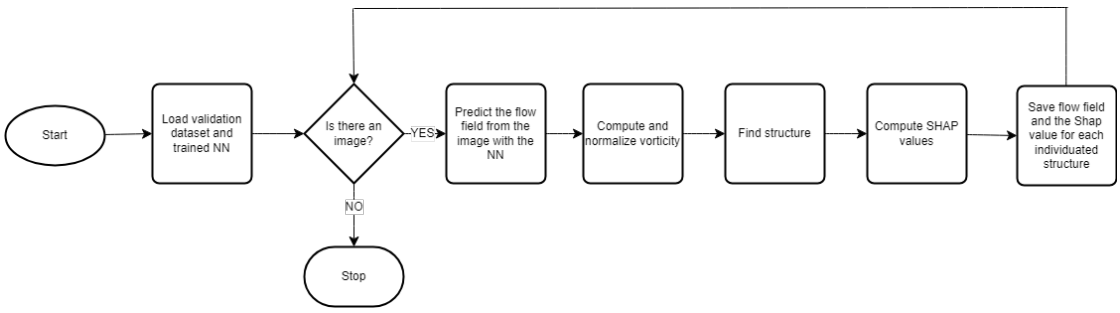

Figure 4.9: Flow chart of the implemented algorithm to apply Kernel-SHAP at our model

The flowchart in Figure 4.9 illustrates the methodology employed to apply the Kernel-SHAP technique to our model. The process entails loading the optimal model, which has been trained for flow prediction, and the validation dataset. Then, for each photogram we compute the vorticity field that in this case is defined as shown in Equation 4.1.

The vorticity values must be normalised for an accurate and

$$\omega^* = \frac{\omega D}{U_e} \tag{4.8}$$

where D is the diameter of of the jet outlet and U_e is the exit velocity of the jet.

Next the structures in the photogram are individuated setting an opportune threshold with functions in the measure class in the skimage library [31]. This process yields a mask image with a *True* value where the normalized vorticity exceeds the specified threshold and a *False* value in all other instances. So this mask is passed to the labels function that enumerate areas with True. An example of he outcome of this phase is illustrated in Figure 4.10 for purposes of illustration. In this case the algorithm has individuated 6 structures.

Following the completion of this procedure, the Shapley value for each structure can be calculated. In order to achieve this, it is necessary to pass the image in question, excluding the i-th structure, to the network. In order to achieve this, the structure's portion of the field has been replaced with the mean field. Subsequently,

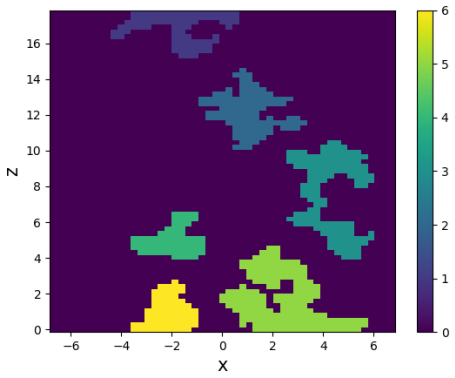

Figure 4.10: Example of individuated structure in a snapshot

the novel field is conveyed as an input to the model, thereby enabling it to undertake a prediction. For example in Figure 4.11 the field of velocity fluctuations without the first structure identified in the snapshot, i.e. the region labeled 1 in Figure 4.10.

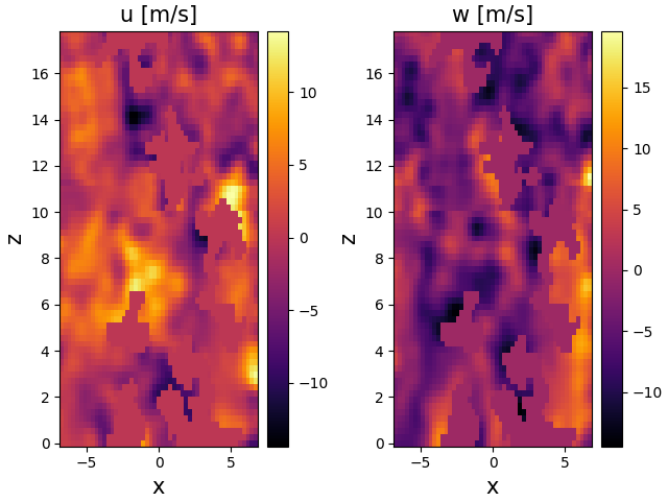

Figure 4.11: Example of reconstructed motion field after structure removal

Ultimately, a comparison of the network output for the true field with the output for the field without the structure, using a kernel-SHAP explainer, allows the SHAP value of the structure to be computed, thus providing insight into its contribution to the prediction of the motion field.

Chapter 5

Results

In this pivotal chapter, we delve into the comprehensive analysis of the results obtained from the the hyperparametric study during the training of neural networks, alongside a detailed examination of the SHAP analysis and statistical scrutiny of turbulent structures. The neural networks, trained with precision, have yielded insights that are critical in understanding the complex dynamics at play. The SHAP analysis, a novel approach in our study, has provided an interpretative power to the neural network's predictions, offering a new dimension to the data's explanatory depth.

5.1 Training of the neural networks

The first step, after building the neural networks as described in section 4.2 is to train the networks. So, as we discussed earlier, we trained six CNNs with U-Net architecture and three with the ResNet architecture.

The training phase involved randomly splitting the dataset, with 60% of the total images designated for training, testing, and validation. The training process was conducted with a batch size of 256 images on a Nvidia GeForce RTX 3080® with 12 GB of memory. The high batch size was possible given the relatively small image dimensions. As seen in section 2.5 the batch size parameter refers to the number of training examples used in one iteration of model training, its choice can have a significant impact on the training and consequently on the final model performance.

It is crucial to evaluate the network's performance and the actual trend of weight optimisation during training in order to ascertain the network's ability to process images that differ from those in the training dataset. Furthermore, error computation on images that have not been previously encountered is essential for assessing the network's generalisation capabilities. This is the reason why the

dataset was divided into a training and a validation set.

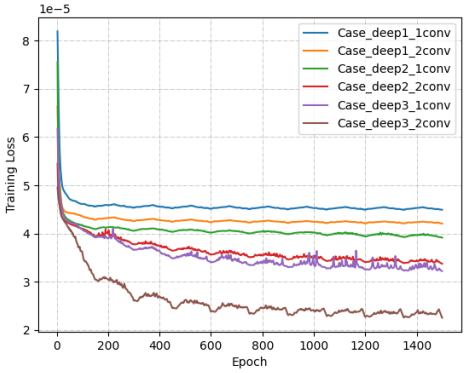

Figure 5.1: Trend of the loss function value during the training of the U-Nets

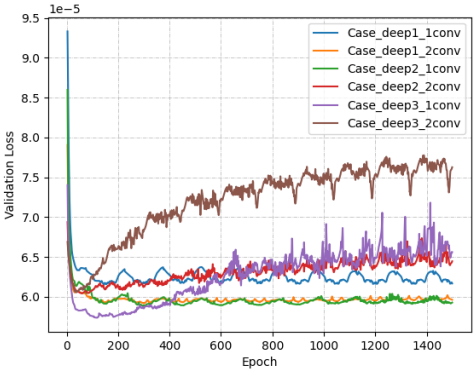

Figure 5.2: Trend of the loss function value during the validation of the U-Nets

Figure 5.1 illustrates the evolution of the loss value throughout the training epoch. It can be observed that the final loss value is relatively low for all networks. However, it is notable that as the network complexity (and consequently the number of learnable parameters) increases, the final loss value decreases. This suggests that the network with three depth layers and two convolutions may be the most performant. Nevertheless, as previously stated, it is essential to examine the validation dataset to gain a deeper understanding of the network's behaviour.

Upon examination of the validation loss value trend during the training process, as illustrated in Figure 5.2, it becomes evident that the circumstances are in fact quite distinct. In particular, it can be observed that the second and third networks exhibit a loss value that is approximately equivalent, while the first network displays a slightly higher loss value. In all other networks, it can be seen that the loss value tends to increase more or less rapidly at a specific epoch number. This is likely due to the networks experiencing the overfitting phenomenon, which prevents them from accurately predicting the motion field. The trained network has therefore been saved at the epoch at which the minimum value of the validation loss was observed.

A comprehensive analysis of the results obtained from all the networks, including an assessment of the quality of the generated images (see Appendix A), reveals that the U-Net with a single layer depth and two convolutions exhibits superior performance. This is why we've tried to train ResNet as anticipated in section 2.3

A review of the training loss value trend in Figure 5.3 reveals that the loss value at the conclusion of the training process for the ResNets is approximately 4×10^{-5}, which is comparable to that of the U-Net. In this instance, the overfitting

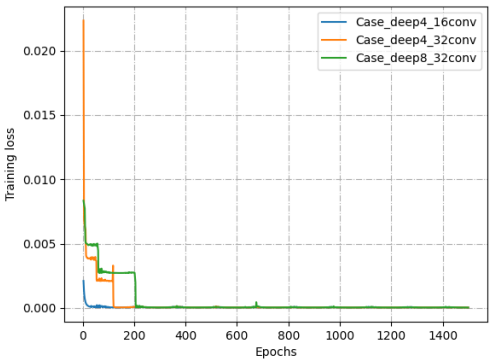

Figure 5.3: Trend of the loss function value during the training of the ResNet

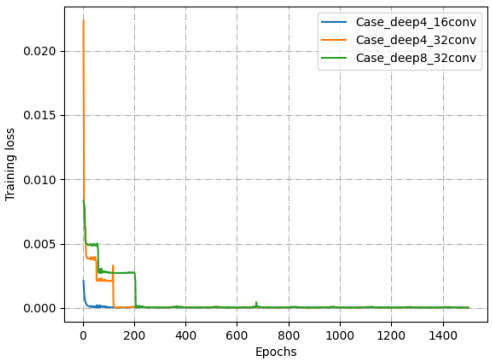

Figure 5.4: Trend of the loss function value during the validation of the ResNet

phenomenon cannot be observed. However, as illustrated in section A.2, the output images exhibit a lack of definition when compared to those produced by the U-Net model.

The trained cGAN has the potential to yield favourable outcomes, but its full potential will only be realised through further refinement. It is evident that the training process of our cGAN is not yet optimised, as evidenced by the observations presented in Figure 5.5 and Figure 5.6. There is a sudden increase in the loss function for the generator, followed by a shift in the overall trend for both the generator and the discriminator. This indicates the necessity for modifications to the training hyperparameters, such as the learning rate or the λ, as illustrated in Equation 2.9.

Furthermore, during the validation phase, a comparable surge in the loss function is observed as showed in Figure 5.7, resulting in a less accurate estimation of velocity values. This presents a significant challenge when attempting to derive physical insights from the predicted data.

As illustrated in Figure 5.9, the predicted field accurately represents the actual flow field, as evidenced by the comparison with the real field in Figure 5.8. However, the predicted image has a lower definition than the actual image. Moreover, the percentage error remains too high to permit its use in the physics analysis of turbulent structures.

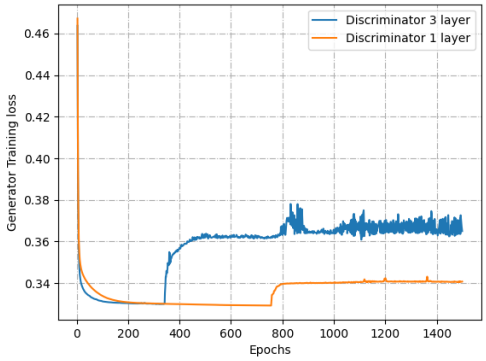

Figure 5.5: Trend of the loss function of the generator value during the training of the cGAN

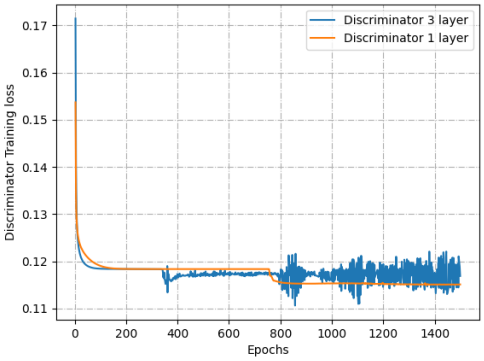

Figure 5.6: Trend of the loss function of the discriminator value during the training of the cGAN

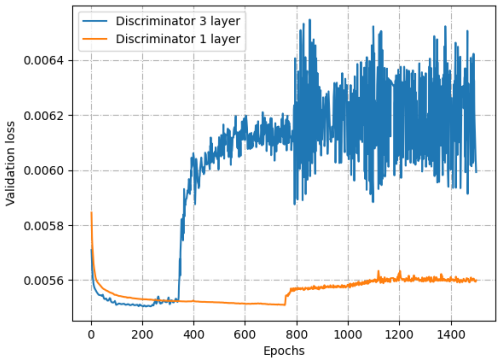

Figure 5.7: Trend of the loss function during the validation of the cgan

5.2 Sensitivity analysis of the implemented algorithm for the detection of turbulent structures

As seen in subsection 4.3.1, the implemented akgorithm for the identification of the turbulent structures in the jet flow was very sensitive to the selected threshold for the normalized vorticity ω^*. Accordingly, a sensitivity analysis of the algorithm at the vorticity threshold was deemed essential.

A Python script is used to load the validation dataset employed in the training process and the most optimised trained network. Each snapshot is then passed

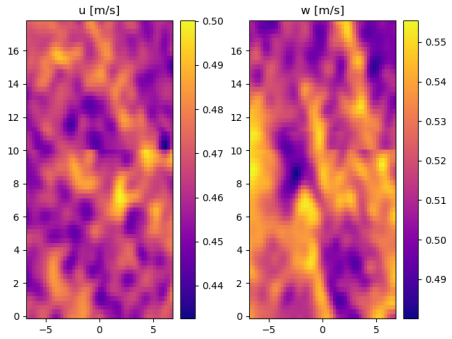

Figure 5.8: Real velocity field, on the left there is the radial component, at the right the streamwise component

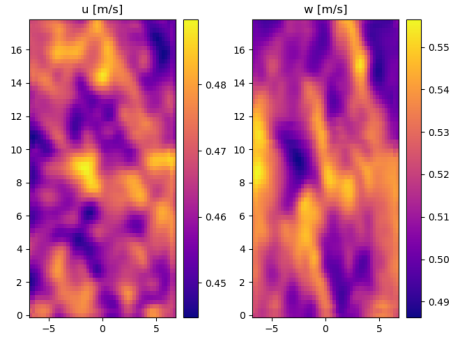

Figure 5.9: Predicted velocity field from the cGAN, on the left there is the radial component, at the right the streamwise component

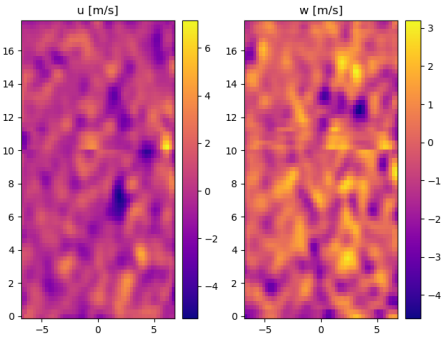

Figure 5.10: Percerntage error on the velocity prediction, at the right the streamwise component

as input to the network in order to obtain the consecutive snapshot from which the vorticity is computed and normalised. Finally, the turbulent structures are identified.

In Figure 5.11 is shown the trend of the number of individuated turbulent structures in all the snapshots with the value of the threshold on the normalised vorticity.

With a mean of about six individuated structures for each photogram, the threshold value with more absolute individuated turbulent structures was $\omega^* = 0.354$.

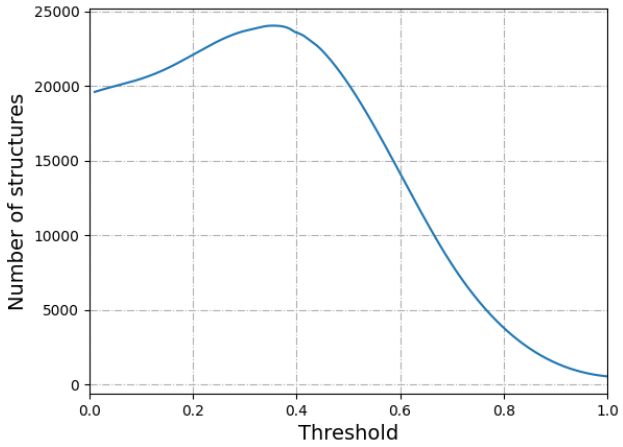

Figure 5.11: Trend of the number of structures finded by the algoithm with the normalized vorticity threshold value

5.3 Statistical analysis of the individuated turbulent structures

Once the threshold for the normalised vorticity has been selected, it is possible to analyse the identified structures using the kernel-SHAP algorithm, as detailed in subsection 4.3.1.

A plot of the Shapley values against the areas of the turbulent structures showed in Figure 5.12 reveals a trivial result: the more important structures are those with the largest areas. This is an obvious consequence of the fact that the greater the area of a turbulent structure, the greater the amount of information deleted from the data input. Consequently, the network will produce a worse result, and the Shapley value will increase. Furthermore, an examination of the physical information reveals no noteworthy insights when considering the relationship between the Shapley values and the integral mean of the normalised vorticity on a structure's area showed in Figure 5.13.

For these reasons, the Shapley values φ computed by the kernel-Shap algorithm implemented in the shap library [30] was normalised as following:

$$\Phi = \frac{|\varphi/A|}{max(|\varphi/A|)} \tag{5.1}$$

where A is the area of the finded turbulent structure. The objective of this standardisation was to establish a methodology for the decoupling of the value and importance of a given structure from its size. Subsequently, the values were scaled

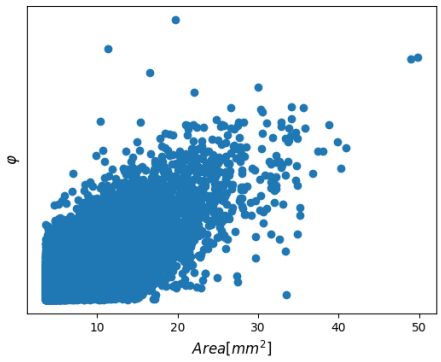

Figure 5.12: Shapley values against the areas of the turbulent structures

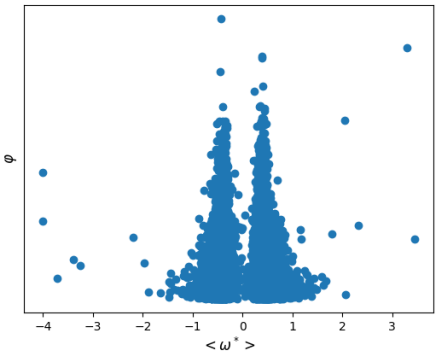

Figure 5.13: Shapley values against the the integral mean of the normalised vorticity on a structure's area

for convenience, with all values subsequently expressed as a proportion between 0 and 1.

From the analysis with the kernel-SHAP algorithm of finded turbulent structure, for each structure we have saved, the photogram and structure's indexes, the velocity field of the structure, the position of the centroid of the structure in the phortogram and the Shap value. Upon completion of the analysis with the kernel-SHAP algorithm, we shall proceed with the statistical analysis of the individuated structures.

5.3.1 Geometry of the individuated structures

The first our curiosity was to see the geometry of the structures. In frist istance we've done a sample check to see the shape of the structures to see that there wasn't structures with strange geometry like that one seen in Figure 4.10, in that case we should have added some filters in the algorithm that in the algorithm that delete too small structures and small holes in the structures in order to ensure that any that are present are joined by a significant number of pixels, in order to ensure that the structures are physically sound.

The areas of the structures is very variagate, but to avoid to have very small structures we've added a filter on the minimum number of pixels of the individuated structures. As will be seen in subsequent graphs there are very small structures characterised by an area of just a few squared millimeters, meanwhile the The largest structure occupies approximately half of the total area of the frame.

As exibith in Figure 5.14 where is the power density function of the areas of the

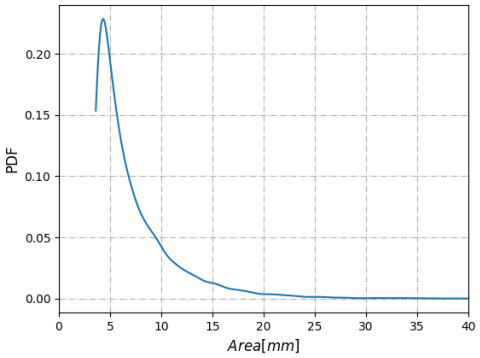

Figure 5.14: Statistical distribution of the area of all identified structures

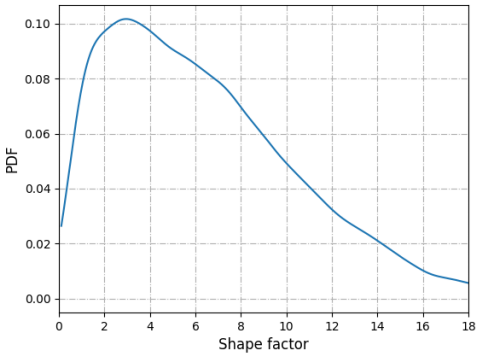

Figure 5.15: Probability density function of the shape factor of the turbulent structures

structures, the majority of the identified structures exhibit relatively limited area.

In Figure 5.15 is also illustrated the probability denity function of the shape factor of the individuatedstructures defined as $length/width$ where length is the dimension in the streamwise direction, and width is the dimension in the radial one. We can see that the structures identified are elongated in the streamwise direction, and this makes sense with the physics knowledge. In fact it is established that the large-scale structures that emerge in the high-momentum fluid trail the vortical structures.

5.3.2 Shapley values

In the course of our investigation, we examined the normalised Shapley values as illustrated in Equation 5.1 for the reason explained before.

As illustrated in Figure 5.16, the larger structure exhibits a relatively low normalised Shap value. However, it should be noted that the normalisation was conducted with reference to the structure's area, which aligns precisely with our intended objective. Furthermore, it is evident that there are additional structures that may be perceived as more significant, despite their smaller size. This is potentially attributed to the influence of vorticity value, which is likely to be higher in smaller structures. Consequently, it can be inferred that the presence of greater vorticity within a structure is indicative of a higher degree of physical information content.

It is also important to consider the structure with normalised Shap value $\Phi = 1$. It seems to be the most important structure individuated and it has also a "normal"

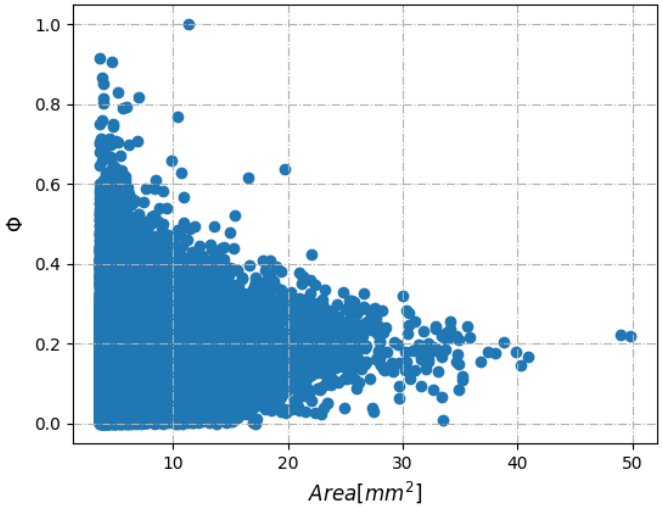

Figure 5.16: Distribution of the normalised Shapley values with the structure's areas

area's value so it can't influence the normalisation. The problem with this structure is the high values of vorticity inside the structure, in fact we shall see that the $< \omega^* >$ has an anomalous value.

It is similarly crucial to contemplate the structure with a normalised Shap value of 1. It appears to be the most significant structure identified, and it also exhibits a typical area value, indicating that it cannot influence the normalisation process. The issue with this structure is the elevated values of vorticity within the structure itself. Indeed, it will be demonstrated that the $< \omega^* >$ has an anomalous value. The structure with a more effectively high Shap value is that with a value of approximately $\Phi = 0.8$, as some of these structures are characterised by a normal value of areas and $< \omega^* >$.

Figure 5.17 illustrates the probability density function of the normalised Shap values, represented by the function Φ. The majority of the structures exhibit a value within the range of 0 to 0.5. Only a few structures display a normalised Shap value exceeding 0.5, although a number of these cases are characterised by the presence of some form of error, which will be discussed in greater detail later in the presentation of appropiate results.

Figure 5.18 depicts the scatter graph of the integral mean of the normalised vorticity, represented by the variable $< \omega^* >$, on the surface of the turbulent structure in relation to the normalised Shapley values.

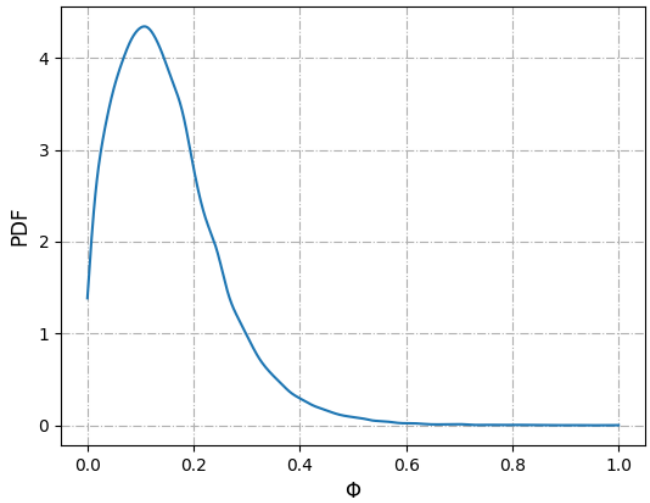

Figure 5.17: Probability density function of the normalised Shapley value Φ of the individuated structures

$$< \omega^* >= \frac{1}{A} \int_A \omega^* d\sigma \tag{5.2}$$

From this graph, it is possible to discern the rationale behind the tendency for Shap values to be excessively high for for some turbulent structures. It is evident that there are several outliers, both in the case of the $< \omega^* >$ value and the value of the function itself. The reason for their presence is due to the occurrence of spikes in velocity values at some points within the frames of the dataset.

Subsequently, the $< \omega^* >$ and Φ data were subjected to a *k-means cluster* analysis in order to ascertain the degree of similarity between the identified structures.

The K-Means algorithm is a clustering algorithm employed in the field of data analysis. This algorithm clusters data by separating it into n groups of equal variance, thereby minimising a criterion known as the sum of squares of inertia. In order for this algorithm to function correctly, the desired number of clusters must be specified. It is particularly well suited to a large number of samples and has been used in a wide range of applications in many different fields. From a dataset $\mathcal{D}$ the algorithm will randomly select k data objects as the initial centroid of the clusters. Then the Euclidean distance between each data object d_i and the k cluster's centroid is computed:

$$d(x_i, y_i) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{5.3}$$

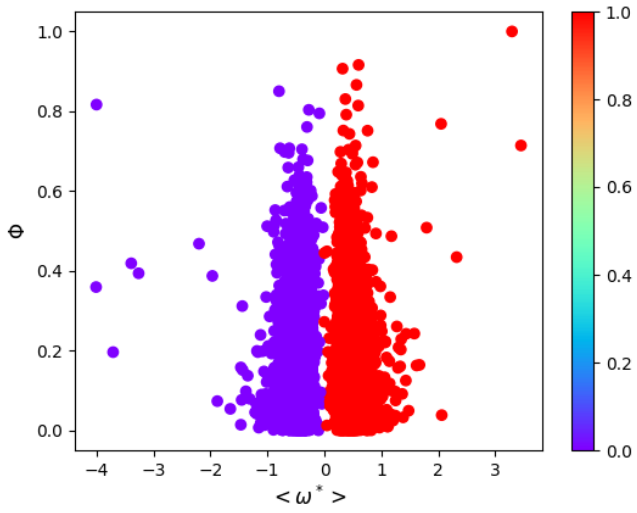

Figure 5.18: Cluster K-Means of the individuated structures

Now, on the basis on this distance the algorithm assign the d_i object at the nearest cluster. For each cluster the centroid must be recalculated. The process to compute the distance of each point from centroid and to associate it at the neraest cluster will will be repeated until the position of the centroids no longer varies.[32]

In this case, based on what we see on the graph, we thought it appropriate to to find three clusters. Obviously we were expecting exactly what we've got, there are two clusters with the $< \omega^* >$ greater and less than zero and it seems to be very symmetric respect to the zero value. In Figure 5.18, the colormap of the point represent the cluster identification, the firt cluster in purple is that one with $< \omega^* >$ less than zero and in red wi can see the cluster with positive values of vorticity's mean intgral.

Following the cluster analysis, the conditioned structures of each cluster were examined to ascertain whether there was a similarity in the structures.

The conditioned structures are the average of all the structures, in this case belongings the $i - th$ cluster. So for each indivituated cluster we have summed the structures' field making their centers match, and then dividing by the number of the structures in the cluster. It is evident that the velocity and the physical plane of the conditioned structures are not realistic, and thus the values are not realistic. Consequently, it is only possible to ascertain whether the structures in the cluster are coherent and to identify what they are.

Figure 5.19 illustrates the conditioned structures of the positive cluster, which are highlighted in red Figure 5.18. It can be observed that the average of these

Algorithm 3 Algorithm implemented to sum the turbulent structures to obtain the conditional mean

procedure SUM_WITH_PADD(x, y)
    ▷ The function pads the arrays to the same size, ensuring that their centres are aligned, and then sums the arrays.
    ▷ x, y are the arrays to be summed
    $target\_shape \leftarrow max(x.shape, y.shape)$
    $padded\_array1 \leftarrow pad\_to\_same\_size(x, max\_shape)$
    $padded\_array2 \leftarrow pad\_to\_same\_size(y, max\_shape)$
    $summed\_array \leftarrow padded\_array1 + padded\_array2$
    return $summed\_array$
end procedure

structures yielded a notable outcome. This result can be interpreted as an indication that the structures within the cluster are coherent with one another.

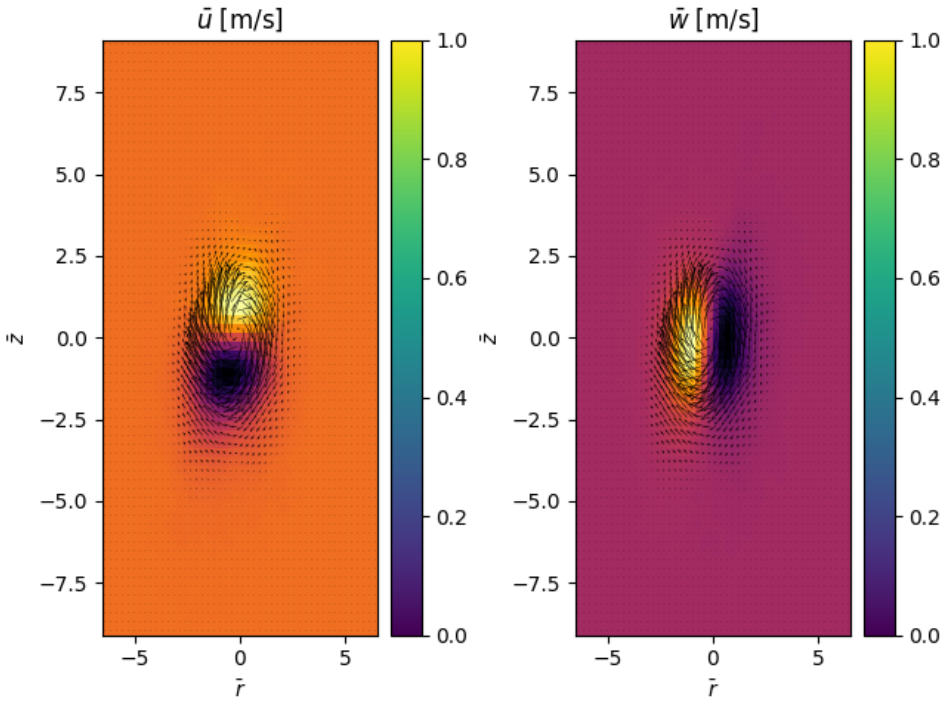

Figure 5.19: Conditioned structures of the red cluster

Similarly, the same average can be performed on the negative cluster, indicated in purple in Figure 5.18 and shown in Figure 5.20. As with the previous cluster, the structures exhibit coherence, and an anti-symmetry is evident due to the opposite

sign of the integral mean of the vorticity. This results in the structures being classified as counterclockwise vortices, in contrast to the first cluster, which was identified as a clockwise vortex.

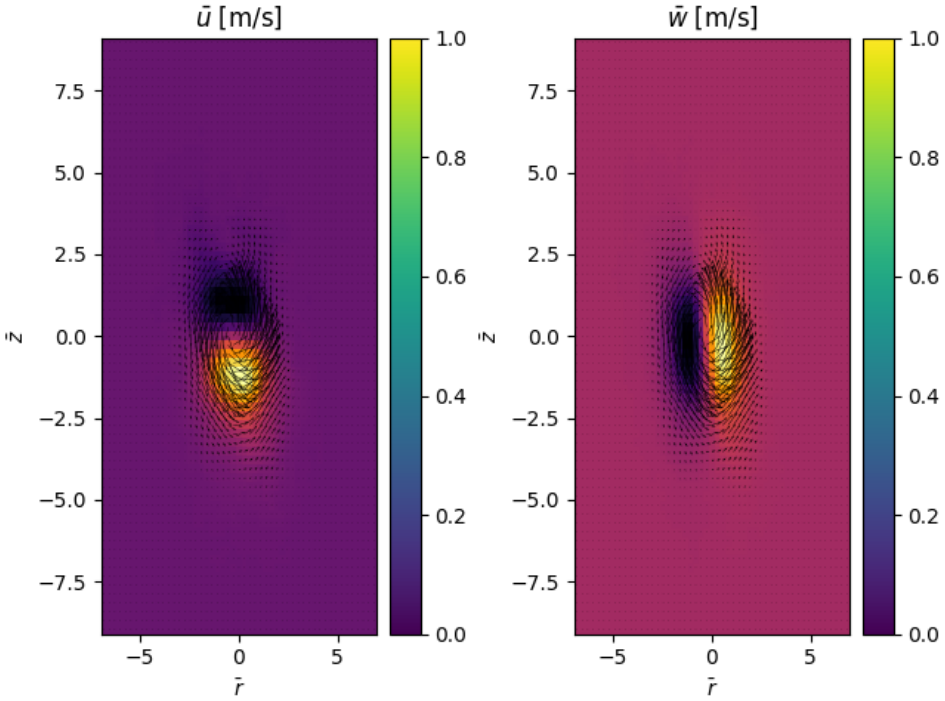

Figure 5.20: Conditioned structures of the purple cluster

5.3.3 Physical quantities

As illustrated in Figure 5.18, the PDF of $< \omega^* >$ exhibits two peaks. These peaks are associated with the clusters of positive and negative structures, respectively. The aforementioned peaks display a high degree of symmetry; however, the PDF's highest peak suggests a little prevalence of negative structures.

From the joint PDF of $< \omega^* >$ and normalised shap values Φ showed in Figure 5.22, is possible to see that both the positive and negative structures have the same importance, the central cluster should be more invistigated because these structures has also the same importance of these but as we've seen before from our investigation it seems to be very different.

Figure 5.25 illustrates the trend of the maximum streamwise velocity on the axis of the jet. The points represent the experimental values of the maximum streamwise time-averaged velocity on the axis of the jet, while the straight line represents their interpolation with the empirical law presented in Equation 3.3. It can be observed

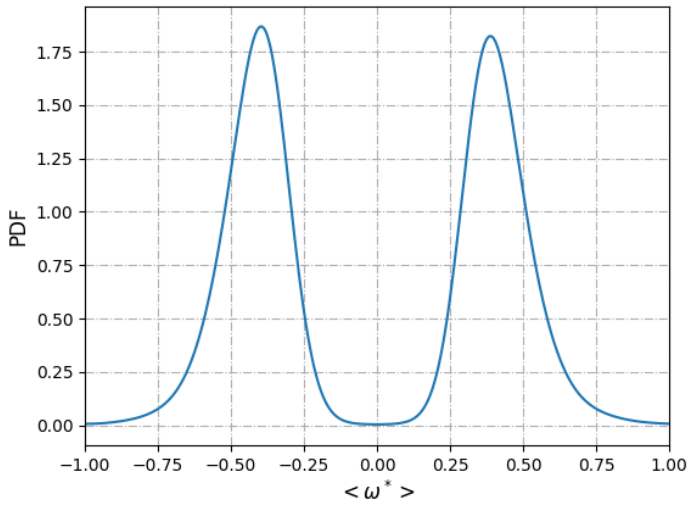

Figure 5.21: Probability density function of $< \omega^* >$ of the individuated structures

that the time-averaged streamwise velocity decreases from approximately 0.6 times the jet velocity to approximately 0.45. This information is of interest in that it allows us to ascertain whether the velocity values displayed in the graph presented in Figure 5.23 were significant. Given that they are comparable, we can consider the results obtained to be reliable.

The symmetry you see in all graph demonstrates that clockwise and counter clockwise vortex are similar wich is expected from symmetry consideration thus lends confidence to the results from the kernel-SHAP analysis.

From the joint probability density function of the normalised Shap values Φ and the integral mean of the normalised vorticity in Figure 5.22 we can see once again the simmetry in the individuated structures. The positive and negative structures are of equal significance and occur at a similar frequency. Meanwhile, the structures with $< \omega^* >$ around zero are distinguished by normalised Shap values within a comparable range to the first, but are less probable than these.

5.3.4 Dissipation function

Moreover, an analysis was undertaken of the dissipation function. In the initial phase of the process, the dissipation function was calculated for each of the snapshots. Subsequently, the dissipation was calculated for each individualised structure, solely within the confines of the structure itself.

The integral mean of the dissipation function on the structure's surface was

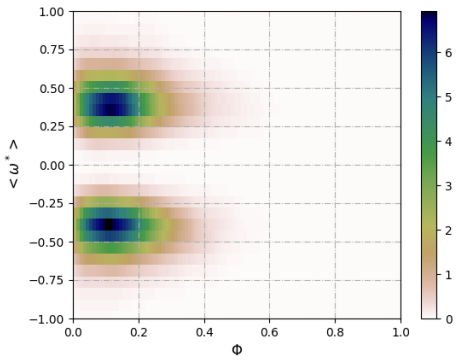

Figure 5.22: Joint PDF of normalised shap values Φ and $< \omega^* >$

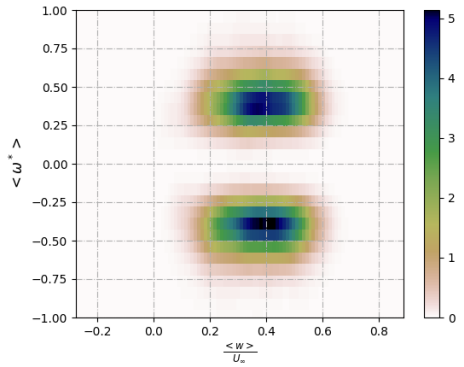

Figure 5.23: Joint PDF of velocity's streamwise component normalised with the jet's velocty U_∞, and $< \omega^* >$

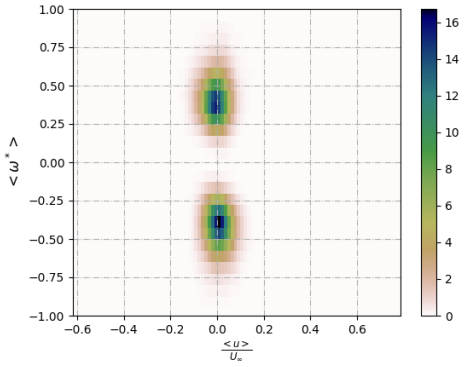

Figure 5.24: Joint PDF of velocity's radial component normalised with the jet's velocty U_∞, and $< \omega^* >$

normalized with the integral mean of the entire dissipation function's field. Figure 5.26 show the probability density function of the normalized integral mean of the dissipation function. We can see that the most part of the individuated structures are responsible of all the energy dissipation in the field.

This is in accordance with the physics knowledge that small-scale vortices are the primary agents of turbulent energy dissipation. As previously observed in chapter 3, the typical dimensions of these vortices are approximately five times the Kolmogorov length scale. However, due to the insufficient spatial resolution of the images, it was not feasible to discern these scales in this particular dataset.

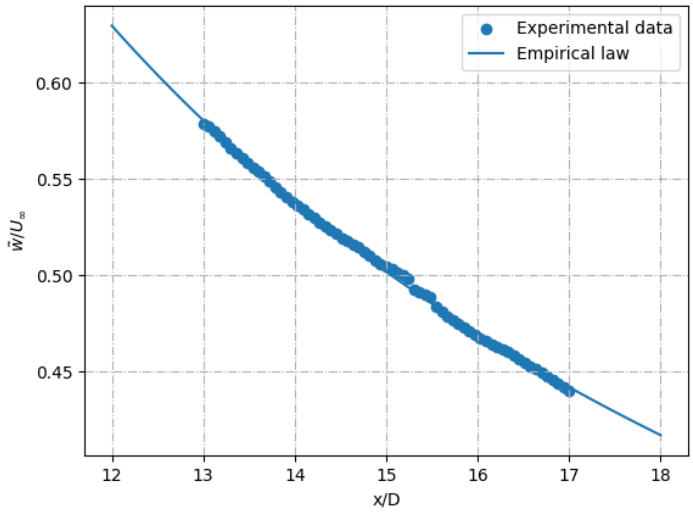

Figure 5.25: Trend of the time-averaged streamwise velocity on the axis of the jet

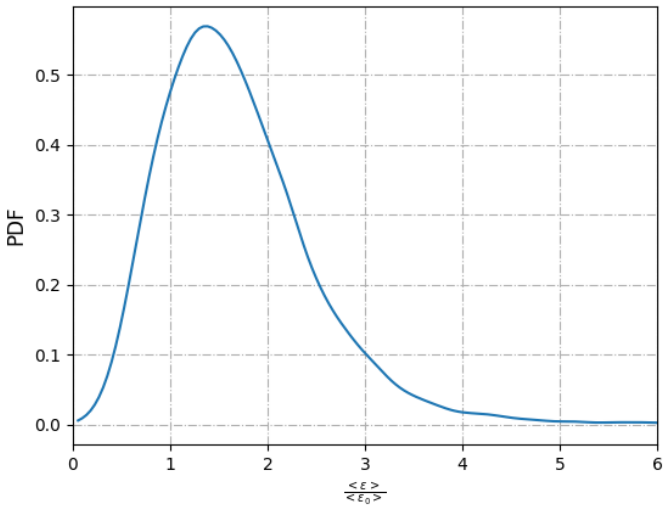

Figure 5.26: Probability density function of the integral mean of the dissipation function

In Figure 5.27, the joint PDF of the normalised Shapley values is presented with contour lines of a darker shade, which evidence that the higher the Shapley values, the greater is the dissipation within the structure. Therefore, this structure

is likely to play an important role in the physics of the jet flow.

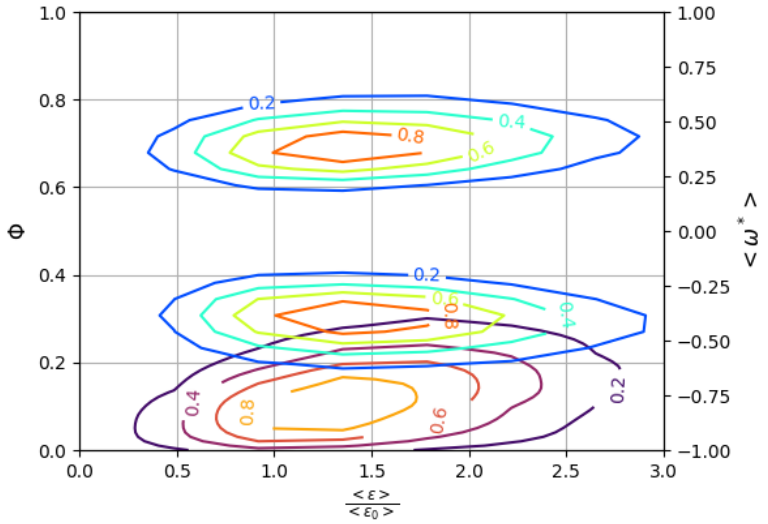

Figure 5.27: Joint PDF of the normalised dissipation function and the normalised Shapley values is illustrated in the dark lines of the contour. Joint PDF of the normalised dissipation function and the integral mean of the normalised vorticity is displayed in the light contour lines.

Chapter 6

Conclusion

In this study, we have explored the intricate dynamics of turbulent structures through the lens of Explainable Artificial Intelligence (XAI) using the SHAP (SHapley Additive exPlanations) algorithm. Our analysis has provided valuable insights into the study of behavior and characteristics of turbulence, which are often complex and difficult to interpret, through the XAI lens.

By leveraging the SHAP algorithm, we were able to demystify the underlying mechanisms driving turbulent phenomena. This approach not only enhanced our understanding of turbulence but also demonstrated the potential of XAI techniques in making complex models more interpretable and transparent.

Our findings highlight several key points:

- **Enhanced Interpretability:** The SHAP algorithm effectively elucidated the contributions of various factors to the turbulent structures, offering a clearer picture of their formation and evolution.
- **Model Transparency:** The application of SHAP in this context underscored the importance of transparency in AI models, particularly in fields where understanding the model's decision-making process is crucial.
- **Practical Implications:** The insights gained from this analysis can inform the development of more precise predictive models and facilitate the creation of models that can offer novel insights into turbulence dynamics. The analysis has demonstrated that it is feasible to extract more pertinent data on the physics of turbulent structures, which can be employed to enhance understanding of physical phenomena in a turbulent flow .

In conclusion, the integration of SHAP with turbulence analysis represents a significant step forward in the application of XAI for the understanding on the physical phenomena that drive turbulent flows. This study sets a precedent for

future research in applying explainable AI techniques to complex scientific problems. As we continue to refine these methods, the potential for XAI to transform our understanding of intricate phenomena remains vast and promising.

6.1 Future development

The prenested techniques are a recent development in this scientific field, and therefore there is considerable scope for further progress.

The preliminary approach is to refine the cGAN predictions, as the generated images show considerable promise but are not yet fit for purpose due to the high error rate on the velocity values. The underlying theory suggest that these networks have the potential to produce more accurate images. However, further improvements are necessary for their utilisation in physical analysis.

A potentially fruitful avenue for elucidating the utility of XAI in turbulence dynamics is to trace the evolution of turbulent structure over time and to examine how these patterns fluctuate in conjunction with the Shap values. This approach could offer insights into the significance of specific structural patterns over time, and whether their importance fluctuates or remains constant.

Another interesting analysis could be to investigate others methods or algorithms to find turbulent structures in the motion field that could be more interesting and also tell us something more on the turbulence dynamics finding different turbulent structures from these. It could be interesting also to use other dataset, more complete and with also the other motion regions of the axisymmetric jet in order to to ascertain whether the algorithm is capable of detecting structures of varying types.

Appendix A

Analysis of the output image of the implemented neural networks

The implemented networks were subjected to an in-depth analysis, with a particular focus on the hyperparameters. The objective of this analysis was to elucidate the impact of these parameters on the network operation and to identify the optimal configuration, which would facilitate the most accurate prediction of the turbulent flow under consideration. In this context, the term "hyperparameters" refers to the number of convolution layers and the depth of the network.

A.1 U-Net

As previously discussed in section 4.2, CNNs with a U-shaped architecture can have varying depth levels. The inclusion of numerous depth levels in a network increases its complexity, leading to a proportional increase in computational cost. Therefore, it is essential to identify an optimal depth level that accurately predicts the flow while avoiding the overfitting problem.

A total of six networks with a U-Net architecture were trained for the purpose of evaluating their quality:

- One layer deep and one convolution;
- One layer deep and two convoplutions;
- Two layers deep and one convolution;
- Two layers deep and two convolutions;

- Three layers deep and one convolution;
- Three layers deep and two convolutions;

One layer deep and one convolution This network consists of a single convolution followed by an output convolution with a kernel size of 1 and an activation function Sigmoid. The initial convolution layer is responsible for extracting essential features from the input data, while the output convolution layer, with its kernel size of 1, adjusts the number of channels to match the desired output format. The Sigmoid activation function is applied to the output layer to ensure that the predicted values are within a specific range, typically between 0 and 1.

Figure A.5 shows images of one of the motion fields predicted by the network in the last validation epoch and its comparison with the real image in Figure A.4. This comparison provides a clear visual representation of the network's performance, allowing for an assessment of the accuracy and quality of the predicted images. It can be observed that the predicted image is highly accurate, closely resembling the real image. This high level of accuracy indicates that the network is effectively learning and generalizing from the training data.

As evidenced by the image and the data presented in Figure A.6, the average error on the images is relatively low, with isolated instances where the error reaches approximately -7%. This low average error suggests that the network is performing well overall, with only a few instances where the predictions deviate significantly from the true values. In this instance, the overfitting phenomenon cannot be observed, as the number of parameters is sufficiently adequate. Overfitting occurs when a model performs well on the training data but fails to generalize to new, unseen data. The absence of overfitting in this case indicates that the network has a good balance between complexity and generalization.

However, the percentage error illustrated in Figure A.6 is unacceptably high. This high percentage error highlights areas where the network's predictions are not as accurate as desired. Consequently, it is imperative to augment the learnable parameter quantity by incorporating additional convolution or deep layers, with the objective of reducing the error in the predicted image. Adding more layers can increase the network's capacity to learn complex patterns and improve its ability to make accurate predictions.

To address the high percentage error, several strategies can be considered. One approach is to increase the depth of the network by adding more convolutional layers. This can enhance the network's ability to capture intricate details and improve the accuracy of the predictions. Another strategy is to experiment with different hyperparameter settings, such as learning rate, batch size, and the number of epochs, to find the optimal configuration for the network. Additionally, implementing

regularization techniques, such as dropout or weight decay, can help prevent overfitting and improve the model's generalization capabilities.

In summary, while the network's current configuration demonstrates high accuracy in predicting the vector fields, the high percentage error indicates areas for potential improvement. By augmenting the learnable parameter quantity and optimizing the network's architecture and training process, the performance of the network can be enhanced, ensuring it produces high-quality, accurate outputs.

One layer deep two convolutions The network comprises two convolutions in sequence, followed by an output convolution as previously described. This sequence of operations is designed to progressively refine the input data, extracting relevant features and enhancing the network's ability to make accurate predictions. The initial convolutions serve to capture essential patterns and details from the input, while the final output convolution adjusts the data to the desired format.

Figure A.8 illustrates the predicted velocity field, with the radial component displayed on the left and the streamwise component on the right. This visualization provides a clear representation of the network's output, allowing for a detailed comparison with the true velocity field. By examining these components, we can assess the network's performance in capturing the dynamics of the velocity field.

A comparison of the predicted field with the true field in Figure A.7 reveals that the predicted images are highly precise. The close alignment between the predicted and true fields indicates that the network is effectively learning and generalizing from the training data. This precision is crucial for applications that require accurate velocity field predictions, as it ensures that the network's outputs are reliable and can be used for further analysis or decision-making.

Notably, the error on the velocity field, as illustrated in Figure A.9, is reduced in comparison to the preceding network. This reduction in error signifies an improvement in the network's performance, highlighting the effectiveness of the current configuration. The lower error rate suggests that the network is better at capturing the underlying patterns in the data, leading to more accurate predictions.

The observed improvements can be attributed to several factors, including the specific arrangement of convolutions and the overall architecture of the network. The sequential convolutions likely enhance the network's ability to extract and refine features, while the output convolution ensures that the final predictions are well-aligned with the true values. Additionally, the training process and hyperparameter settings may have been optimized to further enhance the network's performance.

In summary, the network's architecture, comprising two convolutions in sequence followed by an output convolution, has demonstrated a high level of precision in predicting the velocity field. The comparison with the true field and the reduced error rate underscore the network's effectiveness and reliability. These results highlight the potential of the current configuration to deliver accurate and precise

predictions, making it a valuable tool for applications requiring detailed velocity field analysis.
Ciao

Two layers deep one convolution This network consists of a down block and an up block, as described in section 4.2. Within these blocks, a series of operations are performed to process the input data. Initially, downsampling is conducted, which reduces the spatial dimensions of the input tensor while increasing its depth. This step is followed by a convolution operation, which helps in extracting features from the downsampled data. The purpose of downsampling is to capture more abstract and high-level features by reducing the resolution of the input image.

After the downsampling and convolution operations, the data is passed through an upsampling process. Upsampling restores the spatial dimensions of the tensor to their original size, effectively reversing the downsampling process. This is followed by another convolution operation, which refines the features extracted during the downsampling phase. The final convolution is performed with a kernel size of 1, which adjusts the number of channels to match the desired output format.

Despite these sophisticated operations, a slight reduction in the definition of the predicted image is observed. This reduction in image clarity indicates that the network is not fully able to reconstruct the fine details of the input image. Additionally, the percentage error remains approximately consistent with the previous case, suggesting that the network's overall performance has not significantly improved.

The consistent error rate, coupled with the reduction in image definition, suggests that the network in question exhibits poorer performance compared to other configurations. This could be due to several factors, such as the complexity of the task, limitations in the network architecture, or insufficient training data. The observed performance issues highlight the need for further investigation and optimization.

To address these challenges, it may be necessary to revisit the training process and consider augmenting the dataset to include more diverse examples. This can help the network learn to generalize better and improve its performance on unseen data. Experimenting with different hyperparameter settings, such as learning rate, batch size, and the number of epochs, can also help identify the optimal configuration for the network.

Additionally, implementing advanced techniques like data augmentation, regularization, and fine-tuning of the model can contribute to better performance. Data augmentation can introduce variability in the training data, making the network more robust to different types of input. Regularization techniques, such as dropout or weight decay, can prevent overfitting and improve the model's generalization capabilities. Fine-tuning the model by adjusting the architecture or training process

can also lead to better performance.

In summary, while the network's architecture includes sophisticated operations such as downsampling, upsampling, and convolutions, the reduction in image definition and the consistent error rate indicate areas for potential improvement. By addressing these factors through careful analysis and optimization, the network's performance can be enhanced, ensuring it produces high-quality, accurate outputs.

Two layers depp two concolutions The configuration of this network is analogous to that observed in the preceding case; however, in this instance, two convolutions are conducted in succession within both the up and down blocks. This modification aims to enhance the feature extraction and reconstruction capabilities of the network by allowing it to process more complex patterns and details. The down blocks reduce the spatial dimensions of the input tensor while increasing the depth, capturing more abstract features. Conversely, the up blocks restore the spatial dimensions while reducing the depth, reconstructing the original image from the learned features.

Despite these architectural enhancements, we observe a further reduction in resolution and a notable increase in the percentage error in the velocity values. This reduction in resolution suggests that the network is struggling to maintain the same level of detail and precision as before. The consecutive convolutions within the up and down blocks may be introducing additional complexity that the network is unable to handle effectively, leading to a loss of critical spatial information.

The increase in the percentage error in the velocity values indicates a deterioration in the network's ability to accurately predict these values. This could be due to several factors, including insufficient training data, suboptimal hyperparameter settings, or the inherent complexity of the task. The observed decline in performance highlights the need for further investigation and optimization.

To address these issues, it may be necessary to revisit the training process and consider augmenting the dataset to include more diverse examples. This can help the network learn to generalize better and improve its performance on unseen data. Experimenting with different hyperparameter settings, such as learning rate, batch size, and the number of epochs, can also help identify the optimal configuration for the network.

Additionally, implementing advanced techniques like data augmentation, regularization, and fine-tuning of the model can contribute to better performance. Data augmentation can introduce variability in the training data, making the network more robust to different types of input. Regularization techniques, such as dropout or weight decay, can prevent overfitting and improve the model's generalization capabilities. Fine-tuning the model by adjusting the architecture or training process can also lead to better performance.

In summary, while the configuration of the network includes enhancements

such as consecutive convolutions within the up and down blocks, the reduction in resolution and the increase in percentage error in the velocity values indicate areas for potential improvement. By addressing these factors through careful analysis and optimization, the network's performance can be enhanced, ensuring it produces high-quality, accurate outputs.

Three layers deep and one convolution In this instance, the addition of both a down block and an up block is observed. The down block is responsible for reducing the spatial dimensions of the input tensor while increasing the depth, allowing the network to capture more abstract features. Conversely, the up block restores the spatial dimensions while reducing the depth, aiming to reconstruct the original image from the learned features. Despite these architectural enhancements, it is evident that the predicted images exhibit a further reduction in definition compared to the actual images.

This reduction in image definition suggests that the network is struggling to accurately reconstruct the finer details of the input images. The down block may be losing critical spatial information during the dimensionality reduction process, which the up block is unable to fully recover. Additionally, there is a discernible increase in the error percentage, which is indicative of a deterioration in the quality of the predicted images. This increase in error percentage could be due to several factors, including insufficient training data, suboptimal hyperparameter settings, or the complexity of the images being beyond the current capacity of the network.

The observed decline in image quality and the rise in error percentage highlight the need for further investigation and optimization. One potential approach is to review the training data to ensure it is comprehensive and representative of the target domain. Augmenting the dataset with more diverse examples could help the network learn to generalize better. Another strategy is to experiment with different hyperparameter settings, such as learning rate, batch size, and the number of epochs, to find the optimal configuration for the network.

Additionally, implementing advanced techniques like data augmentation, regularization, and fine-tuning of the model could help improve the network's performance. Data augmentation can introduce variability in the training data, helping the network become more robust to different types of input. Regularization techniques, such as dropout or weight decay, can prevent overfitting and improve the model's generalization capabilities. Fine-tuning the model by adjusting the architecture or training process can also lead to better performance.

In summary, while the addition of both a down block and an up block represents a significant architectural enhancement, the reduction in image definition and the increase in error percentage indicate areas for potential improvement. By addressing these factors through careful analysis and optimization, the network's performance can be enhanced, ensuring it produces high-quality, accurate outputs.

Three layers and two convolutions In this instance, although the structure remains identical to the previous iteration, with two convolutions per block, there is a notable reduction in the clarity of the images produced by the network. This deterioration in image quality is accompanied by a significant increase in the percentage error rate. The consistency in the network's architecture indicates that the observed decline in performance may be attributed to other factors, such as alterations in the training data, modifications in the hyperparameters, or potential overfitting.

The reduction in image clarity suggests that the network is struggling to maintain the same level of detail and precision as before. This could be due to changes in the training data, which might not be as representative or diverse as needed. Variations in the hyperparameters, such as learning rate, batch size, or the number of epochs, could also be contributing to the decline in performance. Additionally, overfitting might be occurring, where the model performs well on the training data but fails to generalize to new, unseen data.

These issues underscore the necessity for continuous monitoring and fine-tuning of the model to guarantee optimal performance and accuracy. Regular evaluation of the training data is essential to ensure it remains representative of the target domain. Experimenting with different hyperparameter settings can help identify the optimal configuration for the network. Implementing regularization techniques, such as dropout or weight decay, can mitigate overfitting and improve the model's generalization capabilities.

Moreover, it is crucial to analyze the network's performance metrics regularly to identify any trends or anomalies that might indicate underlying issues. By doing so, adjustments can be made promptly to address any problems and maintain the network's performance. This iterative process of monitoring, evaluating, and fine-tuning is vital for achieving and sustaining high levels of accuracy and image quality.

In summary, while the network's architecture remains consistent, the decline in image clarity and the increase in error rate highlight the need for ongoing optimization. By addressing factors such as training data quality, hyperparameter settings, and potential overfitting, the network's performance can be enhanced, ensuring it produces high-quality, accurate outputs.

A.2 ResNet

It was therefore decided to train some residual networks, for the reasons previously outlined, as illustrated in section 2.4. In particular, three distinct configurations were trained, each comprising two residual blocks. However, the number of channels in the convolutional layers and the number of convolutions within the residual

blocks were varied in order to investigate the impact of these hyper-parameters on the performance of the network:

- 16 channels, 4 convolutions;
- 32 channels, 4 convolutions;
- 32 channels, 8 convolutions.

Network with 16 channels, 4 convolutions The initial convolution is executed with a kernel size of 1, primarily to augment the channel number of the images from 2 to 16. This step is crucial as it prepares the data for more complex processing by increasing the dimensionality of the feature space. Thereafter, the novel tensor is conveyed to the residual block, which comprises four convolution layers for each block. These residual blocks are designed to facilitate deeper network architectures by allowing gradients to flow through the network more effectively, thereby improving learning and performance.

Subsequent to the residual blocks, the information is transmitted to a convolutional layer. This layer is responsible for restoring the images from 16 channels back to their original dimension. This step ensures that the output of the network is in the same format as the input, making it suitable for further processing or evaluation.

As evidenced by the results presented in Figure A.24, while the percentage error is relatively low, the generated images exhibit a lack of definition. This indicates that although the network is performing well in terms of error metrics, the quality of the images is not satisfactory. The lack of definition in the images could be due to several factors, such as insufficient training data, suboptimal hyperparameter settings, or the inherent limitations of the network architecture.

To address these issues, it may be necessary to revisit the training process and consider augmenting the dataset to include more diverse examples. Additionally, experimenting with different hyperparameters, such as learning rate, batch size, and the number of epochs, could help improve the network's performance. Implementing advanced techniques like data augmentation, regularization, and fine-tuning of the model could also contribute to better image quality.

In summary, while the network's architecture and error metrics are promising, the lack of definition in the generated images highlights the need for further optimization and refinement. By addressing these factors, it is possible to enhance both the accuracy and the visual quality of the network's output.

Network with 32 channels, 4 convolutions The network structure is identical to that of the previous one. However, the input tensor is transformed into 32 channels in the first convolutional layer. This transformation significantly increases

the number of learnable parameters within the network, particularly in the residual blocks. The increase in channels from the initial 2 to 32 allows the network to capture more complex features and patterns from the input data, enhancing its ability to learn and generalize.

The residual blocks themselves remain unchanged in terms of their architecture, but the number of channels has been increased to 32. This modification means that each convolutional layer within the residual blocks now processes a higher-dimensional tensor, which can lead to more detailed and nuanced feature extraction. The increased number of channels contributes to a richer representation of the input data, potentially improving the network's performance.

As evidenced by the results presented in Figure A.27, while the percentage error is marginally higher, the generated images exhibit greater definition. This suggests that the U-Net performs even better in terms of visual quality, despite the slight increase in error rate. The enhanced image definition indicates that the network is able to produce more detailed and accurate reconstructions, which is a crucial aspect of its overall performance.

The observed improvements in image quality can be attributed to the increased capacity of the network, allowing it to learn more complex representations. However, the marginally higher error rate may indicate that the network is also more sensitive to variations in the input data, which could be addressed through further fine-tuning and optimization.

In summary, the modifications to the network, specifically the increase in the number of channels, have led to a notable improvement in the definition of the generated images. While the percentage error has slightly increased, the overall performance of the U-Net is enhanced, demonstrating its capability to produce higher-quality outputs. Continuous monitoring and adjustments will be essential to balance the trade-off between error rate and image quality, ensuring the network achieves optimal results.

Network with 32 channels, 8 convolutions In this network, the initial convolution layer is responsible for transforming the image into a 32-channel tensor. This transformation establishes the foundation for subsequent processing by the network. Subsequently, two residual blocks comprising eight convolution layers each are employed. The residual blocks are designed to enhance feature extraction and improve the network's capacity to learn complex patterns. Notwithstanding the maintenance of this structure, a reduction in image definition is apparent. This decline in clarity is accompanied by a slight increase in the error rate.

The observed deterioration in image quality and the concomitant increase in errors suggest the presence of potential issues that may require attention. These issues may have their origin in a number of factors, including alterations to the training data, fluctuations in the hyperparameters, or the potential for overfitting.

Overfitting occurs when the model demonstrates high performance on the training data set but exhibits poor generalization to new, unseen data. This emphasises the necessity for continuous monitoring and fine-tuning of the model in order to guarantee optimal performance and accuracy.

In order to address these issues, a number of potential strategies can be considered. One potential solution is to conduct a review of the training data and, if necessary, implement modifications to ensure its representativeness of the target domain. Another strategy is to conduct experiments with different hyperparameter settings with the objective of identifying the optimal configuration for the network. Furthermore, the incorporation of regularisation techniques, such as dropout or weight decay, can assist in the mitigation of overfitting and the enhancement of the model's generalisation capabilities.

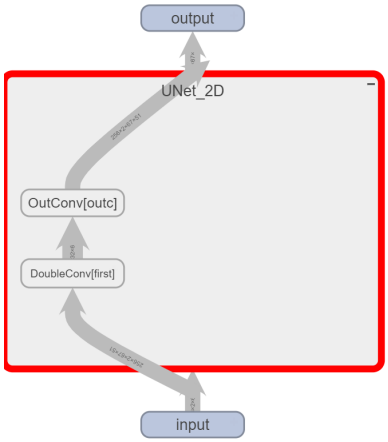

Figure A.1: Structure of the CNN with only one deep layer

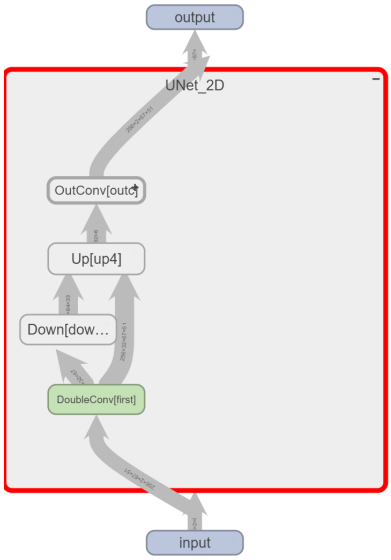

Figure A.2: Structure of the CNN with two deep layers

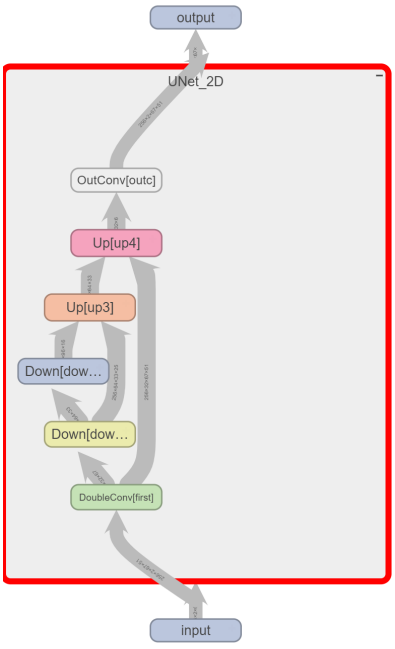

Figure A.3: Structure of the CNN with three deep layers

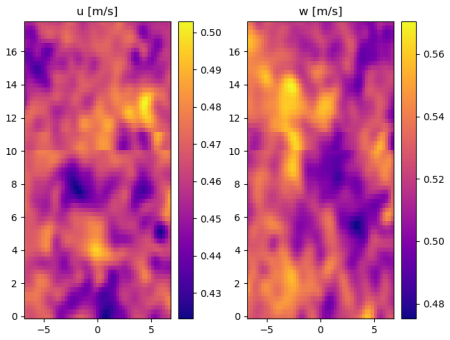

Figure A.4: Case deep1 and 1conv: Real velocity field, on the left there is the radial component, at the right the streamwise component

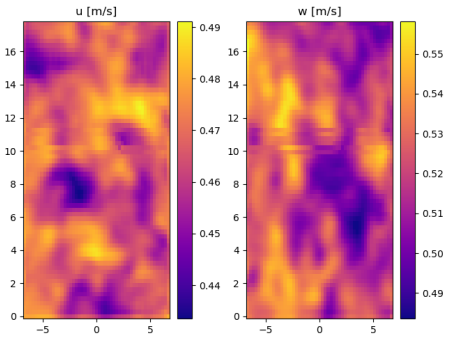

Figure A.5: Case deep1 and 1conv: Predicted velocity field, on the left there is the radial component, at the right the streamwise component

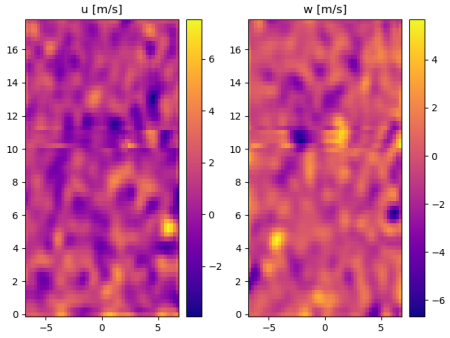

Figure A.6: Case deep1 and 1conv: Percerntage error on the velocity prediction, at the right the streamwise component

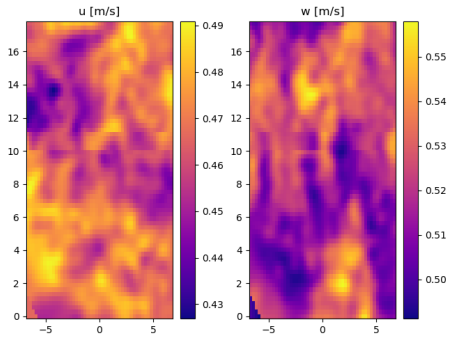

Figure A.7: Case deep1 and 2conv: Real velocity field, on the left there is the radial component, at the right the streamwise component

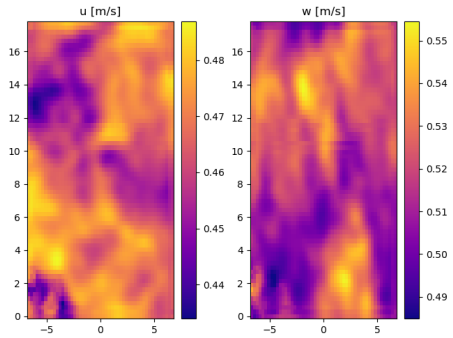

Figure A.8: Case deep1 and 2conv: Predicted velocity field, on the left there is the radial component, at the right the streamwise component

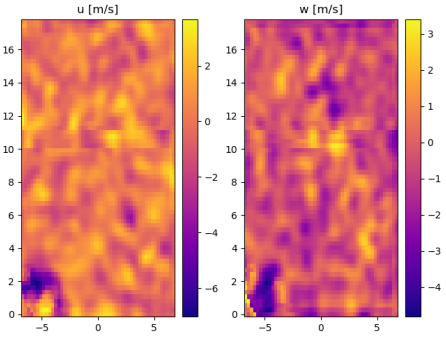

Figure A.9: Case deep1 and 2conv: Percerntage error on the velocity prediction, at the right the streamwise component

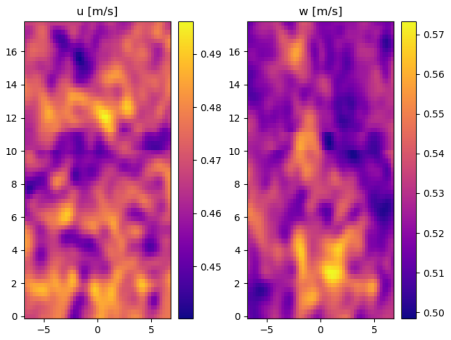

Figure A.10: Case deep2 and 1conv: Real velocity field, on the left there is the radial component, at the right the streamwise component

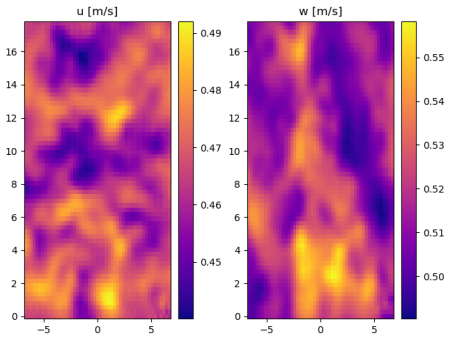

Figure A.11: Case deep2 and 1conv: Predicted velocity field, on the left there is the radial component, at the right the streamwise component

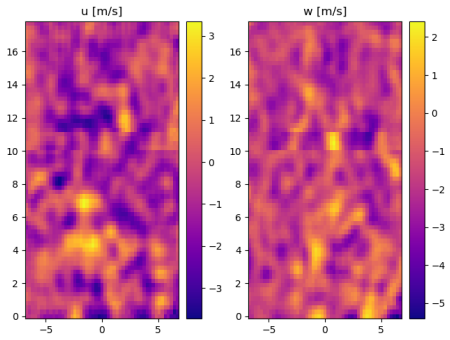

Figure A.12: Case deep2 and 1conv: Percerntage error on the velocity prediction, at the right the streamwise component

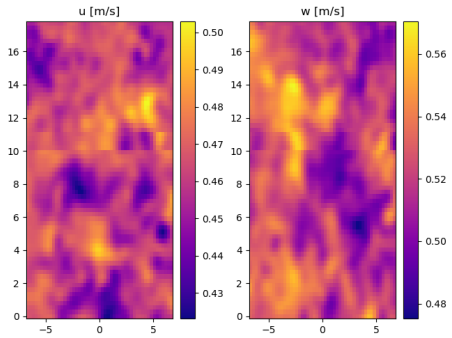

Figure A.13: Case deep2 and 2conv: Real velocity field, on the left there is the radial component, at the right the streamwise component

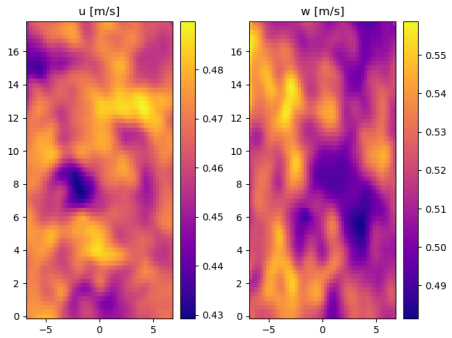

Figure A.14: Case deep2 and 2conv: Predicted velocity field, on the left there is the radial component, at the right the streamwise component

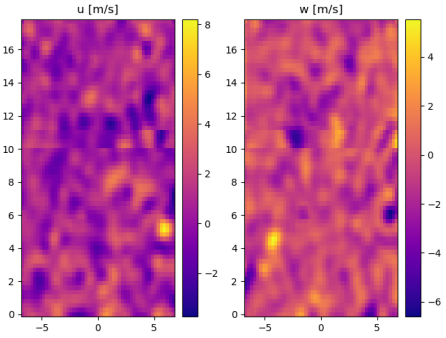

Figure A.15: Case deep2 and 2conv: Percerntage error on the velocity prediction, at the right the streamwise component

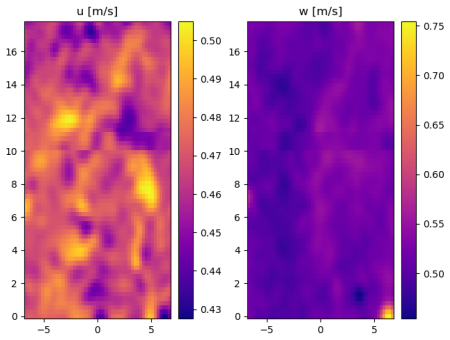

Figure A.16: Case deep3 and 1conv: Real velocity field, on the left there is the radial component, at the right the streamwise component

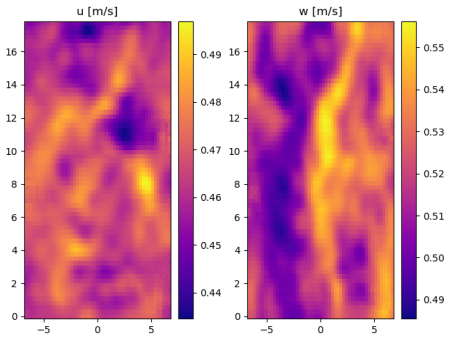

Figure A.17: Case deep3 and 1conv: Predicted velocity field, on the left there is the radial component, at the right the streamwise component

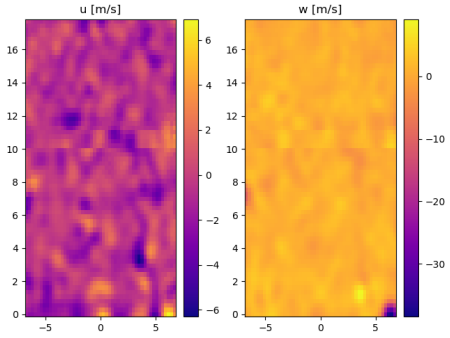

Figure A.18: Case deep3 and 1conv: Percerntage error on the velocity prediction, at the right the streamwise component

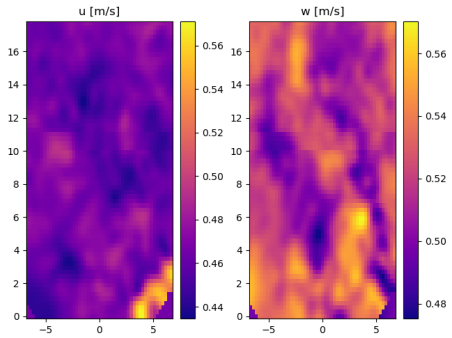

Figure A.19: Case deep3 and 2conv: Real velocity field, on the left there is the radial component, at the right the streamwise component

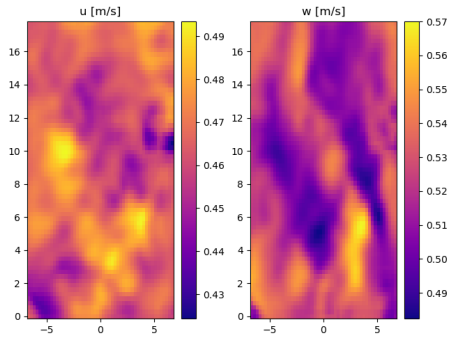

Figure A.20: Case deep3 and 2conv: Predicted velocity field, on the left there is the radial component, at the right the streamwise component

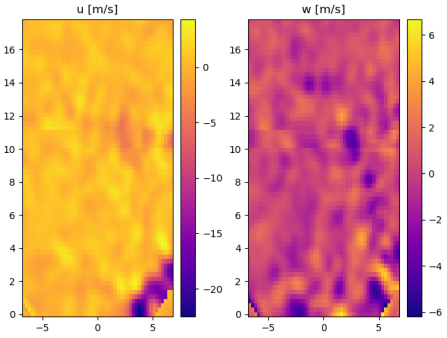

Figure A.21: Case deep3 and 2conv: Percerntage error on the velocity prediction, at the right the streamwise component

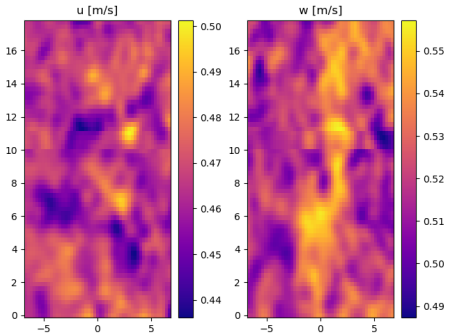

Figure A.22: Network with 16 channels, 4 convolutions: real velocity field, on the left there is the radial component, at the right the streamwise component

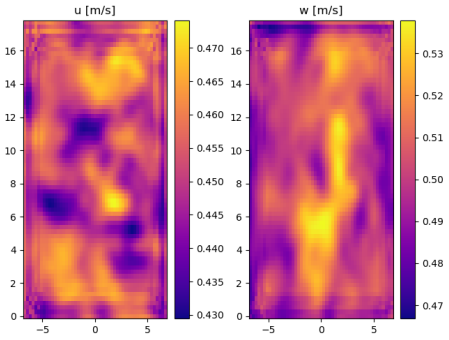

Figure A.23: Network with 16 channels, 4 convolutions: predicted velocity field, on the left there is the radial component, at the right the streamwise component

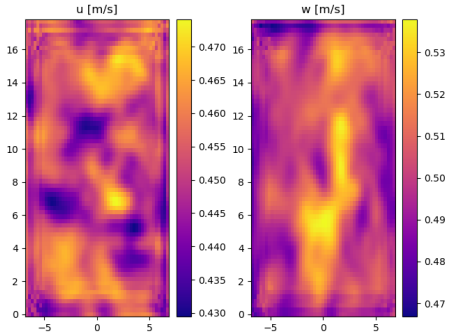

Figure A.24: Network with 16 channels, 4 convolutions: percerntage error on the velocity prediction, at the right the streamwise component

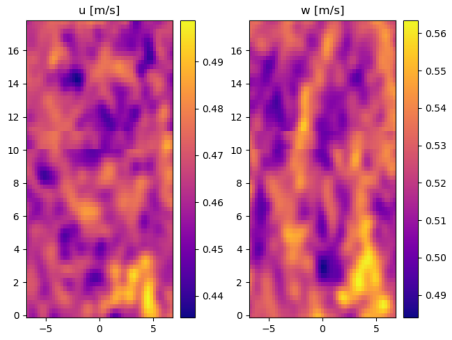

Figure A.25: Network with 32 channels, 4 convolutions: real velocity field, on the left there is the radial component, at the right the streamwise component

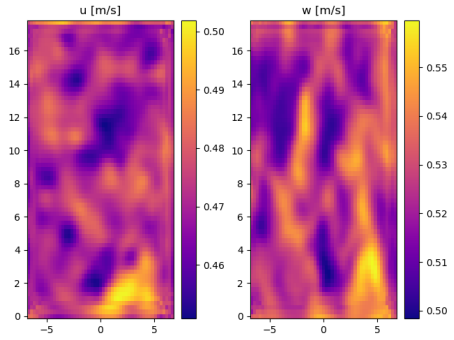

Figure A.26: Network with 32 channels, 4 convolutions: predicted velocity field, on the left there is the radial component, at the right the streamwise component

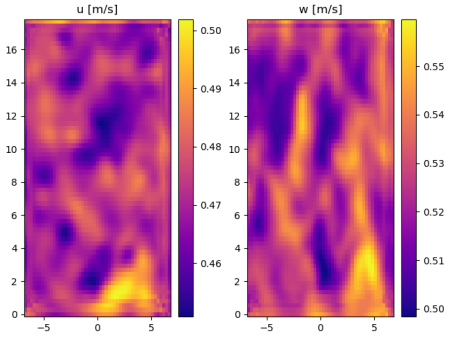

Figure A.27: Network with 32 channels, 4 convolutions: percerntage error on the velocity prediction, at the right the streamwise component

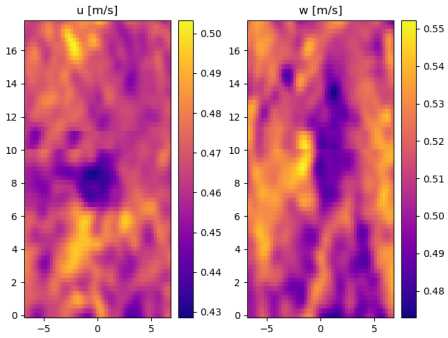

Figure A.28: Network with 32 channels, 4 convolutions: real velocity field, on the left there is the radial component, at the right the streamwise component

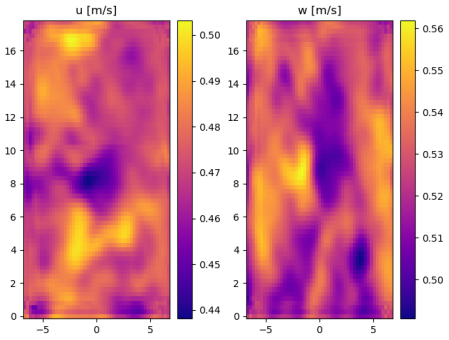

Figure A.29: Network with 32 channels, 4 convolutions: predicted velocity field, on the left there is the radial component, at the right the streamwise component

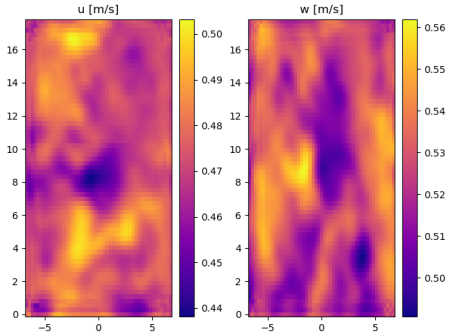

Figure A.30: Network with 32 channels, 4 convolutions: percerntage error on the velocity prediction, at the right the streamwise component

Bibliography

[1] Koldo Portal-Porras, Unai Fernandez-Gamiz, Ekaitz Zulueta, Oscar Irigaray, and Roberto Garcia-Fernandez. «Hybrid LSTM+CNN architecture for unsteady flow prediction». eng. In: *Materials today communications* 35 (2023), pp. 106281–. ISSN: 2352-4928 (cit. on pp. 2, 10, 11).

[2] L. S. Shapley. «17. A Value for n-Person Games». In: *Contributions to the Theory of Games, Volume II*. Ed. by Harold William Kuhn and Albert William Tucker. Princeton: Princeton University Press, 1953, pp. 307–318. ISBN: 9781400881970. DOI: doi:10.1515/9781400881970-018. URL: https://doi.org/10.1515/9781400881970-018 (cit. on p. 2).

[3] Simon Haykin and Simon Haykin. *Neural networks : a comprehensive foundation / Simon Haykin*. eng. 2nd ed. Upper Saddle River: Prentice-Hall, 1999. ISBN: 9780132733502 (cit. on p. 6).

[4] John von Neumann. *The computer and the brain / John von Neumann*. eng. Silliman milestones in science. New Haven: Yale University Press, 1958. ISBN: 0-300-02415-0 (cit. on p. 7).

[5] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. «Backpropagation applied to handwritten zip code recognition». In: *Neural computation* 1.4 (1989), pp. 541–551 (cit. on p. 9).

[6] Keiron O'Shea and Ryan Nash. «An Introduction to Convolutional Neural Networks». eng. In: *arXiv.org* (2015). ISSN: 2331-8422 (cit. on p. 9).

[7] Yuqing Hou, Hui Li, Hong Chen, Wei Wei, Jiayue Wang, and Yicang Huang. «A novel deep U-Net-LSTM framework for time-sequenced hydrodynamics prediction of the SUBOFF AFF-8». eng. In: *Engineering applications of computational fluid mechanics* 16.1 (2022), pp. 630–645. ISSN: 1994-2060 (cit. on p. 10).

[8] Sasha Targ, Diogo Almeida, and Kevin Lyman. «Resnet in Resnet: Generalizing Residual Architectures». eng. In: *arXiv.org* (2016). ISSN: 2331-8422 (cit. on p. 12).

[9] Paszke Adam, Gross Sam, Chintala Soumith, and Chanan Gregory. *PyTorch Documenation*. Available on line. 2024. URL: https://pytorch.org/docs/stable/index.html (cit. on pp. 14, 16, 17, 36).

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification». eng. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 1026–1034. ISBN: 1467383910 (cit. on p. 16).

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. «Image-to-Image Translation with Conditional Adversarial Networks». eng. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 5967–5976. ISBN: 1538604574 (cit. on p. 19).

[12] Ugur Demir and Gozde Unal. «Patch-Based Image Inpainting with Generative Adversarial Networks». eng. In: *arXiv.org* (2018). ISSN: 2331-8422 (cit. on p. 19).

[13] Arun Solanki. *Generative adversarial networks for image-to-image translation*. eng. Amsterdam, Netherlands: Elsevier, 2021. ISBN: 0-12-823613-2 (cit. on p. 20).

[14] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. «Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks». In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[15] Brian Landers. «Mixing Characteristics of Turbulent Twin Impinging Axisymmetric Jets at Various Impingement Angles». PhD thesis. June 2016. DOI: 10.13140/RG.2.1.1028.2482 (cit. on pp. 24, 26).

[16] Pijush Kundu, Ira M. Cohen, and Pijush Kundu. *Fluid mechanics / Pijush K. Kundu, Ira M. Cohen*. eng. 3rd ed. Amsterdam: Elsevier, 2004. ISBN: 0-12-178253-0 (cit. on p. 25).

[17] Syed Harris Hassan, Tianqi Guo, and Pavlos P. Vlachos. «Flow field evolution and entrainment in a free surface plunging jet». In: *Phys. Rev. Fluids* 4 (10 Oct. 2019), p. 104603. DOI: 10.1103/PhysRevFluids.4.104603. URL: https://link.aps.org/doi/10.1103/PhysRevFluids.4.104603 (cit. on p. 26).

[18] JAVIER JIMÉNEZ and ALAN A. WRAY. «On the characteristics of vortex filaments in isotropic turbulence». In: *Journal of Fluid Mechanics* 373 (1998), pp. 255–285. DOI: 10.1017/S0022112098002341 (cit. on p. 28).

[19] Ethan Pickering, Georgios Rigas, Petrônio A. S. Nogueira, André V. G. Cavalieri, Oliver T. Schmidt, and Tim Colonius. «Lift-up, Kelvin–Helmholtz and Orr mechanisms in turbulent jets». eng. In: *Journal of fluid mechanics* 896 (2020). ISSN: 0022-1120 (cit. on p. 28).

[20] Gilles Tissot, Francisco C. Lajús, André V. G. Cavalieri, and Peter Jordan. «Wave packets and Orr mechanism in turbulent jets». eng. In: *Physical review fluids* 2.9 (2017). ISSN: 2469-990X (cit. on pp. 28, 29).

[21] William M'F Orr. «The stability or instability of the steady motions of a perfect liquid and of a viscous liquid. Part II: A viscous liquid». In: *Proceedings of the Royal Irish Academy. Section A: Mathematical and Physical Sciences*. Vol. 27. JSTOR. 1907, pp. 69–138 (cit. on p. 29).

[22] Yuxin Jiao, Yongyun Hwang, and Sergei Chernyshenko. «The Orr mechanism in transition of parallel shear flow». In: *Physical Review Fluids* 6 (Jan. 2021), p. 023902. DOI: 10.1103/PhysRevFluids.6.023902 (cit. on p. 29).

[23] Petrônio A. S. Nogueira, André V. G. Cavalieri, Peter Jordan, and Vincent Jaunet. «Large-scale streaky structures in turbulent jets». eng. In: *Journal of fluid mechanics* 873 (2019), pp. 211–237. ISSN: 0022-1120 (cit. on p. 30).

[24] Sukesh Roy, Joseph D. Miller, and Gemunu H. Gunaratne. «Deviations from Taylor's frozen hypothesis and scaling laws in inhomogeneous jet flows». eng. In: *Communications physics* 4.1 (2021), pp. 1–8. ISSN: 2399-3650 (cit. on p. 32).

[25] Gemunu Gunaratne, Joseph Miller, and Sukesh Roy. «Turbulent Jet Flow Data». In: (Aug. 2020). DOI: 10.6084/m9.figshare.12789926.v2. URL: https://figshare.com/articles/dataset/Turbulent_Jet_Flow_Data/12789926 (cit. on p. 33).

[26] Haval A. Ahmed, Peshawa J. Muhammad Ali, Abdulbasit K. Faeq, and Saman M. Abdullah. «An Investigation on Disparity Responds of Machine Learning Algorithms to Data Normalization Method». eng. In: *ARO (Koya)* 10.2 (2022), pp. 29–37. ISSN: 2410-9355 (cit. on p. 33).

[27] «Predicting Coherent Turbulent Structures via Deep Learning». eng. In: *Frontiers in physics* 10 (2022). ISSN: 2296-424X (cit. on pp. 34, 35).

[28] Andrés Cremades et al. «Identifying regions of importance in wall-bounded turbulence through explainable deep learning». eng. In: *Nature communications* 15.1 (2024), pp. 3864–3864. ISSN: 2041-1723 (cit. on pp. 41, 42).

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. «"Why Should I Trust You?": Explaining the Predictions of Any Classifier». eng. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2016, pp. 1135–1144. ISBN: 1450342329 (cit. on p. 42).

[30] Scott M Lundberg and Su-In Lee. «A Unified Approach to Interpreting Model Predictions». In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf (cit. on pp. 42, 50).

[31] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. «scikit-image: image processing in Python». In: *PeerJ* 2 (2014), e453 (cit. on p. 43).

[32] Shi Na, Liu Xumin, and Guan Yong. «Research on k-means Clustering Algorithm: An Improved k-means Clustering Algorithm». In: *2010 Third International Symposium on Intelligent Information Technology and Security Informatics*. 2010, pp. 63–67. DOI: 10.1109/IITSI.2010.74 (cit. on p. 55).