

LATENT SPACE FORECASTING: AN INTEGRATED VAE-LSTM FRAMEWORK FOR END-TO-END SEQUENTIAL RECONSTRUCTION

R. Rajeev

Institute of Mechanical and Biomedical Engineering
Riga Technical University,
6a Kļipsalas Str., Riga, LV-1048, LATVIA
E-mail: rajeev.ralla@edu.rtu.lv

The forecasting of high-fidelity, transient fluid flows remains a formidable computational challenge due to the high-dimensional and non-linear nature of the data produced by numerical simulations. The study presents a novel hybrid deep-learning framework to address this challenge by combining a Variational Autoencoder (VAE) with a Long Short-Term Memory (LSTM) network for the efficient spatiotemporal prediction of flow fields around a NACA 0012 airfoil. The VAE first learns a compressed, low-dimensional latent representation of the high-dimensional flow data, which includes pressure and velocity components, achieving effective dimensionality reduction while preserving essential spatial features. An LSTM network then learns the temporal dynamics within this latent space to forecast future states. The model was trained and validated on a dataset of 300 time-step snapshots, successfully generating a 15-step-ahead forecast. Results demonstrate the model's high accuracy, with a key metric – the maximum velocity magnitude – predicted with only a 3.8 % error compared to the ground truth simulation data. This study validates the VAE-LSTM architecture as a powerful and computationally efficient tool for forecasting complex fluid dynamics, offering significant potential for applications in real-time control and design optimisation where rapid prediction is critical.

Keywords: *Hybrid deep-learning framework, long short-term memory network, variational autoencoder.*

1. INTRODUCTION

Time series dataset from numerical simulations becomes very complex as it often involves high-dimensional, non-linear, and spatial temporal features. It is very challenging to handle time-dependent problems that require high computational resources, simultaneously capturing spatial patterns of flow and their evolution over time. Forecasting or predicting the high dimensional and non-linear dataset becomes increasingly challenging using conventional machine learning methods due to high dimensionality. Dimensionality reduction of the dataset can significantly alleviate the performance of the model by learning a lower-dimensional representation, which reduces the computational demand by projecting the data into a compact representation [1]. In this study, the numerical simulation was conducted over airfoil (NACA 0012) for 300 time steps, and three flow field features – Pressure (Pa), Velocity -X (m/s), and Velocity-Y (m/s) – were extracted. The resulting dataset comprised (300, 86909, 3) values – spatial points and their flow field features – and occupied 7.40 GB of space. This dataset is generated by the governing equations of the fluid flow, incorporating the non-linearity and pattern evolution over time. To address the modelling challenges posed by such large and complex data, hybrid models were required to reduce the model complexity and utilise the reduced space for forecasting the flow patterns in time without losing significant trends in the original dataset. Variational Autoencoder (VAE) is a special class of generative models learning a probabilistic mapping from high dimensional input data to a lower dimensional latent space, maintaining the continuity and meaningful representation over the latent space [2]. Long Short-Term Memory (LSTM) is an extension of Recur-

rent Neural Network (RNN) capable of generating forecasting models, particularly well-suited to forecasting the fluid flow evolution of the flow patterns in time [3]. After forecasting the latent vectors for future time steps, the decoder component of the VAE reconstructs the full flow field from these latent predictions. This end-to-end VAE-LSTM framework is efficient in complex spatiotemporal flow using reduced number of parameters. Moreover, it offers a hybrid pipeline and provides forecasting via short memory LSTM, direct access to variational latent space manipulation. It ensures conventional forecasting approaches in terms of scalability and computational memory efficiency. To evaluate the performance of the integrated VAE-LSTM framework, a combination of loss metrics was used. In the Variational Autoencoder, the total loss was made up of two key parts: the Kullback-Leibler (KL) divergence [4], which helps shape the latent space by encouraging the learned distribution to stay close to a standard normal distribution, and the reconstruction loss, measured using Mean Squared Error (MSE), which reflects how well the decoder can rebuild the original flow field from the latent variables. For the LSTM component, the loss was also based on MSE, calculated by comparing the predicted latent vectors to the actual ones over time. This allows the model to learn how the compressed flow features evolve step by step. Together, these loss functions drive the training process, ensuring the model not only captures meaningful spatial features but also accurately predicts how they change over time, thereby making it a powerful approach for forecasting complex, high-dimensional, and time-dependent flow data.

2. GEOMETRY PREPARATION

A two-dimensional transient simulation was performed to investigate the unsteady flow behaviour around the airfoil (NACA 0012) geometry. The inlet velocity of 20 m/s in the X-direction depicts the moderate flow conditions. Air is used as fluid with standard properties: density of 1.225 kg/m^3 and the effect of gravity acting on the negative Y-direction of 9.81 m/s^2 . The flow was resolved over 300 transient time

steps, with each step corresponding to 0.1 seconds, with a total of 30 seconds performed in the SolidWorks flow simulation. During the simulation, essential flow field variables were recorded including pressure (Pa), velocity-X (m/s) and velocity-Y (m/s) at each time step. The simulation provides a comprehensive dataset for modelling and implementing machine learning techniques.

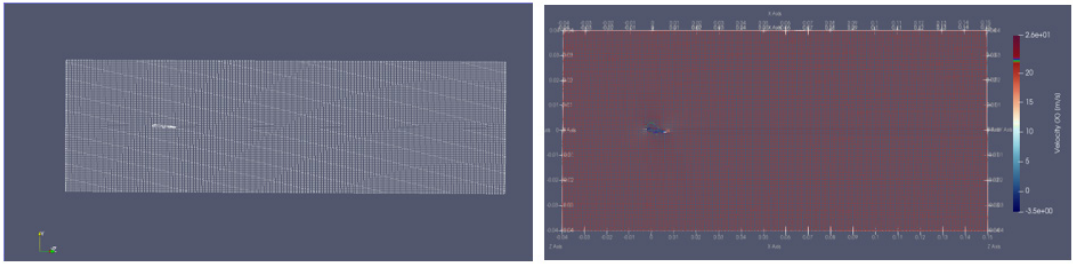


Fig. 1. Mesh of the domain (Left), the velocity-X direction (Right) at $t = 0.1 \text{ sec}$.

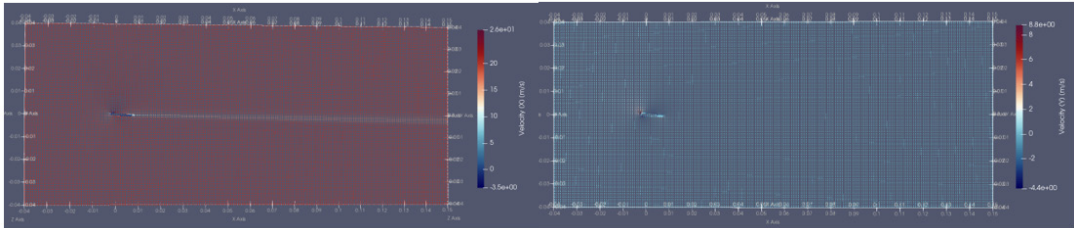


Fig. 2. Velocity-X (Left), velocity-Y direction (Right) at $t = 0.50 \text{ sec}$.

Figure 1 depicts the symmetric mesh in the fluid domain on the left side of the image; on the right side, there is the inflow of 20 (m/s) over the airfoil, with min -3.5 (m/s) and max 26 (m/s) of air moving over the airfoil at time step $t = 0.1 \text{ sec}$. Most of the domain on the right side maintains a constant flow around 20 (m/s) as inflow velocity, but experiences interaction of the airfoil shape. The simulation continued

at $t = 0.5 \text{ sec}$ captures the velocity-X and velocity-Y. The flow progresses as the time t increases.

As demonstrated in Fig. 2, velocity-X has the maximum velocity of 26 (m/s) captured during the simulation; on the other side, there is velocity-Y that has maximum velocity of 8.8 (m/s). As the inflow direction is in the X-direction, both velocities are in the expected range.

3. DATA PREPARATION

The dataset comprises a spatiotemporal field variable such as pressure; velocity-u and velocity-v are organised as 300 sequential snapshots (time steps) in $m_columns$ and $n_spatial$ points in rows. Each snapshot contains a two-dimensional matrix of dimensions (86909, 3); aggregating 100 snapshots gives third-order tensor $\chi \in \mathbb{R}^{300 \times 86909 \times 3}$ representing 300-time steps, each with 86909 spatial points and 3 field variables. The objective is to preprocess the data into a neural network compatible format while preserving the physical features of each variable and avoiding value-based generalization. This ensures that the machine learning model does not prioritise field variables with higher numerical magnitudes [5].

A two-step approach is adopted for

organising and preparing the datasets. First, each snapshot corresponding to a given time step is processed separately. Second, the processed data are flattened and combined into a single dataset. Flattening transforms the two-dimensional matrix or single snapshot (m, n) into a one-dimensional vector, stacking all rows one after the other, or row concatenation turns the (86909×3) matrix into a vector of shape (1,260727). The final dataset preserves the sequential spatial order and its associated field variables ($p0, u0, v0, p1, u1, v1$) into a one-dimensional vector. The multiple snapshots are combined into two-dimensional matrices of shape (300, 260727), resulting in a large-scale dataset that can be directly utilised for the machine learning model training [6].

$$2DMatrix(86909 \times 3) \rightarrow Flattening \rightarrow 1DMatrix(1,260727). \quad (1)$$

4. NORMALIZATION

Normalization is an essential pre-processing step before feeding because flow field variables, such as pressure, velocity-u, and velocity-v, have different units and magnitudes. Without normalization, variables with larger numerical ranges may dominate the learning process, potentially leading to biased training and numerical instability [7]. Normalizing ensures that no single variable dominates due to its inherent scale, and standard scaling preserves relative fluctuations while removing bias from the absolute magnitude. A standard scaler is a normalization technique that transforms the dataset to have a mean (μ) of 0 and a standard deviation (σ) of 1 to ensure numer-

ical stability and balanced feature contribution.

$$X_{scaled} = \frac{x - \mu}{\sigma}. \quad (2)$$

However, some features may contain zero variance, meaning all the values are identical, dividing by zero would result in Nann or infinities. To prevent this, we introduce zero standard deviation with a very small number, $\epsilon = 1e - 8$ to ensure the division is numerically safe while keeping normalized values effectively centered around zero.

$$X_{norm} = \frac{x - \mu}{\sigma_{safe}}, \quad (3)$$

where X is the input data, $\sigma_{safe} = \max(\sigma, \epsilon)$ is the standard deviation, replaced with ϵ if it is a zero value.

Pressure Distribution Analysis

It is necessary to convert the flow field data loaded 500 snapshots with 86909 points and 3 variables to normalize. Before normalization, the Pressure (Pa) variable exhibited a mean (μ) value of

$$\mu_{before} = 101122.72[Pa].$$

With a standard deviation of

$$\sigma_{before} = 4463.82[Pa].$$

This indicates that most of the pressure [Pa] values were centred around 101122.72 [Pa] with typically fluctuations of about $\pm 4463.82[Pa]$. After applying Z-score normalization (standardisation), the transformed data has a mean value of

$$\mu_{After} = 0.1547,$$

and the standard deviation of

$$\sigma_{After} = 0.9868.$$

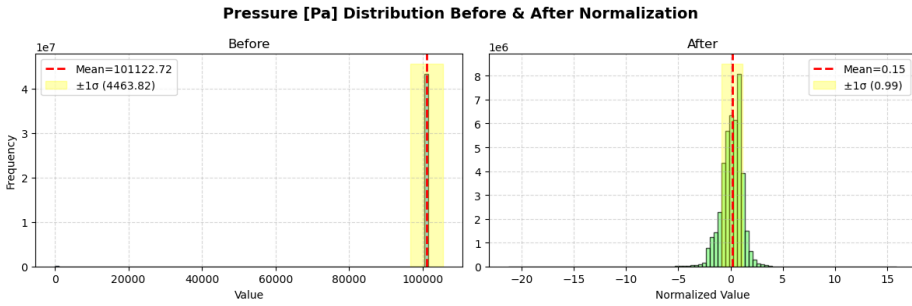


Fig. 3. Normalization of pressure distribution (before and after).

Velocity (x) [m/s] Distribution

The velocity (x) [m/s] centred around 19.8996 [m/s], with typically fluctuations of about ± 1.26 [m/s].

$$\mu_{before} = 19.90[m/s].$$

With a standard deviation

$$\sigma_{before} = 1.26[m/s].$$

After applying Z-score normalization (standardisation), the transformed data has a mean value of

$$\mu_{After} = 0.1545,$$

and a standard deviation

$$\sigma_{After} = 0.9868.$$

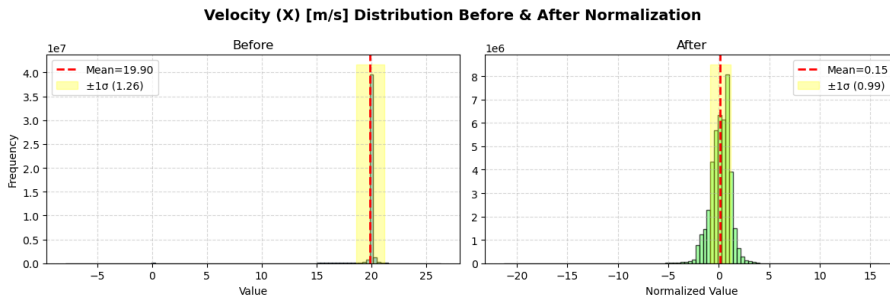


Fig. 4. Velocity (X)[m/s] distribution before and after normalization.

Velocity (y) [m/s] Distribution

The velocity (x) [m/s] centred around 0.21 [m/s], with typical fluctuations of about ± 0.20 [m/s]:

$$\mu_{before} = -0.2138[m/s].$$

With a standard deviation

$$\sigma_{before} = 0.1982[m/s].$$

After applying Z-score normalization (standardisation), the transformed data has a mean value of

$$\mu_{After} = 0.1545,$$

and a standard deviation

$$\sigma_{After} = 0.9868.$$

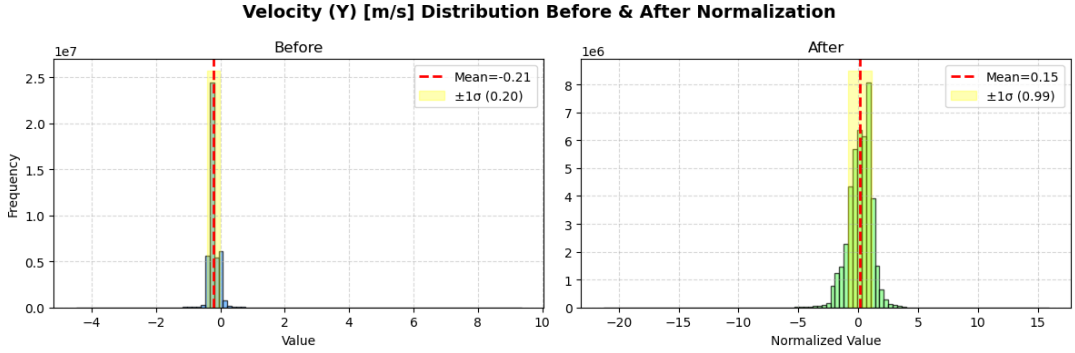


Fig. 5. Velocity-Y distribution before and after normalization.

5. VARIATIONAL AUTOENCODERS (VAES)

The governing equations of the fluid flow, such as the Navier-Stokes equations, are mathematically deterministic; fluid flow in engineering applications and natural systems exhibits inherently non-deterministic behaviour [8]. The fluid flow solutions are extremely sensitive towards the initial, boundary conditions, material properties, and operating conditions, which lead to unavoidable disturbances or uncertainties, resulting in variability in the fluid flow patterns. The inherent instability of fluid flow during transitional conditions and its strong sensitivity to minute disturbances lead to rapid changes in the velocity and pressure distributions [9]. Such sensitivity to initial and boundary conditions defines the characteristics of fluid flow. Despite the underlying deterministic laws, it is practi-

cally impossible to measure the exact flow states at every point in a fluid volume with zero error; it is impossible to impose initial and boundary conditions such as inlet velocity profiles and temperature gradients over the real system with mathematical perfection. The flow description at a single moment or snapshot cannot provide overall information of the fluid flow, but rather a probability distribution. This mathematical way describes all the possible flow states. Instead of a single snapshot, it provides a complete range of flow states. Probability distribution provides a framework to understand the uncertainty and randomness of fluid flow states. It defines all the possibilities, such as pressure (p), velocity (u), and time (t) assigning a probability density to each state and answering how likely it is to

observe velocity (u) evolution over time (t), given uncertainty that quantifies which flow states are common, rare, and impossible. To describe the variability of the fluid flow, probability theory provides a framework that offers a tool not as a single deterministic solution but as a distribution over all the possible flow states. Autoencoders (AE) are a separate class of neural networks specialised in unsupervised learning models capable of learning complex data that are rich in

non-linear data efficiently and compressing input data into a compressed latent space. This process is called dimensional reduction. Regular architecture of AE encodes input data and decodes or reconstructs to the original space but is unable to learn meaningful outputs. The sequential compression of the input data via neural architecture progressively reduces its dimensionality while preserving essential features.

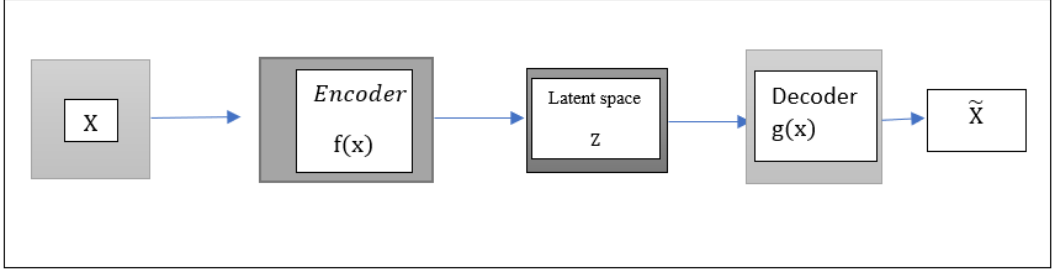


Fig. 6. Autoencoder construction.

This process culminates in the latent space, a compact information underlying patterns and structures of the captured data. The latent space construction by compressing high-dimensional features into low-dimensional features enables high efficiency in data storage. The complexity occurs when performing any operation on the latent space; the values are non-physical and require interpretation to verify the meaningful variations in the original space. In fluid flow applications, AEs are effective for dimensionality reduction; however, capturing the underlying process through learning remains intractable.

Unlike AEs, a Variational Autoencoder (VAE) captures meaningful data or flow patterns by learning the underlying probability distribution, which facilitates the generation of a new, high-fidelity sample from the original dataset. The uncertainty events of the flow data are considered by modelling them as variations within the learned latent

space via probability distribution properties of the input data. Probabilistic assumption of each snapshot to a random variable in the dataset, $X \in \mathbb{R}^D$, where X is drawn from an unknown true distribution $p_{data}(X)$ that the VAE seeks to approximate through a parametrized model $p_{\theta}(X)$ to capture both the mean behaviour and variability of the flow field. The probability theory provides a consistent framework for the quantification of uncertainty of the fluid flow scenarios; it allows making the prediction models for the given flow field dataset. The field flow data consist of variables such as geometric coordinates (m), Pressure (Pa), Velocity (X) [m/s] and Velocity (Y) [m/s] at a point.

$$P(p, u, v). \quad (4)$$

This can be implemented by concepts of joint probability, which describes the probability of two or more events (Pressure, velocities (x, y) directions) happening at the same time. Joint probability can

be extended by combining multiple event combinations of interest to understand the relationship between variables. Then, the marginal probability of pressure, $P(p)$ is found by integrating the joint probability over all values of u, v marginalizing the other variables.

$$P(p) = \int P(p, u, v) du dv. \quad (5)$$

The two fundamental rules of probability theory for the continuous variables, the sum rule, also called the marginalization rule, tells how the marginal probability density function (PDF) of one variable is obtained by integrating the joint probability over the other variables. Product rule is also called chain rule of probabilities; it

expresses the joint probability as a product of conditional probability and a marginal probability. Conditional probability describes the likelihood of an event occurring given that another event has already occurred.

$$P(p, u, v) = P(p|u, v).P(u, v). \quad (6)$$

This expresses the joint (PDF) of pressure, and velocities u and v . $P(p|u, v)$ is a conditional PDF of pressure given velocities and the product of the joint PDF of u and v .

The further decomposition of $P(u, v) = P(u|v).P(v)$ and by combining both $P(p|u, v).P(u, v)$.

$$P(p, u, v) = P(p|u, v).P(u, v) = P(p|u, v).P(u|v).P(v), \quad (7)$$

where

$P(v)$ – how velocity v is distributed;

$P(u|v)$ models how u depends on v ;

$P(p|u, v)$ models how pressure depends on both velocity components.

By extending the usage of Bayes' Theorem, we can rewrite or invert the conditional probability to invert the dependencies, which plays a central role in machine learning and flow pattern recognition. The posterior probability of pressure given observed velocities can be expressed as follows:

$$P(p|u, v) = \frac{P(u, v|p).P(p)}{P(u, v)}. \quad (8)$$

Prior probability represents the initial belief about a variable before the observation is made, $P(p)$ probability distribution

of pressure. Likelihood $P(u, v|p)$ describes how likely the observed velocities are given the specific pressure (p). Posterior probability $P(p|u, v)$, the updated variables after observing the velocities. $P(u, v)$ is the overall probability of observed data or fluid flow data; it also acts as a normalization factor adjusting the scale of posterior probability $P(p|u, v)$ that must equal 1, so that the total probability is valid. The interference of pressure fields is based on velocity observations and is fundamental to generative modelling and flow pattern analysis.

6. THE PRINCIPLE OF VAE

In a generative model, the core idea is to approximate the true distributions of the input dataset, here the fluid flow dataset,

by using Neural Networks (NNs). Assuming the input dataset would inherently follow a distribution like the normal distribu-

tion, instead of a single point VAE covers the area under the curve centred around the mean value corresponding to the standard deviation.

$$X \sim P_\theta(X), \quad (9)$$

where the X – input dataset, $P_\theta(X)$ – the distribution of input dataset and their attributes. To approximate the true distribution, a certain level of inference or trained models are required to make predictions from the given dataset. VAEs are interested in joint distributions of input dataset via encoders to the latent space $P_\theta(x, z)$, or between the input dataset to the latent variables Z , where θ represents the learned parameters during the training of the NN. This can be evaluated by the marginal distribution property:

$$P_\theta(x) = \int P_\theta(x, z) dz. \quad (9)$$

By considering the possible attributes of the input dataset, the learned parameters of the NN can describe the distribution of the original dataset. However, the above equation of the marginal distribution of the dataset is intractable due to infeasible integration over the latent space.

$$P_\theta(x) = \int P_\theta(x|z)P(z)dz \quad (10)$$

is computationally impossible, as $P(z)$ is a prior distribution over z , $P_\theta(x|z)$, is a likelihood and marginal likelihood $P_\theta(x)$, typically modeled as a Gaussian distribution of over observation x and z .

Approximating the likelihood, without a suitable loss function or constraint the networks may bypass the latent space z , resulting in random approximation of likelihood, which is not an actual solution. While the joint distribution satisfies symmetric property $P(x, z) = P(z, x)$, the conditional distributions $P(x|z)$ and $P(z|x)$ are not symmetric and cannot interchange. Optimisation of the marginal likelihood distributions can be resolved through the Bayes' Theorem.

$$P_\theta(x) = \frac{P_\theta(x|z)P(z)}{P_\theta(z|x)}, \quad P_\theta(z|x) = \frac{P_\theta(x|z)P(z)}{P_\theta(x)}. \quad (11)$$

However, the true posterior distribution $P_\theta(z|x)$ is intractable, the objective of VAE to approximate $\sim P_\theta(z|x)$ with a tractable distribution is computationally feasible. A variational inference model (an encoder) from the VAE makes $P_\theta(z|x)$ tractable, which learns to approximate posterior distribution $Q_\phi(z|x)$.

$$Q_\phi(z|x) \approx P_\theta(z|x), \quad (12)$$

where

ϕ – the learned parameters of this approximate distribution;

$Q_\phi(z|x)$ – variational posterior.

The hidden layers of the encoders in VAE are trained to compute the mean $\mu(x)$ and standard deviation $\sigma(x)$ of the posterior distribution $Q_\phi(z|x)$. The latent space (z) in AEs are known as a bottleneck since the latent dimension of (z) is smaller than input dataset (x), decoder models of the reconstructed output $\tilde{x}$ which is like the input dataset of (x). AEs are not good enough to produce or generate a new structure due to the latent space structure, since latent space was not continuous and data points in latent space were grouped easily for classification and clustering of the corresponding dataset. Operations such as regression or interpolation are not performed due to the discrete nature of the dataset in the latent space. VAE encoders for each input data points (x) are trained to compute the mean $\mu(x)$ and standard distribution $\sigma(x)$ that is modelled as a multivariate Gaussian distribution with a diagonal covariance matrix. It implies that the model prepares the latent variables that are not mutually correlated, which simplifies the covariance matrix structure as a closed form and remains the latent vector (z) tractable. The sampling layers from a multi-variant Gaussian $G(z_\mu, z_\sigma)$ with the mean and variances, respectively, ensure

that the latent space remains a continuous and smooth interpolation.

$$Q_\varphi(z|x) = N(z; \mu(x), \text{diag}(\sigma(x)^2)). \quad (13)$$

However, the latent space (z) does not contain a single vector, instead a Gaussian distribution. Updating the model parameters (θ) depends on $\mu(x)$ and $\sigma(x)$ to minimise the loss function or reconstruction losses. However, sampling or picking points from this distribution is a non-

differentiable operation; the model draws a random number from the Gaussian latent space. It cannot use a back-propagation technique to update the model parameters through the sampling steps. To overcome this challenge, VAEs apply a reparameterization trick, which introduces a random noise $\epsilon \sim N(0,1)$ sampled from a standard normal distribution (mean 0, std 1).

$$Z = \mu(x) + \sigma(x) \cdot \epsilon. \quad (14)$$

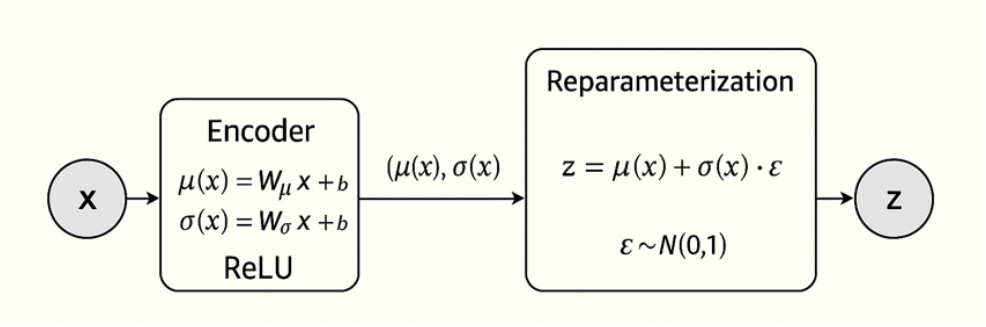


Fig. 6. Reparameterization.

The inference model, $Q_\varphi(z|x)$ is an estimation to $P_\theta(z|x)$ and it is necessary to identify its relationship. Thus, it is possible to approximate the $P_\theta(x)$ true distribution of the input dataset. The Kullback-Leibler (KL) divergence determines the distance

between two conditional densities. KL divergence is an important metric to evaluate how well $Q_\varphi(z|x)$ approximates with intractable $P_\theta(z|x)$, by reducing the KL divergence losses.

$$D_{KL}(Q_\varphi(z|x)||P_\theta(z|x)) = \mathbb{E}_{z \sim Q}[\log Q_\varphi(z|x) - \log P_\theta(z|x)]. \quad (15)$$

Using the Bayes' Theorem,

$$Q_\varphi(z|x) = \frac{P_\theta(x|z)P(z)}{P_\theta(x)}. \quad (16)$$

KL divergence and variational lower bound:

$$D_{KL}(Q_\varphi(z|x)||P_\theta(z|x)) = \mathbb{E}_{z \sim Q}[\log Q_\varphi(z|x) - \log P_\theta(z|x)]. \quad (17)$$

Bayes' expansion:

$$D_{KL}(Q_\varphi(z|x)||P_\theta(z|x)) = \mathbb{E}_{z \sim Q}[\log Q_\varphi(z|x) - \log P_\theta(x|z) - \log P_\theta(z) + \log P_\theta(x)]. \quad (18)$$

Since $\log P_\theta(x)$ does not depend on the latent space z , it is possible to reformulate it

for simplifying the KL divergence equation rearranging for the marginal log-likelihood:

$$\log P_\theta(x) = \mathbb{E}_{z \sim Q} [\log P_\theta(x|z)] - D_{KL}(Q_\phi(z|x) || P_\theta(z)) + D_{KL}(Q_\phi(z|x) || P_\theta(z|x)). \quad (19)$$

$$\log P_\theta(x) - D_{KL}(Q_\phi(z|x) || P_\theta(z|x)) = \mathbb{E}_{z \sim Q} [\log P_\theta(x|z)] - D_{KL}(Q_\phi(z|x) || P_\theta(z)). \quad (20)$$

The term $P_\theta(x)$ is also known as the evidence. The equation above is a decomposition of the log-marginal likelihood such that $\log P_\theta(x) = \mathbb{E}_{z \sim Q} [\log P_\theta(x|z)]$ represents

the Evidence Lower Bound (ELBO) and KL divergence term between the approximate posterior to true posterior.

$$ELBO(x) = \mathbb{E}_{z \sim Q} [\log P_\theta(z|x)] - D_{KL}(Q_\phi(z|x) || P_\theta(z)), \quad (21)$$

where

$\mathbb{E}_{z \sim Q} [\log P_\theta(z|x)]$ is a reconstructed error;

$D_{KL}(Q_\phi(z|x) || P_\theta(z))$ acts as a regularizer and ensures the latent space remains close to prior $P(z)$.

In VAEs, we have two loss functions, the $\mathbb{E}_{z \sim Q} [\log P_\theta(z|x)]$ represents the reconstruction likelihood, where z is the latent space. Reconstruction loss (L_R) is implemented to accurately reconstruct the input data (x) from its latent space. The second term $D_{KL}(Q_\phi(z|x) || P_\theta(z))$ quantifies

the difference between the approximate posterior $Q_\phi(z|x)$ and the prior distribution $P_\theta(z)$; Q_ϕ is a Gaussian distribution; $P_\theta(z) = P(z) = N(0, I)$ is the standard deviation. It acts like a regularizer, enforcing smoothness and continuity in the latent space $P_\theta(z)$.

$$-D_{KL}(Q_\phi(z|x) || P_\theta(z)) = \frac{1}{2} \sum_{j=1}^j \left(1 + \log(\sigma_j)^2 - (\mu_j)^2 - (\sigma_j)^2 \right), \quad (22)$$

where j is the actual dimension of the latent space, σ_j and μ_j are functions that enable the meaningful sampling or output from the latent space. The total losses are compressed into a combination of reconstruction loss + KL loss.

$$L_{VAE} = L_R + L_{KL}. \quad (23)$$

7. LONG SHORT-TERM MEMORY (LSTM)

LSTM is the extension of the Recurrent Neural Networks (RNNs) capable of handling time series data for the forecasting.

Since the fluid flow data are time series, the objective of the LSTM is to forecast next time steps.

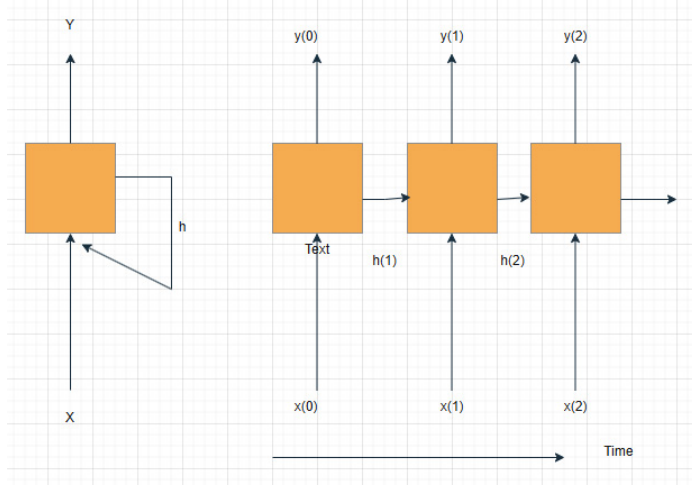


Fig. 7. Recurrent neural network.

Figure 6 shows the basic structure of RNNs, where $x(0)$ is input data and $y(0)$ – the output and $h(1)$ is the hidden layers feed-forwarded to another layer. The architecture of neurons of RNNs are different from the regular neuron in VAEs. However, feed-forward mechanisms in RNNs receive inputs and produce the output again sending it back to itself. At each time step t , neuron receives input x_t as well as its own output from the last time step $y_{(t-1)}$.

$$y_{(t)} = \phi(W_x X_{(t)} + W_y Y_{(t-1)} + b). \quad (24)$$

$$y_{(t)} = \phi([X_{(t)}, Y_{(t-1)}]W + b), \quad W = \begin{bmatrix} W_x \\ W_y \end{bmatrix}. \quad (25)$$

According to Eq. (24), X_t is an input matrix ($m \times n_{inputs}$) that contains all the features of the input data. ($Y_{(t-1)}$) is output of the previous time step; ($m \times n_{neurons}$) matrix contains the output layers. W_x are the connection weights from the inputs of the current time step. W_y contains the connecting weights from previous time steps.

b is the bias term where each neuron learns independently from the input values. Since there is no output at the initial neuron, we can consider it to be 0. We can generate many layers of recurrent neurons at each time step (t); each neuron receives both input and output from the previous layer or time step ($t - 1$). According to Eq. (25), each neuron has two weights (W_x^T, W_y) and single bias (b); these weights are concatenated vertically into single weight W . The dataset from the current input and previous output ($X_{(t)}, Y_{(t-1)}$) is concatenated horizontally into a single dataset matrix.

The output of each neuron at a time step not only depends on current input features but also on the previous step due to circulating its own output to maintain a form of memory. LSTM networks enhance the capability of learning long patterns from a large dataset and short patterns based on the problem. LSTMs are well suited for modeling time series.

8. VAE – LSTM FOR TIME-SERIES FORECASTING

The fluid flow over the airfoil was simulated, capturing the temporal evolu-

tion of the flow field. At each time step $t = 0.1$ sec, the simulation recorded pres-

sure (Pa), velocity-x (m/s), and velocity-y (m/s). A total of 300 snapshots were collected, resulting in a dataset of total matrix of (300, 260727); each row is snapshot. Due to the high dimensionality of the dataset, a direct analysis and forecast modelling become very computationally expensive. This study proposes a hybrid architecture that integrates VAEs with LSTMs for time series forecasting. Taking the advantage of VAEs, encoders effectively capture meaningful data or flow patterns by learning the underlying probability distribution and mapping onto lower dimensional space or latent space (z) capturing both structure and uncertainty. Decoder reconstructs back to the original dimensional space. The LSTM is then employed to model the temporal dynamics within this latent space (z); this helps reduce computation in high-dimensional space (300,260727) by operating instead in a low-dimensional latent space (300,128), facilitating LSTMs focus only on learning short-term memory. This architecture enables the efficient and accurate forecasting of future flow states while significantly reducing computational costs. To ensure the latent representations are meaningful and retain all the patterns, it is advisory to conduct full VAE loss as described in Eq. (23). This includes the reconstruction loss and KL divergence, ensuring the latent variables conform to a well-defined probabilistic structure. The completely trained end-to-end allows both compression and well-captured structures in the latent space. The encoder approximates the posterior distribution as expressed in Eq. (13). For each input $x_t \in \mathbb{R}^D$ at time step t , the encoder outputs mean and variation:

$$\mu_t = f_\mu(x_t), \log \sigma_t^2 = f_\sigma(x_t). \quad (26)$$

By reparameterization:

$$z_t = \mu_t + \mu_t \odot \epsilon_t, \epsilon_t \sim N(0,1). \quad (27)$$

The latent variable $z_t \in \mathbb{R}^D$ is obtained as the reparameterization allows the model to back propagate and updates the encoder's weights and biases as expressen in Eq. (13). The training objective is governed by the Evidence Lower Bound (ELBO), formulated in Eq. (21), which defines the total loss as a combination of KL divergence provided in Eq. (22). This loss function ensures that the encoder learns a compact and meaningful latent space, while the decode id is trained to reconstruct the original dimensional output. This is an important aspect, as the trained VAE latent space not only reduces the dimensionality but also captures the well-defined and meaningful patterns. A hybrid architecture comes when the LSTMs take inputs from VAE latent space $z(t)$ for forecasting.

$$Z = z_1, z_2, z_3, \dots \dots \dots z_t, \quad z_t \in \mathbb{R}^L. \quad (28)$$

At each time step $t=(0.1,0.2,\dots\dots\dots 1)sec$, and the length of the latent sequence $Z = z_1, z_2, z_3, \dots \dots \dots z_t$ should be the same, $t = z_t$ for typical sequential modeling setup. The crucial step follows when the previous hidden state is ($h_{t-1} \in \mathbb{R}^H$) , so each hidden state h_t is a point in the latent space that depends on the previous state h_{t-1} . In this case, there is no previous state, and the latent space starts at z_1 but z_0 is the initial anchor, usually zero or learned. Then, the initial hidden layer $h_0 = 0$. Then, at $t = 1$ the hidden state becomes:

$$\begin{aligned} h_{1(t=1)} &= f(h_0, x_1), \\ h_{2(t=2)} &= f(h_1, x_2). \end{aligned} \quad (29)$$

It is important to understand the hidden and cell state evolutions, the cell state functions as a long-term memory, encoding and preserving patterns learned across the entire sequence during training. While the hidden state captures the recent patterns called short-term memory that are used during the forecasting process to make predictions.

The cell state and hidden state are interdependent, the hidden state itself is influenced by the cell state, so the accuracy of short-

term predictions depends on the long-term memory.

9. VAE ENCODER-DECODER

The VAE encoder architecture is presented below.

```
# Configuration
CONFIG = {
    # VAE Configuration
    "latent_dim": 128, # latent space dimensions
    "hidden_layers": [1024, 512] # Two hidden layers
    "dropout_rate": 0.3,
    "l2_reg": 1e-5, # L2 regularization
    "learning_rate": 3e-5,
    "batch_size": 32,
    "epochs": 150,}

inputs = layers.Input(shape=(input_dim,))
x = layers.Dense(
    self.config["hidden_layers "],
    activation='relu',
    kernel_regularizer=regularizers.l2(self.config["l2_reg"]),
    kernel_initializer='he_normal'
)(inputs)
x = layers.Dropout(self.config["dropout_rate"])(x)
x = layers.Dense(
    self.config["hidden_layers "] // 2,
    activation='relu',
    kernel_initializer='he_normal'
)(x)

z_mean = layers.Dense(self.config["latent_dim"], name='z_mean')(x)

# Constrained log variance
z_log_var = layers.Dense(
    self.config["latent_dim"],
    name='z_log_var',
    kernel_initializer='zeros',
    activation='tanh'
)(x)
z_log_var = layers.Lambda(lambda x: 2 * x)(z_log_var) # Scale to [-2, 2]

# Sampling layer
z = layers.Lambda(
    self._sampling,
    name='z'
)([z_mean, z_log_var])

return models.Model(inputs, [z_mean, z_log_var, z], name="encoder")
```

10. RESULTS

The integrated VAE-LSTM framework demonstrated forecasting techniques over the time series dataset. The model successfully predicted the latent space vector for future time steps t_{t+1} , which were subsequently decoded to reconstruct the full-state flow variables. A critical analysis of the forecasting accuracy was conducted by comparing predicted values with the simulation dataset over the same time. The model demonstrated a high degree of accuracy in predicting this metrics. The actual simulation recorded maximum velocity of 26 m/s, while the forecast flow field from the LSTM decoder pipeline produced a value of 25 m/s. This has a remarkably small error of approximately 3.8 % for the multi-step forecast of highly non-linear system. Such an error is not a failure of

the model but is the expected and inherent consequence of the model architecture and training objective. There is an excellent reconstruction loss to the value reconstruction loss (1.3 and 2.1), indicating that the encoder-decoder is effective at compressing the high-dimensional flow field data into latent representation and subsequently reconstructing it with minimal error. The decoder successfully learned to generate flow fields from the latent variable, which is the foundational requirement for the forecasting pipeline. The slight deviation of the total_loss = recon_loss (1.5) + β *KL_loss (4.2) = 5.7; this total loss is dominated by the reconstruction error and the KL divergence, which the model balances to make the reconstructions while keeping a smooth and continuous latent space.

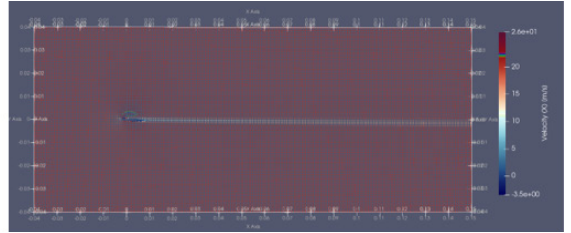
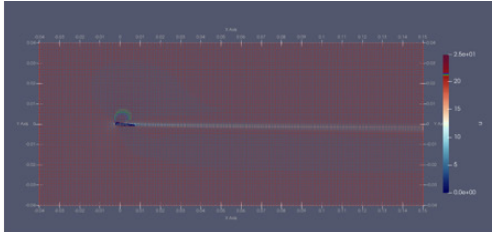


Fig. 8. LSTM forecast (left) and actual numerical simulation at time step 301.

The forecast (left) velocity-X displays a maximum value of 25 m/s, whereas actual simulation displays 26 m/s, which is only 3.8 % of deviation from the forecast to actual simulation at time step 301. Both the

predicted and actual fields exhibit identical flow patterns, with similar flow structures around the airfoil captured across their spatial points.

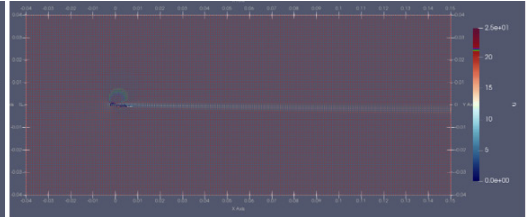
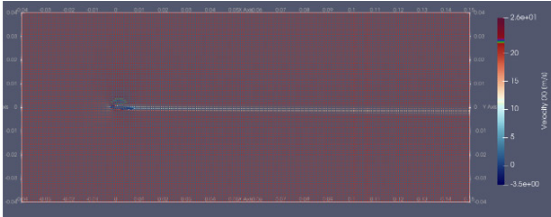


Fig. 9. Actual numerical simulation (left) and LSTM forecasting at time step 302.

The actual numerical simulation of velocity-X displays a maximum value of 26 m/s, whereas LSTM forecasting displays 25 m/s. It still shows only 3.8 % of deviation from the forecast to actual simulation at

time step 301. Both the predicted and actual fields exhibit identical flow patterns with similar flow structures around the airfoil captured across their spatial points.

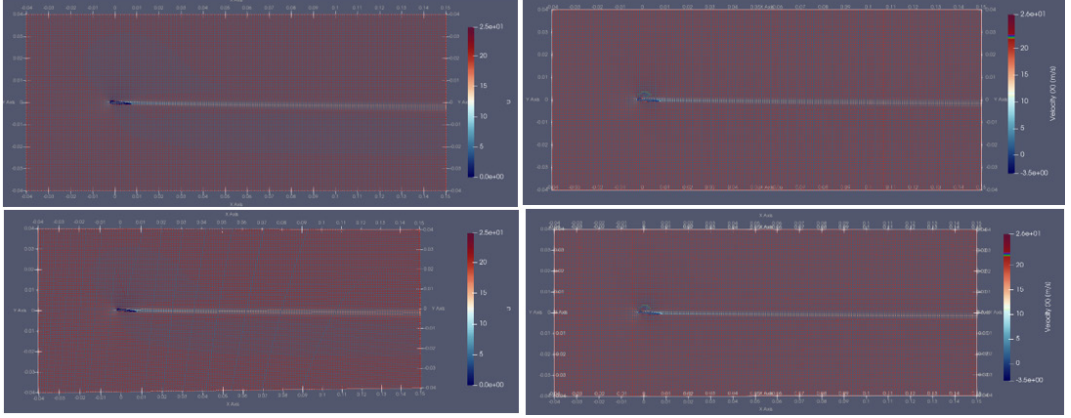


Fig. 10. LSTM forecasting of velocity-X (left) and actual numerical simulation at time steps 305 and 315.

According to Fig. 10, at time steps 305 and 315, the comparison shows a high degree of agreement, with LSTM forecast flow field exhibiting identical structures to the actual simulation. Both techniques reveal that the model captures the evolution of flow over the time step. In actual numerical simulations, after a certain time step, the flow patterns become constant and display

similar structures over subsequent time steps. The LSTM forecasting of velocity X also displays the constant flow patterns over time without using fluid flow governing equations. This successful extrapolation that is far beyond the input window confirms that the hybrid models effectively use short-term forecasting of the complex system.

11. CONCLUSION

This study successfully developed and demonstrated an integrated VAE and LSTM network framework for the efficient and accurate forecasting of high-dimensional, non-linear fluid flow fields. Faced with the significant challenge of modelling complex spatiotemporal data from numerical simulations, the hybrid approach effectively addressed the curse of dimensionality by learning a compressed, probabilistic latent representation of the flow. The VAE component proved highly effective in its primary

role of dimensionality reduction, distilling the essential spatial features from the vast original dataset (with dimensions of $300 \times 86,909 \times 3$) into a compact latent space. Despite observed overfitting in the latent space regularization, evidenced by a high Kullback-Leibler divergence on validation data, the decoder achieved exceptional performance, reliably reconstructing the flow fields with a low reconstruction loss. This confirmed the model's robust capability to preserve critical physical information

within the compressed representation. The LSTM network successfully leveraged this structured latent space to learn the temporal dynamics of the flow evolution. The forecasting results are highly promising; the model accurately predicted the evolution of key flow parameters for a 15-step-ahead forecast horizon. **3.8 % error** is a particularly strong indicator of the model's precision, especially given the highly non-

linear and sensitive nature of the fluid system governed by the Navier-Stokes equations. In conclusion, the VAE-LSTM framework presents a powerful, end-to-end pipeline for spatiotemporal forecasting in complex physical systems. It offers a formidable advantage over conventional methods by drastically reducing computational demands while maintaining high predictive accuracy.

REFERENCES

1. Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(35), 1798–1828. <https://doi.org/10.1109/TPAMI.2013.50>
2. Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems* (2nd ed.). O'Reilly Media.
3. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.
4. Holmes, P. L. (2012). *Turbulence, coherent structures, dynamical systems and Symmetry* (2nd ed.). Cambridge University Press.
5. Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations (ICLR 2014)*, Banff, AB, Canada. <https://doi.org/10.48550/arXiv.1312.6114>
6. Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In E. P. Xing & T. Jebara (Eds.), *Proceedings of the 31st International Conference on Machine Learning (ICML 2014)* (Proceedings of Machine Learning Research, Vol. 32, No. 2, pp. 1278–1286). PMLR. <https://doi.org/10.5555/3044805.3045035>
7. Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656, 5–28. <https://doi.org/10.1017/S0022112010001217>
8. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>