IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Internship Report

Subject: *Estimation of the local energy dissipation rate of a turbulent flow through machine learning approaches*

Name: Thomas Favre

Supervisor: Carlos Granero-Belinchon

Employer: IMT Atlantique , Brest

Department: Mathematical and Electrical Engineering

Internship Type: 2nd year engineering research internship, 4 months

Academic Year: 2024–2025

Contents

1 Background

1.1 Disclaimer

This report aims to provide a comprehensive summary of the work conducted during my internship at IMT Atlantique, Brest, under the supervision of Carlos Granero-Belinchon.

It is important to emphasize that this document does not present novel discoveries. Rather, it serves as a synthesis of various exploratory approaches undertaken throughout the internship, many of which led to inconclusive or unsuccessful outcomes. Consequently, the report encompasses a diverse range of ideas and methodologies, reflecting the iterative nature of the research process. Readers may find the structure of this document somewhat unconventional due to the breadth of topics and experimental directions explored.

1.2 Physical Context

Turbulent flows are characterized by chaotic, multi-scale motions in which energy is injected at large scales, transferred through intermediate scales, and ultimately dissipated at small scales due to viscosity. This process is described by the classical energy cascade, as introduced in Kolmogorov's theory of turbulence.

In such flows, the dissipation rate of turbulent kinetic energy is a key quantity that governs small-scale dynamics and plays a central role in processes such as scalar mixing, momentum transport, and the behavior of Lagrangian tracers. Mathematically, the dissipation rate is defined as in [8]:

$$\varepsilon = \frac{1}{2}\nu \left\langle \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)^2 \right\rangle,$$

where ν is the kinematic viscosity, u_i are the components of the velocity field, and the angle brackets denote averaging over space or time.

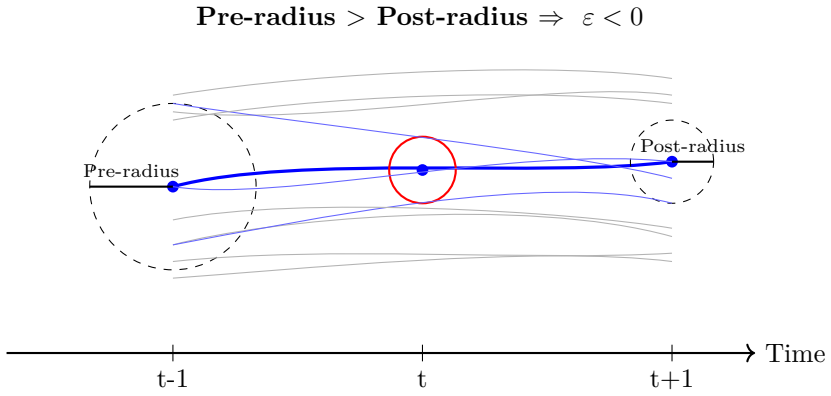
In this work, we focus on the case of isotropic turbulence, where flow properties are statistically uniform in all directions. Under this assumption, the dissipation rate simplifies to a scalar quantity that can be estimated from a single velocity gradient component as in [3]:

$$\varepsilon = 15\nu \left\langle \left(\frac{\partial u_i}{\partial x_i} \right)^2 \right\rangle.$$

Lagrangian particles in a turbulent flow interact with structures across a wide range of scales. The local dissipation encountered by a particle directly affects its acceleration, dispersion, and interaction with the surrounding fluid.

Cheminet et al proposed [1] an approach to estimate the dissipation from Lagrangian trajectories than relies on the use of several nearby particles. By identifying particles within a certain spatial neighborhood at each time step, it becomes possible to approximate the local velocity gradient tensor surrounding the particle of interest. This gradient tensor is then used to compute the instantaneous dissipation rate along the particle's trajectory.

To illustrate this process, the diagram below shows a focused particle trajectory surrounded by spatial neighborhoods at three consecutive time steps. The thick blue trajectory represents the particle of interest, while the other blue trajectories correspond to neighboring particles used to estimate the local velocity gradient tensor and, consequently, the dissipation. The shrinking or expansion of these spatial neighborhoods over time reflects changes in local deformation rates, which directly inform the dissipation experienced by the central particle.



Our objective is to overcome this limitation by predicting the dissipation using only the trajectory of the particle itself—without requiring access to its neighbors or the global velocity field.

More precisely, we aim to learn a function g such that:

$$\hat{\varepsilon}_t = g(v_{t-n/2}, v_{t-n/2+1}, \ldots, v_{t+n/2},\ a_{t-n/2}, a_{t-n/2+1}, \ldots, a_{t+n/2}), \tag{1}$$

where $\hat{\varepsilon}_t$ is the estimated dissipation at time step t, v_{t-k} is the velocity vector, and a_{t-k} is the acceleration vector at time step $t-k$.

That is, our goal is to estimate the dissipation at a specific point along a particle's trajectory using only its velocity and acceleration over a time window centered around the target time step.

1.3 Dataset and Assumptions

We used an experimental dataset consisting of particle trajectories in a turbulent flow. The experimental conditions are described in detail in [2].

We assume the flow is *stationary*, *isotropic*, and *homogeneous*. That is, statistically, there is no preferred direction, location, or moment in time. This implies that statistical properties computed from one subset of trajectories (e.g., along a specific spatial direction, in a given region, or over a time interval) should be consistent with those computed from another subset.

The dataset comprises 10 experimental runs, labeled Run 1 through Run 10.

Before proceeding, we first verify whether the assumptions of stationarity, isotropy, and homogeneity hold in the data.

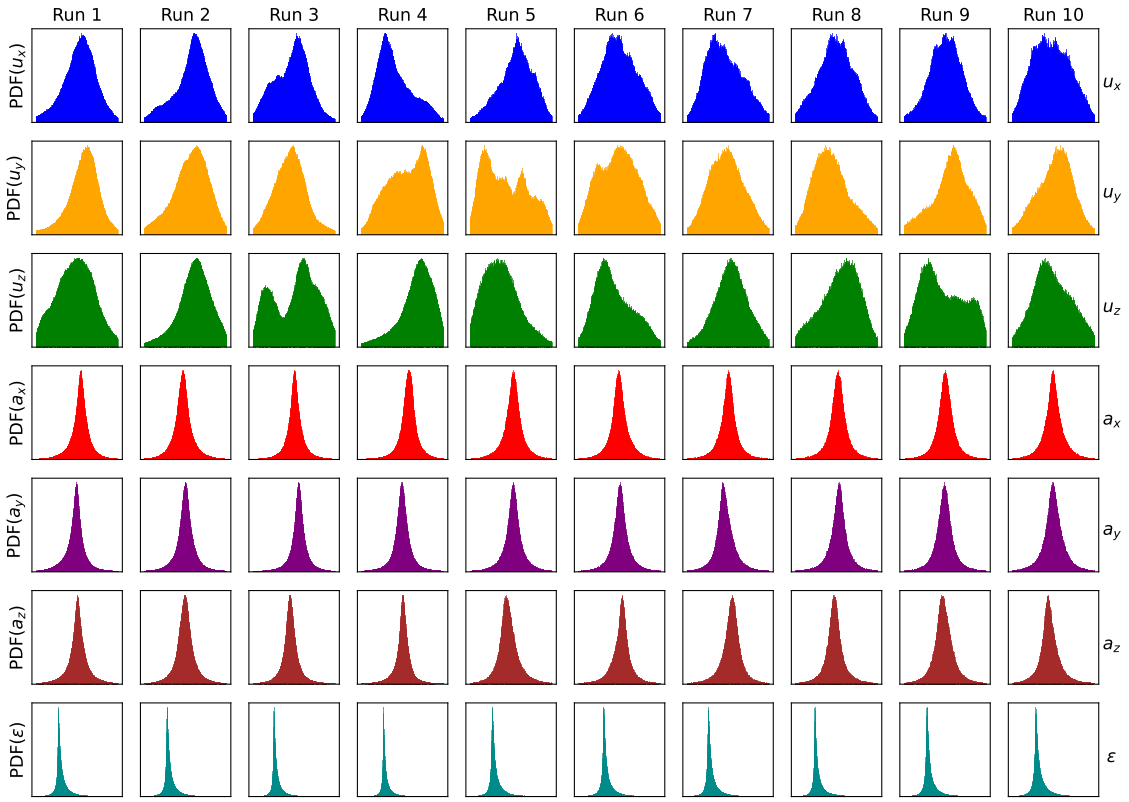


Figure 1: Distributions of velocity, acceleration and dissipation for Runs 1 to 10.

We observe the following:

- The acceleration and dissipation distributions are consistent across runs, suggesting that the flow is stationary.
- These distributions are also similar in all directions, indicating isotropy.

However, the velocity vector distributions show more variability across runs, making it harder to confirm isotropy or stationarity for velocity.

Additionally, we observe that the dissipation distribution is asymmetric with a heavy tail. This implies that large dissipation events—though rare—occur with non-negligible probability.

To better understand the statistical properties of the data, we plot violin plots of the velocity, acceleration, and dissipation vectors for Run 1, using 3 consecutive time steps.

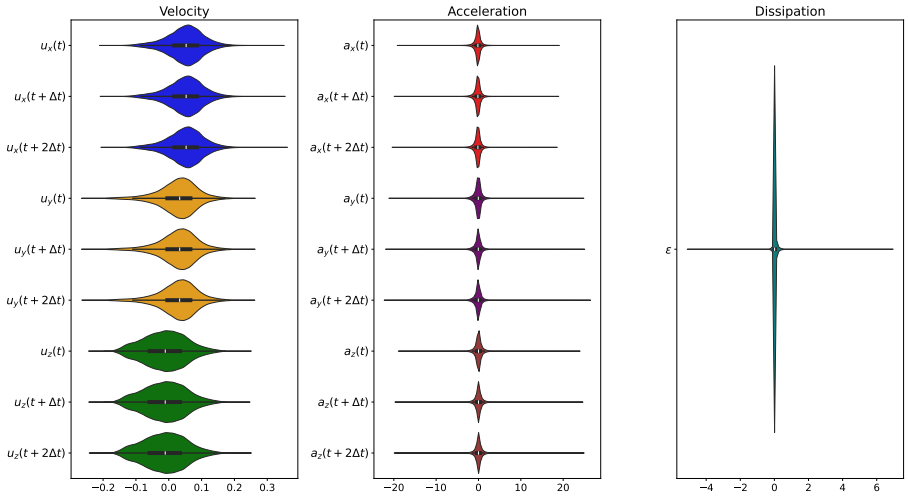


Figure 2: Violin plots of velocity, acceleration, and dissipation vectors (Run 1, 3 time steps).

We observe the presence of outliers, particularly in the acceleration and dissipation distributions. These outliers correspond to extreme values, which complicate the estimation of dissipation, especially for rare but significant events—referred to here as *extreme events*.

Are these extreme events physical or experimental errors?

1.4 Dependency Between Dissipation and Particle Dynamics

We now investigate whether a dependency exists between dissipation and the velocity or acceleration vectors of a particle. To this end, we compute the *mutual information* between dissipation and these quantities using the method introduced in [5]. To obtain a more robust estimate, we split the dataset into 100 subsets and compute the mutual information between the dissipation and each dimension within each subset. We then calculate the mean and standard deviation of the mutual information across all subsets for each dimension.

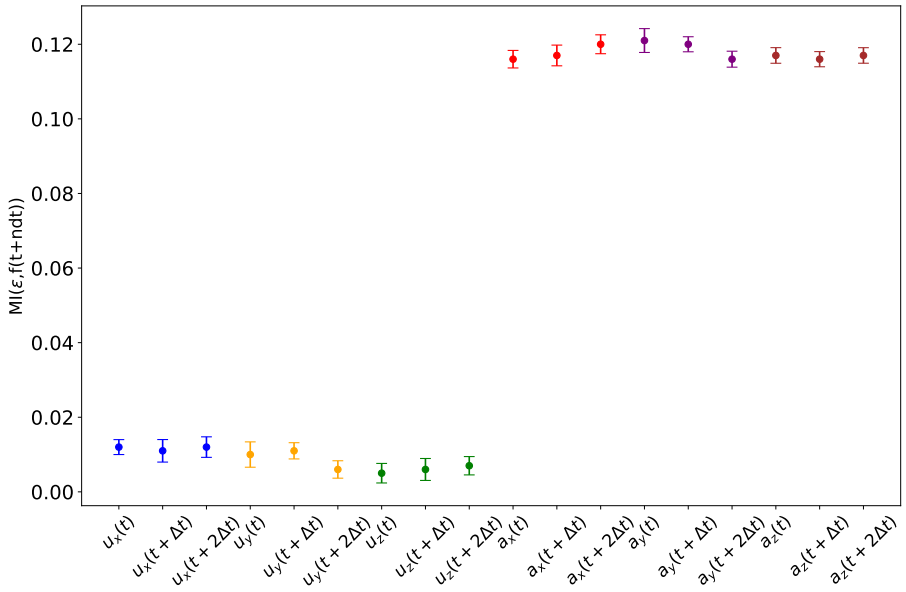


Figure 3: Mutual information between dissipation and velocity/acceleration vectors over 3 time steps.

The results show that mutual information is consistently higher between dissipation and acceleration than between dissipation and velocity. This indicates a stronger statistical dependence between dissipation and acceleration. However, since mutual information is unbounded, it does not allow us to directly quantify the strength of this relationship.

1.5 Dataset Transformation: Analog-Based Framework

Our objective is to estimate the dissipation experienced by a particle using only its own trajectory, without relying on the trajectories of spatially neighboring particles. To achieve this, we adopt an *analog-based* approach.

The underlying hypothesis is that two trajectories that are close in *phase space*—that is, they exhibit similar velocity and acceleration vectors—should have similar dissipation values, regardless of their spatial proximity. This assumption forms the basis of the analog-based framework.

To implement this approach, we transform the dataset into a format that allows for the comparison of trajectory segments based on their dynamic features. The transformation consists of the following steps:

- We retain only trajectories with a minimum duration of 20 time steps, and for which the dissipation has been estimated using at least 10 neighboring particles.
- Each trajectory is divided into segments of p time steps. Between each segment, there is a stride s to reduce temporal correlation between segments.
- Distances in the phase space are measured using the Euclidean distance.

Let $\boldsymbol{X}$ denote the set of all trajectory segments, and $\boldsymbol{y}$ the corresponding dissipation values. Each segment in $\boldsymbol{X}$ is represented as a matrix of size $p \times 6$, where the six columns correspond to the three components of velocity and the three components of acceleration. Each element of $\boldsymbol{y}$ is a scalar representing the estimated dissipation associated with the corresponding segment.

In practice, we typically use $p = 5$ and $s = 3$.

Run	Number of Trajectory Segments
Run 1	4,521,015
Run 2	4,190,507
Run 3	2,437,601
Run 4	3,407,781
Run 5	951,386
Run 6	1,356,068
Run 7	974,321
Run 8	917,217
Run 9	1,025,617
Total	**19,781,513**

Table 1: Number of trajectory segments of size $p = 5$ in each run.

2 Analog-Based Model

2.1 Model Description

The analog-based model is a non-parametric regression approach that estimates the dissipation of a particle based on its trajectory segment. We start by identifying the k nearest segments from the database that are most similar to the current segment in terms of their velocity and acceleration vectors. This similarity is measured via the Euclidean distance between the current segment and all segments in the database.

Next, a weighted regression is performed on the dissipation values of these analogous segments to estimate the dissipation of the current segment. The weights are determined by the distance

between the current segment and the analogs — the closer the analog, the greater its influence in the regression.

For consistency, we reproduce the approach from [4].

For a given number k of analogs, the minimization problem is formulated as:

$$\min_{\beta_t} \|\vec{y} - \boldsymbol{X}_p \beta_t\|_W$$

where:

- $\vec{y}$ is the vector of dissipations, and $\boldsymbol{X}_p$ is a $k \times (p+1)$ matrix containing all analog vectors, with acceleration and velocities and a first column of ones:

$$\vec{y} = \begin{pmatrix} x_{t'_1} \\ \vdots \\ x_{t'_k} \end{pmatrix}, \quad \boldsymbol{X}_p = \begin{pmatrix} 1 & x_{t'_1 - dt} & \cdots & x_{t'_1 - p\cdot dt} \\ \vdots & \vdots & & \vdots \\ 1 & x_{t'_k - dt} & \cdots & x_{t'_k - p\cdot dt} \end{pmatrix} \tag{2}$$

- $\boldsymbol{W}$ is a $k \times k$ diagonal matrix containing the weights of each analog. The non-zero diagonal elements w_{ii} are defined as:

$$w_{ii} = \frac{e^{-||\vec{x}_t^{(p)} - \vec{x}_{t'_i}^{(p)}||/\lambda}}{\sum_{j=1}^{k} e^{-||\vec{x}_t^{(p)} - \vec{x}_{t'_j}^{(p)}||/\lambda}} \tag{3}$$

  where:

  - $\vec{x}_t^{(p)}$ is the segment of interest,
  - $\vec{x}_{t'_i}^{(p)}$ is the i-th analog segment,
  - $\lambda = \text{median}\left(\{||\vec{x}_t^{(p)} - \vec{x}_{t'_i}^{(p)}||\}_{1 \le i \le k}\right)$.

- β_t is the vector of regression coefficients, which depends on t via the selected analogs.

The ordinary least squares estimator for this weighted regression is:

$$\hat{\beta}_t = \left(\boldsymbol{X}_p^\top \boldsymbol{W} \boldsymbol{X}_p\right)^{-1} \boldsymbol{X}_p^\top \boldsymbol{W} \vec{y} \tag{4}$$

The analog prediction $\hat{x}_t^{(p)}$ and its variance σ_t^2 are given by:

$$\hat{x}_t^{(p)} \equiv \hat{\beta}_t^\top \vec{x}_t^{(p)}, \quad \sigma_t^2 = \sum_{i=1}^{k} w_{ii} \left(\xi_{t,i}\right)^2 \tag{5}$$

where $\vec{\xi}_t = \vec{y} - \boldsymbol{W}\boldsymbol{X}_p \hat{\beta}_t$ are the residuals.

2.2 Results

We compute the root mean squared error (RMSE) of the analog-based model for various values of k and p. The dataset consists of trajectories from runs 1 to 5.

$$\text{RMSE}_{\text{analog}}(k,p) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}_{t,i}^{(p)} - y_{t,i}\right)^2} \tag{6}$$

where N is the number of segments in the training set, $\hat{y}_{t,i}^{(p)}$ is the predicted dissipation for segment i at time t, and $y_{t,i}$ is the true dissipation.

In practice, we fix $t = \left\lfloor \frac{p}{2} \right\rfloor$, corresponding to the middle of the segment, and evaluate the RMSE across different k and p values.

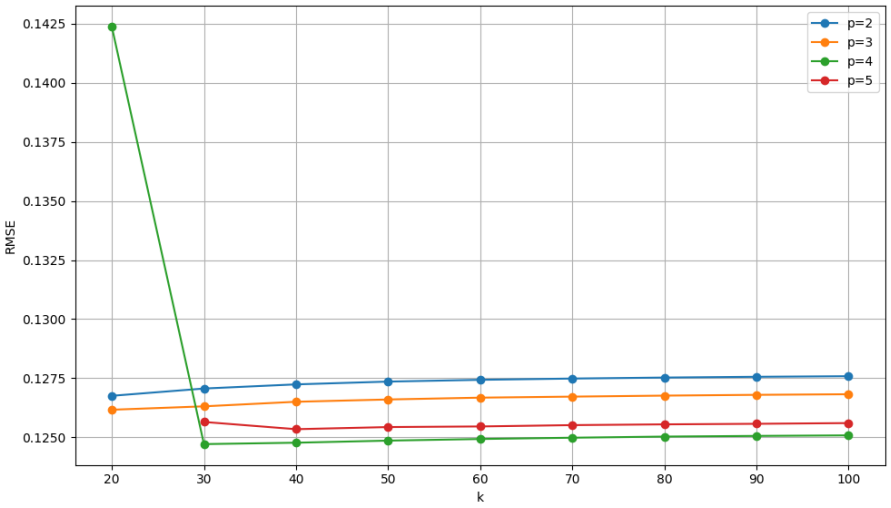


Figure 4: RMSE of the analog-based model for varying values of k and p.

We observe that the RMSE increases steadily with k and decreases with p. Furthermore, at least $6 \times p$ analogs are required to obtain a well-defined regression. Indeed, if $k < 6p$, then $\boldsymbol{X}_p$ has fewer rows than columns, causing its pseudo-inverse to be ill-defined.

Despite this, the analog-based model performs poorly, with an RMSE approximately equal to the standard deviation of dissipation (0.1). This implies that a trivial model always predicting zero would achieve a comparable RMSE.

This figure illustrate that the analog-based model doesn't work well.

Figure 7 displays the histogram of predicted dissipation versus true dissipation for $k = 100$ and $p = 3$.

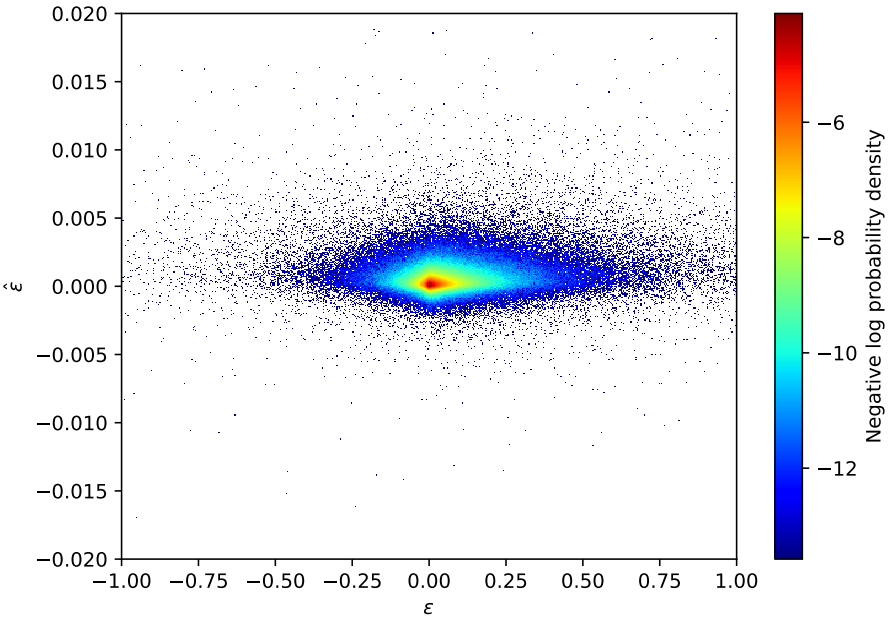


Figure 5: Histogram of predicted dissipation vs true dissipation for the analog-based model with $k = 100$ and $p = 3$.

The predicted dissipation is concentrated around zero. Moreover, the variance of predicted

dissipation is higher for true dissipation values close to zero.

To confirm this, we plot the histogram of the variance of predicted dissipation versus true dissipation for the same model parameters.

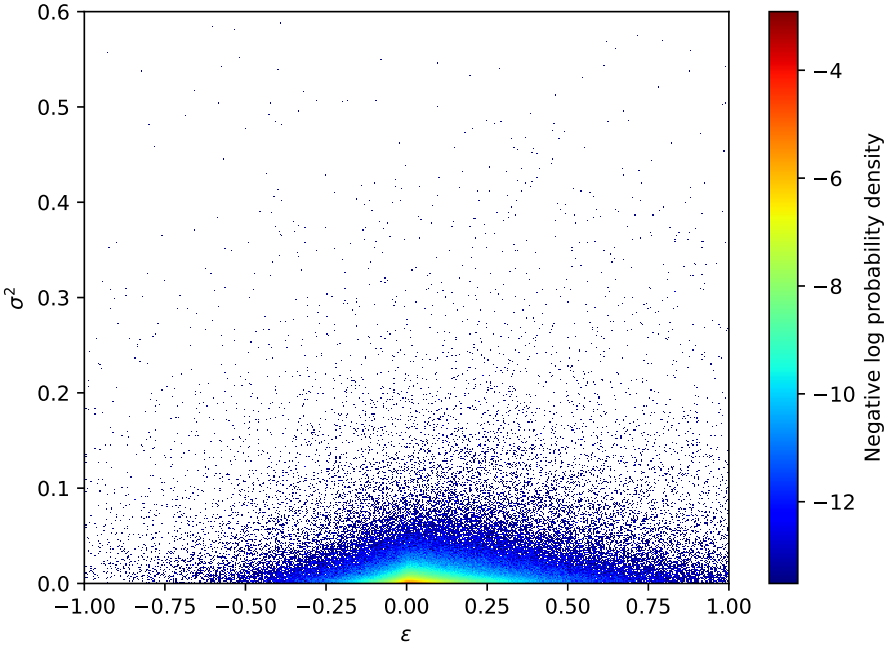


Figure 6: Histogram of variance among predicted dissipation vs true dissipation for the analog-based model with $k = 100$ and $p = 3$.

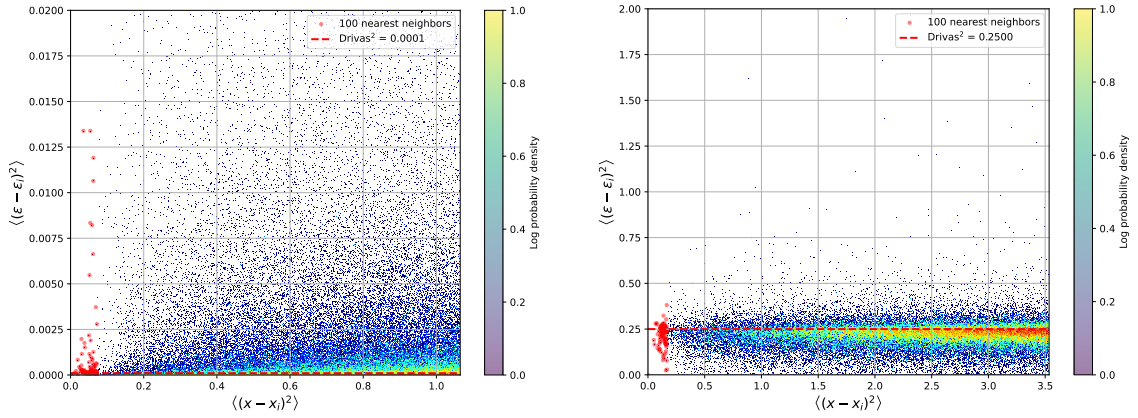
We observe that the variance of predicted dissipation is greater near zero dissipation values, confirming the previous observation.

2.3 Data exploration

After the failure of the analog-based model, we explored the data to understand why the model did not perform well.

We believe the main reason for the poor performance is that, around each point in phase space, there are many other points very close to it with dissipation values close to zero. Zero dissipation states are in all regions of the phase space. Extreme events are surrounded by zero dissipation events.

To illustrate this, we select points with different dissipation values and plot the distance to the 10,000 closest points in phase space, along with the absolute difference between the dissipation of the selected point and that of its closest neighbors.



(a) Dissipation = -0.01

(b) Dissipation = 0.5

(c) Dissipation = -1

(d) Dissipation = 2

Figure 7: Distance to the 10,000 closest points in phase space versus the absolute difference between dissipation values for points with different dissipation values.

We observe that most of the points are equally distributed around the line $y = \epsilon^2$, which means that the dissipation values of the closest points are close to 0.

We also observe that the 100 closest points in phase space exhibit significantly different dissipation values.

This confirms our hypothesis that the analog-based model fails to predict the dissipation because the closest point in phase space has a dissipation value close to zero, regardless of the dissipation value of the point we are trying to predict. However, we notice that the higher the dissipation value, the farther the closest points are in phase space, which suggests that there is some geometrical structure in the phase space that we can exploit to improve the model.

2.4 Data visualization

To better understand the data, we visualize the phase space of the segments in the training set by applying dimensionality reduction techniques—specifically, Principal Component Analysis (PCA) and t-SNE [10]—to project the high-dimensional data into two dimensions. This visualization aims to reveal any underlying geometrical structures in the phase space that could be leveraged to enhance model performance.

We begin by applying PCA to the segments in the training set. The resulting two principal components are discretized into five dissipation categories. The segments are then plotted in the PCA space and color-coded according to their dissipation category.

The dissipation thresholds used to define these categories are derived from the cumulative distribution of dissipation values, as shown in the figure below.

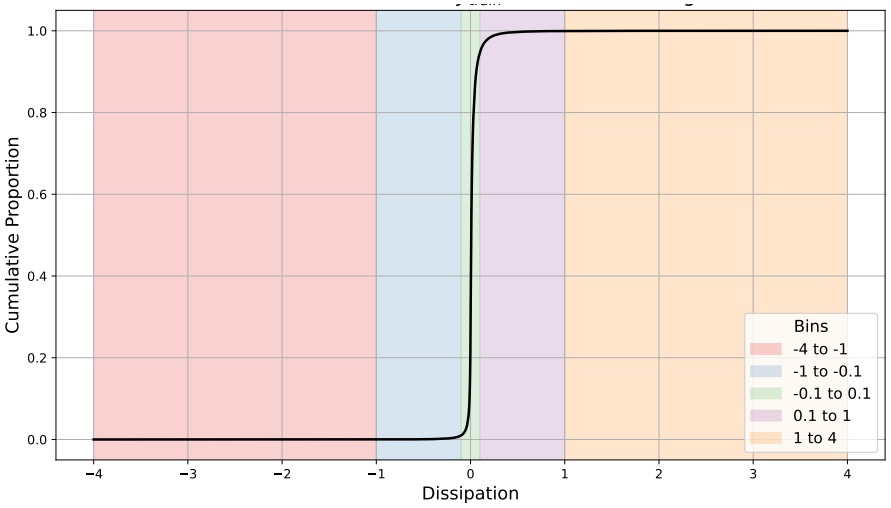


Figure 8: Cumulative distribution of dissipation values with categories in the background.

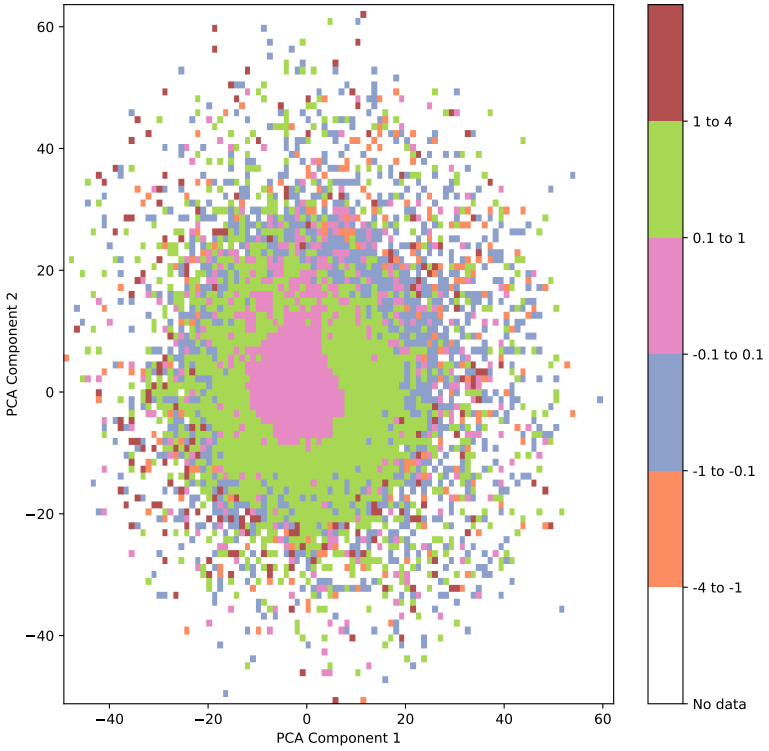


Figure 9: PCA of the segments in the training set, colored according to their dissipation category.

We observe that there is some structure in the phase space, as the segments with low dissipation values are clustered together, while the segments with high dissipation values are more spread out.

Next, we do the same with t-SNE.

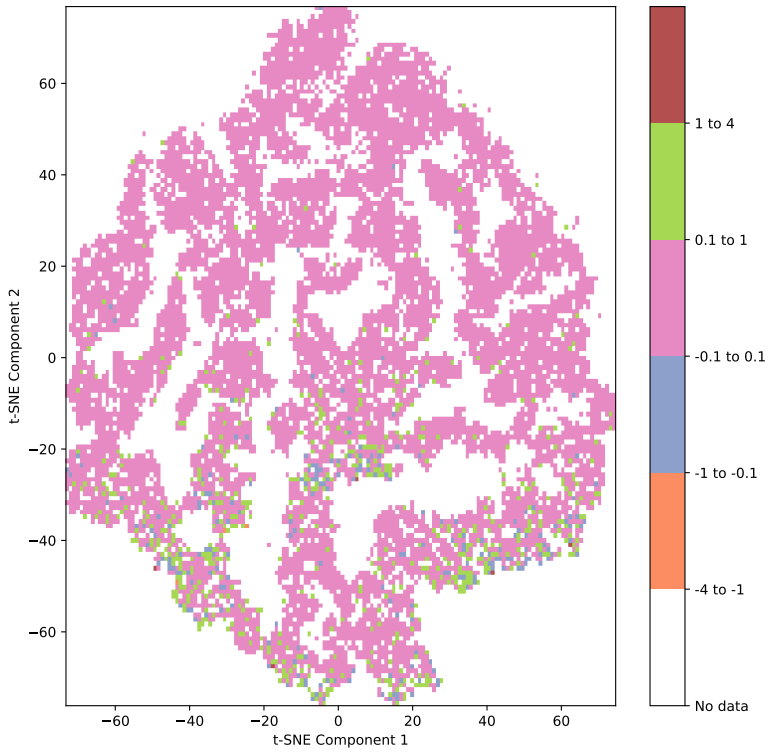


Figure 10: t-SNE of the segments in the training set, colored according to their dissipation category.

We observe that t-SNE does not reveal any additional structure compared to PCA and even seems to hide the structure that was visible in the PCA representation.

3 Distance Learning Model

3.1 Distance Learning with Mahalanobis Distance

Based on the previous analysis, we hypothesize that modifying the distance metric used to compute the k nearest neighbors can enhance the performance of the analog-based model. To this end, we adopt the framework proposed in [7] to learn a data-adaptive distance metric.

The core idea is to optimize a Mahalanobis distance matrix $\boldsymbol{A}$ such that the mean squared error (MSE) between the average dissipation of the k nearest neighbors of a given segment and the dissipation of the segment itself is minimized.

For each segment x in the training set, we define the weighting of its neighbors as follows:

$$p(x_j \mid x) = \frac{\exp\left(-\|A(x - x_j)\|^2\right)}{\sum_{k \in I(x)} \exp\left(-\|A(x - x_k)\|^2\right)} \tag{7}$$

where $I(x)$ denotes the index set of neighboring segments. This expression assigns a weight to each neighbor x_j based on its Mahalanobis distance to x.

We define the prediction as the weighted average of the dissipation values of the k nearest neighbors:

$$\hat{y}(x) = \sum_{j \in I(x)} p(x_j \mid x) y_j \tag{8}$$

and aim to minimize the following loss function:

$$\mathcal{L}(\boldsymbol{A}) = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{y}(x_i) - y_i\right)^2 \tag{9}$$

where N is the number of segments in the training set, x_i is the i-th segment, and y_i is its corresponding dissipation value.

We compute the gradient of the loss function with respect to the distance matrix $\boldsymbol{A}$ as:

$$\frac{\partial \text{MSE}}{\partial A} = \frac{2}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^T \sum_{k \in I(i)} \frac{\partial p(k \mid i)}{\partial A} y_k \tag{10}$$

with

$$\frac{\partial p(k \mid i)}{\partial A} = A\, p(k \mid i) \left(\sum_{l \in I(i)} p(l \mid i)\, \mathbf{x}_{il} \mathbf{x}_{il}^T - \mathbf{x}_{ik} \mathbf{x}_{ik}^T \right) \tag{11}$$

where $\mathbf{x}_{il} = \mathbf{x}_l - \mathbf{x}_i$ is the difference between segments i and l.

We acknowledge the authors of [7] for providing an implementation to compute the gradient of the loss function with respect to the matrix $\boldsymbol{A}$. Although we adapted their code to our setting, it did not perform as expected: the loss function did not decrease, and the matrix $\boldsymbol{A}$ remained unchanged even when increasing the learning rate.

We hypothesize that this issue stems from using a weighted average of the dissipation values of the k nearest neighbors, which has been shown to be a poor predictor of dissipation itself, as discussed in [4].

To address this limitation, we propose using a weighted linear regression model instead of a weighted average.

As before, we compute the gradient of the loss function with respect to the distance matrix $\boldsymbol{A}$:

$$\frac{\partial \text{MSE}}{\partial A} = \frac{2}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^T \frac{\partial \hat{y}_i}{\partial A} \tag{12}$$

where $\hat{y}_i$ denotes the predicted dissipation for segment i, obtained via a weighted linear regression model:

$$\hat{y}_i = \hat{\beta}_i^T x_i \tag{13}$$

Here, $\hat{\beta}_i$ is the vector of regression coefficients, which depends on both the distance matrix $\boldsymbol{A}$ and the segment i:

$$\hat{\beta}_i = \arg\min_{\beta} \sum_{j=1}^{k} \left(y_j - \mathbf{X}_j^\top \beta \right) P_{ij} \left(y_j - \mathbf{X}_j^\top \beta \right) \tag{14}$$

where $P_{ij} = p(x_j \mid x_i)$ is the weight of segment j relative to segment i, as defined in Equation (7), and $\mathbf{X}_j$ represents the embedding of segment j in the phase space.

The solution to this minimization problem is given by:

$$\hat{\beta}_i = \left(\sum_{j=1}^{k} P_{ij} \mathbf{X}_j \mathbf{X}_j^T \right)^{-1} \sum_{j=1}^{k} P_{ij} \mathbf{X}_j y_j \tag{15}$$

Let us define:

$$A_i = \sum_{j=1}^{k} P_{ij} \mathbf{X}_j \mathbf{X}_j^\top \tag{16}$$

$$b_i = \sum_{j=1}^{k} P_{ij} \mathbf{X}_j y_j \tag{17}$$

so that $\hat{\beta}_i = A_i^{-1} b_i$.

We differentiate $\hat{\beta}_i = A_i^{-1} b_i$ using the identity:

$$\frac{\partial (A^{-1} b)}{\partial \boldsymbol{A}} = -A^{-1} \left(\frac{\partial A}{\partial \boldsymbol{A}} \right) A^{-1} b + A^{-1} \frac{\partial b}{\partial \boldsymbol{A}} \tag{18}$$

which yields:

$$\frac{\partial \hat{\beta}_i}{\partial \boldsymbol{A}} = -A_i^{-1} \left(\frac{\partial A_i}{\partial \boldsymbol{A}} \right) \hat{\beta}_i + A_i^{-1} \left(\frac{\partial b_i}{\partial \boldsymbol{A}} \right) \tag{19}$$

The derivatives of A_i and b_i are given by:

$$\frac{\partial A_i}{\partial \boldsymbol{A}} = \sum_{j=1}^{k} \frac{\partial P_{ij}}{\partial \boldsymbol{A}} \mathbf{X}_j \mathbf{X}_j^\top \tag{20}$$

$$\frac{\partial b_i}{\partial \boldsymbol{A}} = \sum_{j=1}^{k} \frac{\partial P_{ij}}{\partial \boldsymbol{A}} \mathbf{X}_j y_j \tag{21}$$

and thus:

$$\frac{\partial \hat{\beta}_i}{\partial \boldsymbol{A}} = -A_i^{-1} \left(\sum_{j=1}^{k} \frac{\partial P_{ij}}{\partial \boldsymbol{A}} \mathbf{X}_j \mathbf{X}_j^\top \right) \hat{\beta}_i + A_i^{-1} \left(\sum_{j=1}^{k} \frac{\partial P_{ij}}{\partial \boldsymbol{A}} \mathbf{X}_j y_j \right) \tag{22}$$

with $\frac{\partial P_{ij}}{\partial \boldsymbol{A}}$ as defined in Equation (11).

Despite successfully deriving the gradient of the loss function with respect to the matrix $\boldsymbol{A}$, its computation is prohibitively expensive in practice. Indeed we have 3 loop in the computation of the gradient:

- the outer loop iterates over all segments in the training set, (4 million segments),
- the middle loop iterates over the k nearest neighbors of each segment (up to 100 neighbors),
- the inner loop iterates over the k nearest neighbors of each segment to compute the distance between the segment and its neighbors (up to 100 neighbors).

This results in a complexity of $O(N \cdot k^2)$, where N is the number of segments in the training set and k is the number of nearest neighbors. Therefore, we investigate alternative strategies to learn the distance matrix $\boldsymbol{A}$.

3.2 Distance Learning with an Autoencoder

We explore another approach to learn a distance that is adapted to the data. We use an autoencoder to learn a representation of the data that is tailored to the dissipation.

To do so, we train an autoencoder with the following loss function:

$$\mathcal{L}(f, g) = \frac{1}{N} \sum_{i=1}^{N} \|g(f(x_i)) - x_i\|^2 + \lambda \cdot \mathcal{L}_{\text{triplet}}(f) \tag{23}$$

where:

- f is the encoder,
- g is the decoder,
- $\|g(f(x_i)) - x_i\|^2$ is the reconstruction loss,
- $\mathcal{L}_{\text{triplet}}(f)$ is the triplet loss, defined as:

$$\mathcal{L}_{\text{triplet}}(f) = \frac{1}{N} \sum_{i=1}^{N} \left[\sum_{j=1}^{k} \left\| f(x_i) - f(x_j^+) \right\|_2 - \left\| f(x_i) - f(x_j^-) \right\|_2 + \alpha \right]_+ \tag{24}$$

where:

- x_j^+ is a positive sample (with similar dissipation to x_i),
- x_j^- is a negative sample (with different dissipation from x_i),

- α is the margin,
- $[\cdot]_+$ denotes the hinge loss (i.e., $\max(0, \cdot)$).

The goal of this loss function is to learn a representation that minimizes the reconstruction error while ensuring that segments with similar dissipation values are close in the latent space, and segments with different dissipation values are far apart.

We train the autoencoder on the training set, which consists of all segments from Runs 1 to 5.

In addition to the triplet loss used to encourage separation in the latent space, we introduce a *consistency loss* to assess how well the learned representation preserves the structure of the physical variable of interest, namely the dissipation.

Let $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ denote the dataset, where $\mathbf{x}_i \in \mathbb{R}^D$ is an input segment and $y_i \in \mathbb{R}$ its associated dissipation. Let $f : \mathbb{R}^D \to \mathbb{R}^d$ be the encoder mapping to a latent representation $\mathbf{z}_i = f(\mathbf{x}_i)$. For each point i, let $\mathcal{N}_k(i)$ be the set of its k nearest neighbors in the latent space, excluding i itself.

The consistency loss is defined as:

$$\mathcal{L}_{\text{cons}} = \frac{1}{Nk} \sum_{i=1}^{N} \sum_{j \in \mathcal{N}_k(i)} |y_j - y_i| \tag{25}$$

This loss penalizes discrepancies in dissipation values among neighboring points in the latent space, encouraging smoothness and local coherence of the physical quantity across the latent manifold.

We use this loss on the test set to assess the encoder's performance.

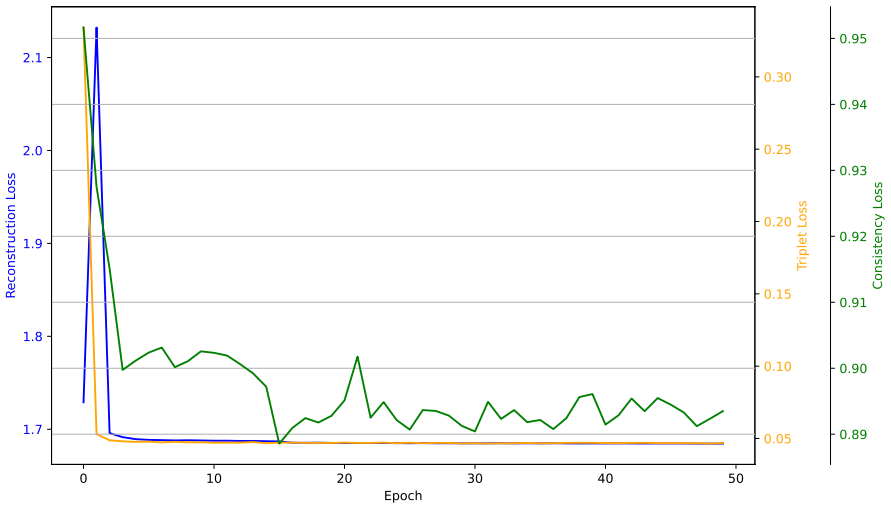


Figure 11: Training loss of the autoencoder.

We observe that during the first epoch, the triplet loss ,the reconstruction loss and the consistency loss decrease. However, after the first few epochs, both the triplet loss and the reconstruction loss remain constant, while the consistency loss fluctuates.

We believe that the autoencoder does not learn a useful representation of the data.

To verify this, we use an analog-based regression model with the learned representation as input.

We obtain the same poor performance metric as before, indicating that the autoencoder fails to learn a meaningful representation of the data.

We believe this is due to suboptimal training of the autoencoder, and that more careful training could lead to better results.

However, we do not aim to design an overly complex model, as we wish to maintain interpretability in order to gain insights into the underlying physical processes.

4 Uniform Manifold Approximation and Projection (UMAP)

4.1 Motivation for Using UMAP

We noticed earlier that with PCA we could observe some structure in the phase space of the segments in the training set. This suggests that there is some geometric structure in the phase space that we can exploit to improve the model. However, we failed to use this structure with the analog-based model and the distance learning model.

To further explore this structure, we use UMAP.

UMAP is a dimensionality reduction technique based on manifold learning. So basically, UMAP learns the manifold structure of the data and projects it into a lower-dimensional space while preserving the local structure of the data. This is particularly useful because it allows us to use the geodesic distance in the learned manifold to compute the k nearest neighbors of a segment in the phase space, which may be more adapted to the data than the Euclidean distance.

To gain a better understanding of the UMAP algorithm, we refer to the original paper [6] or to the documentation How UMAP works.

4.2 UMAP Results

We then filter the segments based on their dissipation values and plot them in the UMAP space, coloring them according to their dissipation category.

The first plot on the left shows the UMAP representation of all segments in the training set, colored according to their dissipation category. The plot in the middle shows the UMAP representation of the segments with dissipation values between -0.5 and 0.5, also colored by their dissipation category. The plot on the right shows the UMAP representation of the segments with dissipation values outside the interval $[-0.5, 0.5]$.

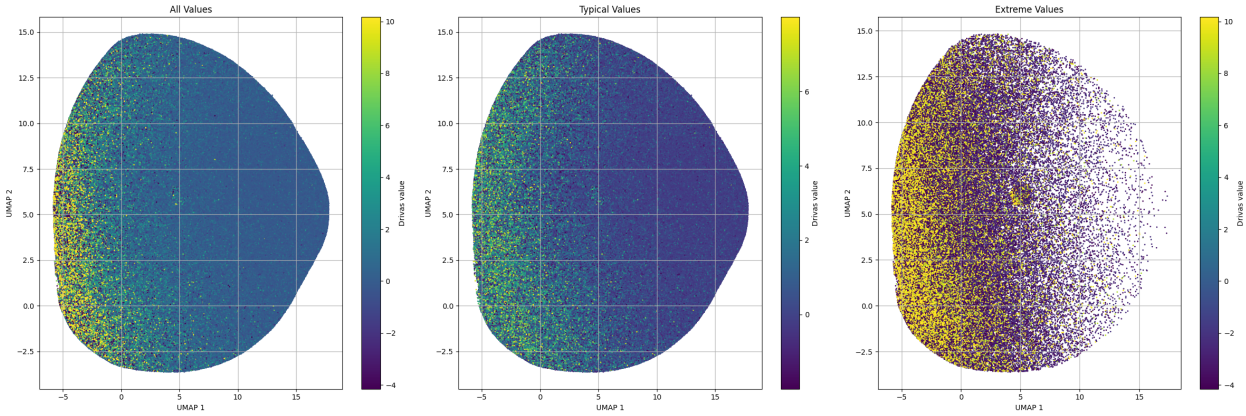


Figure 12: UMAP Representation of Training Segments by Dissipation Category

The value associated with the color is normalized according to: $\frac{\epsilon - 0.012}{0.04}$

We notice that UMAP captures some sort of topological structure in the data.

To ensure that UMAP can be used to estimate the dissipation, we separate the range of dissipation into 5 categories, as we did for the PCA plot. We partition the first component of the UMAP embedding, and in each bin, we calculate the proportion of each category.

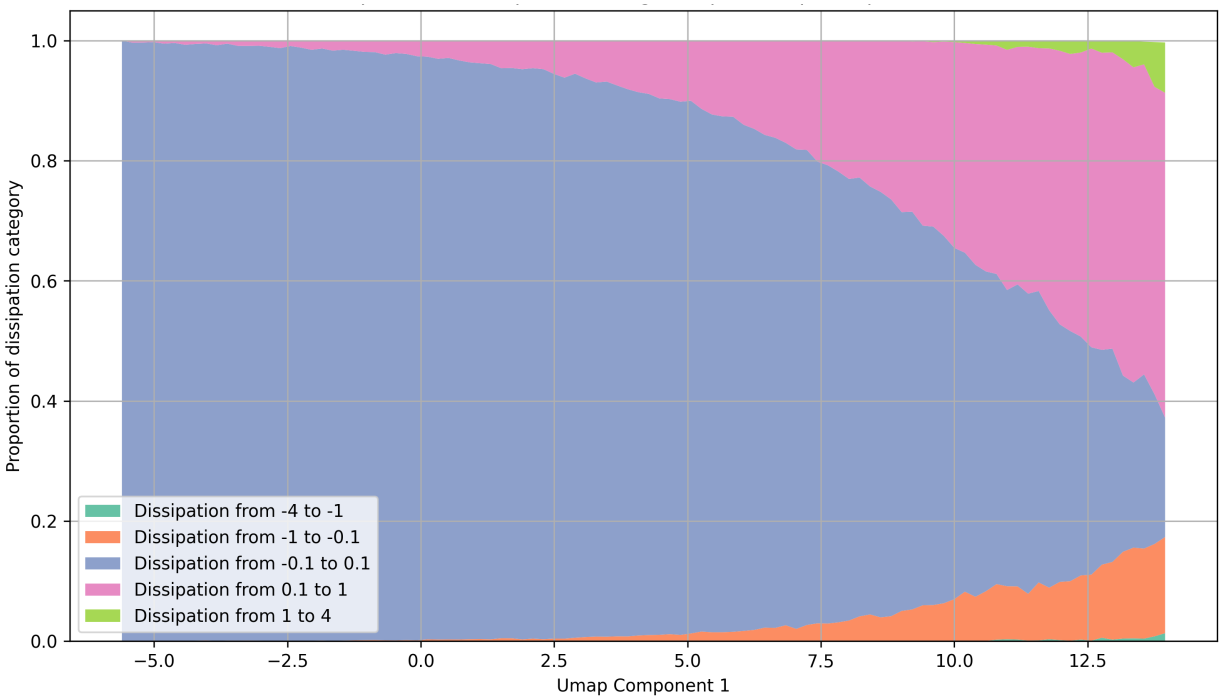


Figure 13: Proportion of Dissipation Categories for UMAP Component 1

We observe that the segments with low dissipation values are more concentrated on the left side of the UMAP embedding, while the segments with high dissipation values are more concentrated on the right side. This suggests that UMAP captures some kind of topological structure in the data, which can be exploited to estimate the dissipation.

We can also use the UMAP algorithm with higher dimensions. For instance, if we use UMAP to reduce to dimension 10, we still observe some structure in the data that we can exploit to estimate the dissipation.

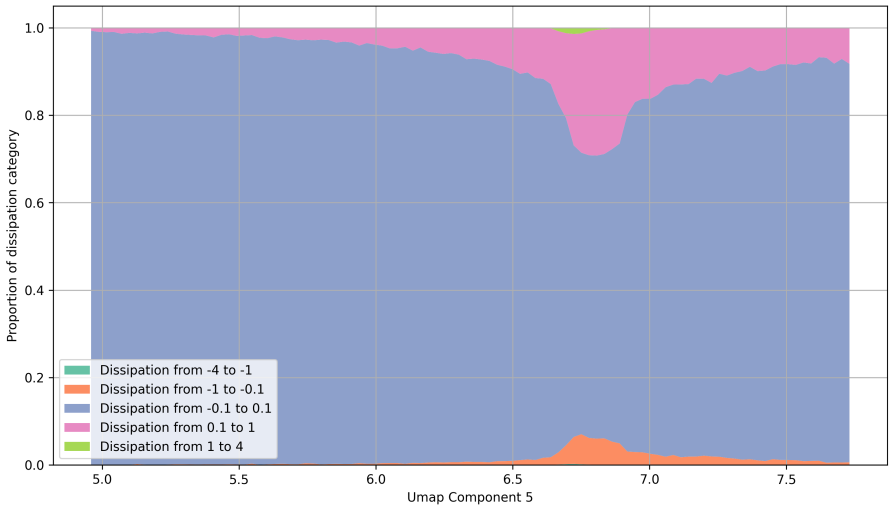


Figure 14: Proportion of Dissipation Categories for UMAP Component 5

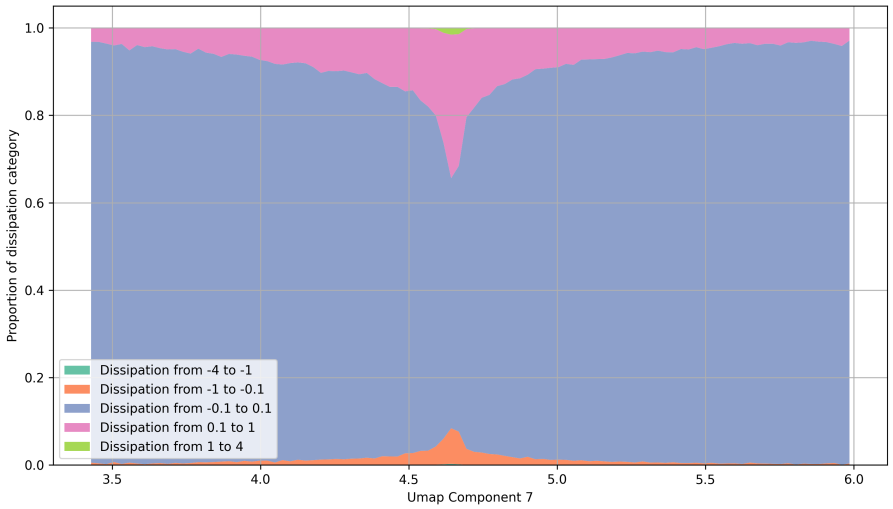


Figure 15: Proportion of Dissipation Categories for UMAP Component 7

We notice in the figure above that when the proportion of segments with dissipation values between 0.1 and 1 increases, the proportion of segments with dissipation values between 1 and 4 also increases.

So we conclude that there is some structure in the UMAP embedding that can be exploited to detect extreme events of the dissipation, and that there are some zones in the manifold where the dissipation values are strong in absolute value.

We can therefore conclude that there are some zones in the manifold that are linked to extreme events, i.e., segments with high dissipation values.

However, we have an issue with the UMAP algorithm: if we insert a new segment into the phase space, we cannot compute its UMAP embedding without recomputing the entire UMAP embedding of the training set. This difficult the use of UMAP for classification or regression purposes. To counter this issue, we can learn the UMAP embedding of the training set with a neural network, and then use this neural network to compute the UMAP embedding of the new segment without recomputing the whole embedding.

After training this neural network, we can use it to compute the UMAP embedding of a new segment and then use a regression model to estimate the dissipation of the new segment based on its UMAP embedding.

5 Parametric UMAP

5.1 Training a Parametric UMAP Model

To better understand what parametric UMAP is, we refer to the original paper [9] or to the documentation Parametric UMAP.

We train a parametric UMAP model to learn the UMAP embedding of the segments in the training set. We use the UMAP implementation from the Python package `umap`.

We reduce the dimensionality to 10 to obtain 10 components.

We plot the training loss of the parametric UMAP model, which is cross-entropy:

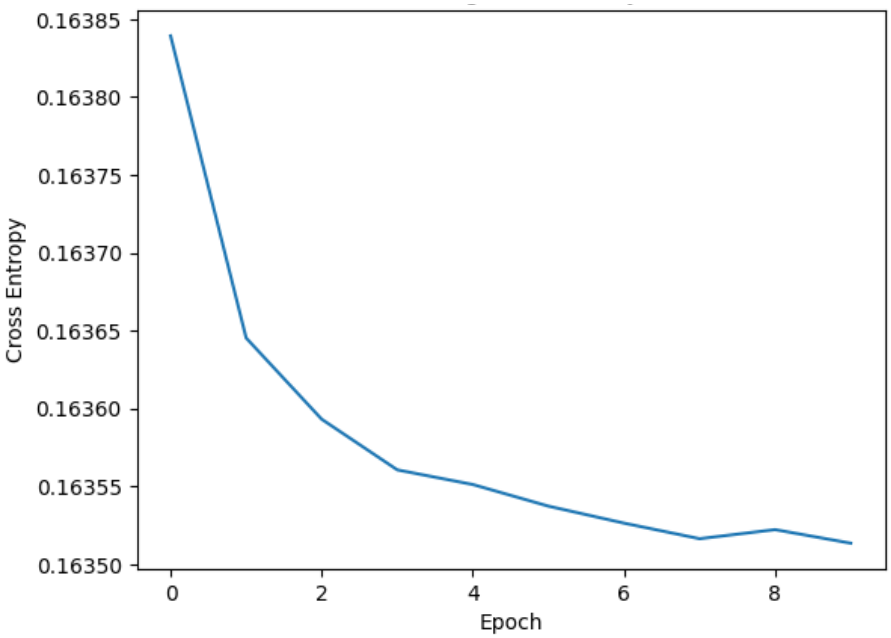


Figure 16: Training loss of the parametric UMAP model.

We plot the proportion plot as before to verify that we observe the same behavior:

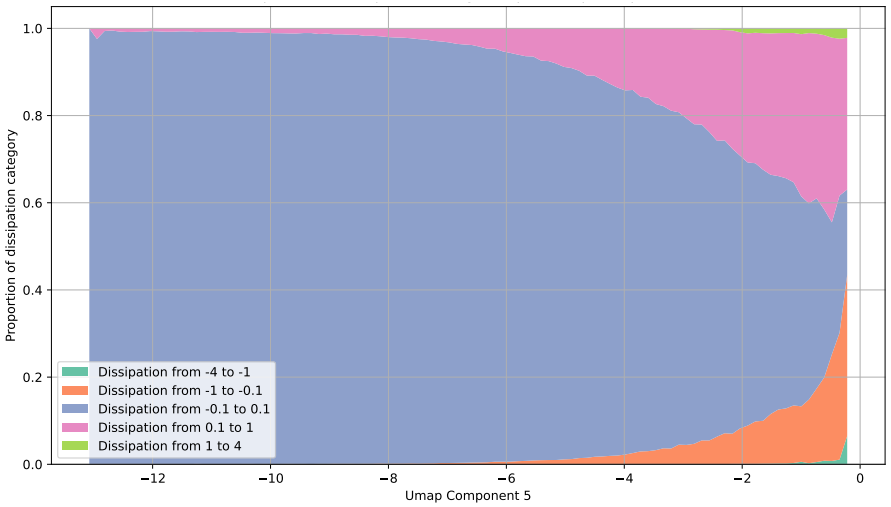


Figure 17: Proportion of dissipation categories for the fifth component of the parametric UMAP embedding.

We conclude that the parametric UMAP has the same behavior as the standard UMAP algorithm, and that we can use it to estimate the dissipation of a new segment.

5.2 Using Parametric UMAP for Regression

We simplyfy the regression task by binning the dissipation values into 5 categories, as we did for the PCA and UMAP plots. We recall that the dissipation values are in the range $[-4, 4]$, and we define the following categories:

Table 2: Definition of Dissipation Categories

Category	Dissipation Range
0	$[-4,\ -1)$
1	$[-1,\ -0.1)$
2	$[-0.1,\ 0.1)$
3	$[0.1,\ 1)$
4	$[1,\ 4)$

we use a random forest classifier to predict the dissipation category of a segment based on its UMAP embedding.

So the full pipline is as follows:

- We compute the UMAP embedding of the segments in the training set using the parametric UMAP model.
- We bin the dissipation values into 5 categories.
- We train a random forest classifier to predict the dissipation category of a segment based on its UMAP embedding.
- We use the trained random forest classifier to predict the dissipation category of a new segment based on its UMAP embedding.

5.3 Results

We begin by evaluating the performance of the Random Forest classifier with a confusion matrix.

To recall, the confusion matrix is a table that summarizes the performance of a classification model by comparing the predicted categories with the true categories.

Mathematically, for a multi-class classification problem with K classes, the confusion matrix $\mathbf{C} \in \mathbb{N}^{K \times K}$ is defined by:

$$C_{i,j} = \sum_{n=1}^{N} \mathbf{1}_{\{y_n = i \wedge \hat{y}_n = j\}}$$

where:

- N is the total number of samples,
- y_n is the true class label of the n-th sample,
- $\hat{y}_n$ is the predicted class label for the n-th sample,
- $\mathbf{1}_{\{\cdot\}}$ is the indicator function, which equals 1 if the condition inside the brackets is true, and 0 otherwise,
- $C_{i,j}$ counts the number of times a sample from true class i was predicted as class j.

Thus, the diagonal entries $C_{i,i}$ represent correct predictions (true positives per class), while the off-diagonal entries indicate misclassifications.

We normalized the confusion matrix by dividing each entry by the total number of samples in the corresponding true class, which allows us to interpret the values as proportions rather than raw counts.

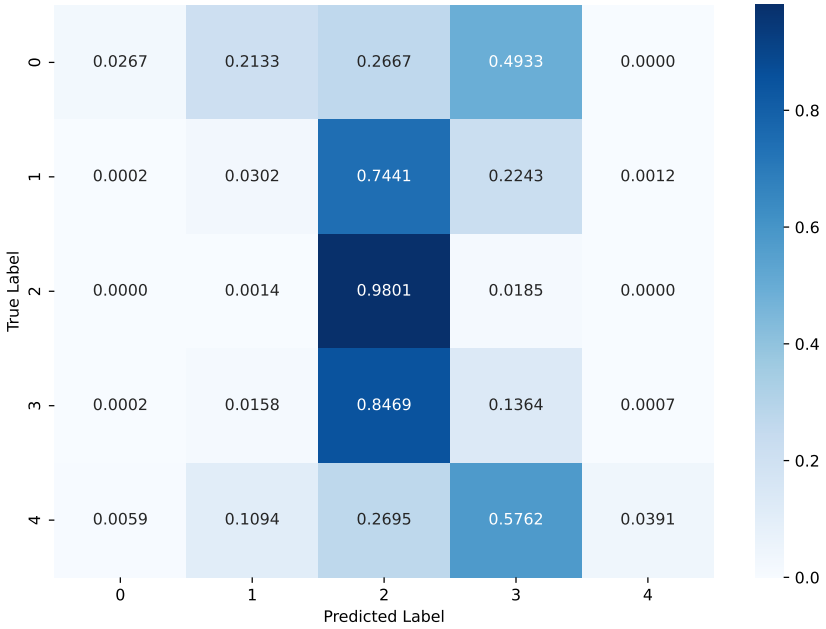


Figure 18: Confusion matrix of the Random Forest classifier on the test set.

We also plot the recall- precision curve, which is a plot of the precision and recall of the classifier for each class.

We remind that the precision and recall are defined as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$
$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

where:

- TP (True Positive) is the number of samples correctly predicted as belonging to a class,
- FP (False Positive) is the number of samples incorrectly predicted as belonging to a class,
- FN (False Negative) is the number of samples that belong to a class but were not predicted as such.

So the recall mesure the ability of the classifier to find all the samples that belong to a class, while the precision measures the ability of the classifier to not predict samples that do not belong to a class.

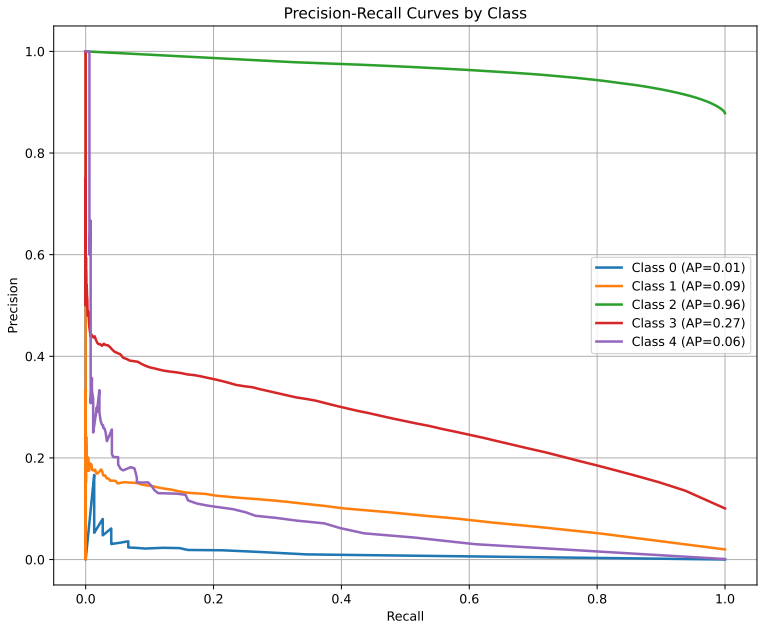


Figure 19: Precision-Recall curve of the Random Forest classifier on the test set.

We noticed that when the recall increases, the precision decreases sharply for each class except for class 2, which has a high precision and recall. This indicates that the classifier is able to find all the samples that belong to class 2, but it is not able to find all the samples that belong to the other classes without also predicting many samples that do not belong to these classes.

This is the same behavior we observed in the regression model, where the model was able to predict the dissipation values close to zero, but it was not able to predict the dissipation values far from zero without also predicting many samples that do not belong to these classes.

We also compute the classification report, which summarizes the precision, recall, and F1-score for each class.

Dissipation Range	Precision	Recall	F1-score	Support
$[-4, -1)$	0.111	0.027	0.043	75
$[-1, -0.1)$	0.171	0.030	0.051	10 053
$[-0.1, 0.1)$	0.896	0.980	0.936	442 908
$[0.1, 1)$	0.391	0.136	0.202	50 732
$[1, 4)$	0.260	0.039	0.068	512

Table 3: Classification report metrics for each dissipation category.

Overall, the model fails to distinguish between dissipation categories, assigning the majority of samples—regardless of their true class—to the "normal event" category. This indicates a significant misclassification across multiple categories.

6 Conclusion

Over the course of this four-month internship, we investigated several approaches for estimating the dissipation of turbulent velocity fields using analog-based models, distance learning techniques, and manifold learning methods, including UMAP and parametric UMAP.

We began by implementing an analog-based model, which did not achieve satisfactory accuracy in predicting dissipation. In an effort to improve its performance, we explored distance learning approaches—specifically, learning a Mahalanobis distance matrix to refine similarity metrics. However, this method proved computationally expensive and ultimately ineffective in enhancing the model's predictive capabilities.

Subsequently, we experimented with autoencoders to learn a low-dimensional representation of the data. Unfortunately, this approach also failed to yield meaningful improvements.

Our focus then shifted to manifold learning techniques. Using UMAP and parametric UMAP, we aimed to uncover latent structures within the data's phase space. We found that UMAP successfully captured aspects of the topological structure, which could be leveraged for dissipation estimation. Building on this insight, we trained a parametric UMAP model to learn the UMAP embedding of segments from the training set and used it to classify the dissipation category of new segments.

The classification was done with a random forest classifier. To asses the performance of the model, we employed confusion matrices and precision–recall curves. The results indicated that the model performs well for segments with dissipation values near zero, but struggles to accurately classify high-dissipation segments without also producing a significant number of false positives. This behavior is reminiscent of the limitations observed in the analog-based model.

Nonetheless, our findings suggest that UMAP-based approaches are capable of capturing meaningful topological features in the data. We believe that further research in this direction could improve dissipation prediction and provide deeper insights into the geometry of turbulent flows.

7 Acknowledgments

I would like to thank my internship supervisor, Carlos Granero-Belinchón, for his guidance and support throughout the internship. I also thank Dominique Pastor, as well as other professors and researchers, for their valuable feedback and discussions, which contributed to deepening my understanding of the subject and sparked my interest in research.

References

[1] Adam Cheminet, Damien Geneste, Antoine Barlet, Yasar Ostovan, Tarek Chaabo, Valentina Valori, Paul Debue, Christophe Cuvier, Fran çois Daviaud, Jean-Marc Foucaut, Jean-Philippe Laval, Vincent Padilla, Cécile Wiertel-Gasquet, and Bérengère Dubrulle. Eulerian vs lagrangian irreversibility in an experimental turbulent swirling flow. *Phys. Rev. Lett.*, 129:124501, Sep 2022.

[2] Adam Cheminet, Damien Geneste, Antoine Barlet, Yasar Ostovan, Tarek Chaabo, Valentina Valori, Paul Debue, Christophe Cuvier, Fran çois Daviaud, Jean-Marc Foucaut, Jean-Philippe Laval, Vincent Padilla, Cécile Wiertel-Gasquet, and Bérengère Dubrulle. Eulerian vs lagrangian irreversibility in an experimental turbulent swirling flow , supplementary materials. *Phys. Rev. Lett.*, 129:124501, Sep 2022.

[3] Laurent Chevillard, Bernard Castaing, Alain Arneodo, Emmanuel Lévêque, Jean-François Pinton, and Stéphane G. Roux. A phenomenological theory of eulerian and lagrangian velocity fluctuations in turbulent flows. *Comptes Rendus Physique*, 13(9):899–928, 2012. Structures and statistics of fluid turbulence/Structures et statistiques de la turbulence des fluides.

[4] Ewen Frogé, Carlos Granero-Belinchon, Stéphane G. Roux, Nicolas B. Garnier, and Thierry Chonavel. Analog-based forecasting of turbulent velocity: Relationship between unpredictability and intermittency. 2024.

[5] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Phys. Rev. E*, 69:066138, Jun 2004.

[6] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. 2020.

[7] Paul Platzer, Arthur Avenas, Bertrand Chapron, Lucas Drumetz, Alexis Mouche, Pierre Tandeo, and Léo Vinour. Distance Learning for Analog Methods. October 2024. working paper or preprint.

[8] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

[9] Tim Sainburg, Leland McInnes, and Timothy Q Gentner. Parametric umap embeddings for representation and semisupervised learning. *Neural Computation*, 33(11):2881–2907, 2021.

[10] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.