**Exploring the Pioneer Works: First Applications of Neural Networks in Urban Wind Field Computation.**

Armand’s comments concerning the PDF File:



Ameir SHAA, Claude GUET and Armand ALBERGEL

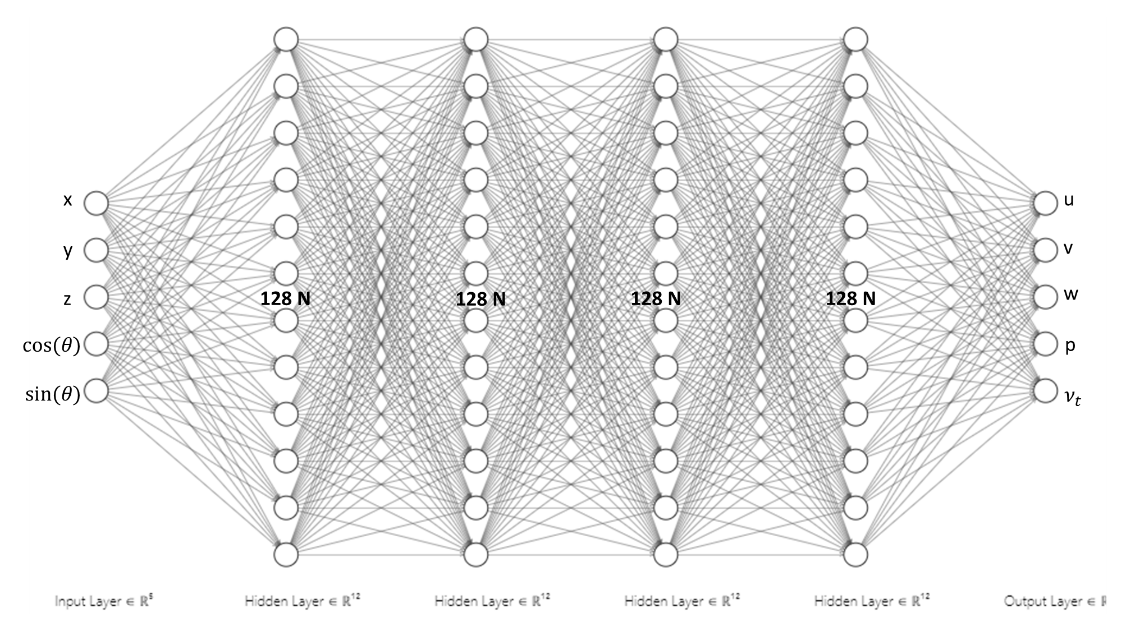
14/10/2023

# Preliminary comments

1. I would like to express my gratitude for sharing your pioneering work and for the effort you've put in such a short span of time.
2. I regret that we didn't have the opportunity for a more in-depth discussion during our last interaction. I would like to propose the idea of expanding and regularizing our exchanges on a weekly or bi-weekly basis. This would provide us with the necessary time and platform to delve deeper into the subject matter.
3. To facilitate our discussions, I suggest that we use a better structured approach. I propose creating a dated, numbered, and organized presentation in either PowerPoint (PPT) or PDF format. This will help us keep track of our progress and ensure that our discussions are as efficient and productive as possible.
4. Emphasizing the importance of careful attention to graphics is crucial for both comprehension and persuasion. Understanding the issue of pixelation, which can hinder the readability of presented documents, is essential. Clearly defining the problem is key, especially in the context of illustrating cross-sectional views in relation to the geometry of obstacles, among other aspects.

# Specific comments

|  |  |
| --- | --- |
|  | May we suggest a standard cover page featuring   * Logos : the Descartes, CNRS@CREATE, and ARIA Technologies - SUEZ logos (as seen in the header of this document ? ), * Date the presentation date, * and an order number, such as PM001 (Progress Meeting).   PINN stands for Physics-Informed Neural Network. It is a type of neural network architecture and methodology used in machine learning and scientific computing, particularly in solving partial differential equations (PDEs) and inverse problems in physics and engineering.  The key idea behind PINNs is to combine the power of neural networks for approximating complex functions with the physical laws and constraints governing a particular system. In other words, PINNs are designed to incorporate prior knowledge of the underlying physics into the neural network model.  Here's how a PINN typically works:   1. **Data Collection:** You collect data points from the system or experiment, which may include measurements or observations of the variables of interest. 2. **Neural Network Architecture**: You design a neural network, often a deep neural network, that takes the system's input parameters as input and produces an output. This output could be a prediction or approximation of the system's behavior. 3. **Loss Function:** In addition to the standard loss function used for training neural networks, PINNs incorporate a physics-informed loss term. This term enforces that the neural network's output adheres to the underlying physical equations, such as PDEs, boundary conditions, or other constraints. The physics-informed loss term ensures that the neural network model respects the laws of physics. 4. **Training:** The model is trained by minimizing the combined loss, which includes both the standard data-driven loss and the physics-informed loss. This optimization process allows the model to learn the underlying physics and make accurate predictions. 5. **Prediction and Analysis**: Once trained, the PINN can be used to make predictions, solve PDEs, or estimate physical quantities of interest for new input conditions. |
|  | * **Input**: Limiting the input variables to x, y, z, and should be enough in this phase of the study. Nu-t (is associated with turbulence production/destruction and, therefore, should be considered an output (in the RANS model, it is proportional to the square of the kinetic turbulent energy and inversely proportional to turbulent dissipation rate y). Rho and nu-molecular are constants in this context. While we need them to verify RANS, it doesn't seem necessary to complicate the input. * **Output**: May we consider u, v, w, p, ? The Reynolds stress tensor is not needed I think as we consider only the trace (k) and is sufficient for RAND PDE. should be directly be used in dispersion model. * Reynolds stress tensor (no learning data and not very useful) |



|  |  |  |
| --- | --- | --- |
| **Some Parameters** | | |
| * Infinite epochs - instead the criteria for stopping is loss\_{n} - loss\_{n-1} < ε for 10 consecutive epochs where n is the epoch number and ε = 1E-5 (user defined) | OK While the criterion of monitoring loss reduction over 10 consecutive epochs is a reasonable approach, it's important to tailor the stopping criterion to the specific requirements of your task and dataset. My understanding is that some datasets may require more or fewer epochs to reach a suitable solution. Adapting the stopping criterion accordingly can help account for this variability. |
| * We have the data for 7 angles, [0, 30, 60, 90, 120, 150, 180] in degrees. * We concatenate the data for angles = [0, 30, 60, 120, 150, 180] and then take 80% of the dataset with random seed = 42 for training and 20% for testing * By using the whole dataset, we hope to make the NN learn about wind angle such that the parameters become functions of the wind angle. * We also would like the stress tensor to be learnt so that we do not have to assume models involving turbulence (ie. model free approach) | * I understand that for the learning phase:   + the whole Code\_Saturne 3D output field is used to select the learning data set (80%) and the testing data set (20%) The selection of 20% is randomly done (with random seed = 42 ?). I am afraid that the 20% may miss or under-represent “interesting” part of the flow (around obstacles / wake …). * Please confirm:   + the normalization of the different physical data   + the shuffling done to ensure that the training data is presented in a random order during each epoch.   + which activation function selected (tanh? as Bowen?) , |
| * Then using the trained neural network, we predict the data for angle = 90 | * May be see the influence in selecting another angle? |

|  |  |
| --- | --- |
|  | Adam, short for "Adaptive Moment Estimation," is an extension of the stochastic gradient descent (SGD) optimization method and is known for its efficiency and effectiveness in a wide range of tasks.  Good scores: risk of overfitting? |

|  |  |
| --- | --- |
|  | * Thank you for understanding and for fixing the excessive pixelation in the image originates. It may be attributed to various factors, such as graphical artifacts, color scale, irregular grid, and more. It is important to see the flow. * Are x and y and P normalized? |

|  |  |
| --- | --- |
|  | * The main flow is parallel to x, v and w are most of the time identically 0 except in the wake. * The anti-correlation of the vertical component is concerning |

|  |  |
| --- | --- |
|  | * Interesting to follow the entire curve for both the training set and the testing set as |

|  |  |  |
| --- | --- | --- |
| Data loss | Data + Div | Data +div +RANS |
|  |  |  |

# Concluding remarks

**1. The training phase is relatively time-consuming, and it appears to require few thousand epochs minimum. However, this extended training may lead to overfitting, as indicated by the significant degradation in performance on the validation dataset.**

**2. The incorporation of physics (divergence and RANS) demonstrates a positive impact, enhancing almost all performance scores. This is an encouraging outcome in my opinion.**

**3. Vertical velocity (Velocity:2) emerges as particularly challenging to learn and generate.**

# Suggestions for the next steps

**1. Consider revisiting the graphical procedures of Wang Zhe and/or Bowen to create figures that offer an immediate visual representation of the flow patterns.**

**2. Verify the data sampling of both training and test datasets and ensure proper normalization of all variables to mitigate numerical artifacts.**

**3. To streamline the NN model, consider initially testing with outputs u, v, and w, and then progressively add variables such as nut, P, or others as needed?**

**4. Explore ways to reduce the risk of overfitting, for instance, by experimenting with a lighter neural network architecture, such as reducing the number of neurons per layer from 128 to 64 for testing purposes."**

**5. Evaluate whether additional wind directions are necessary for a more comprehensive understanding of the flow dynamics.**

**Feel free to adapt these suggestions according to your specific vision and research context.**