SHAP values seminar

Andrés Cremades Botella[*,1], Sergio Hoyas[2], Ricardo Vinuesa[*,1]

* andrescb@kth.se, rvinusa@kth.se

1 FLOW, Engineering Mechanics, KTH Royal Institute of Technology, Stockholm SE-100 44, Sweden.

2 Instituto Universitario de Matemática Pura y Aplicada, Universitat Politècnica de València, 46022 Valencia, Spain.

Funded by the European Union
European Research Council
Established by the European Commission

Introduction

Index

- **Introduction**
- Methodology
- Tutorial
- Conclusions

- Era of data
  - Machine learning is becoming part of our lives: Fluid mechanics research is not an exception
- Black box model behavior

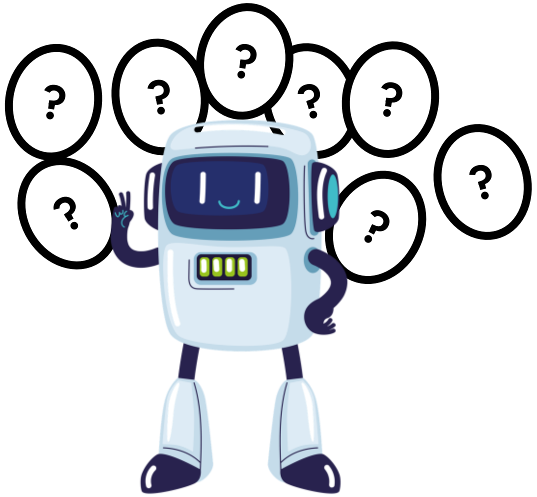


Image Designed by Freepik

- We need to understand the predictions and the models

Image Designed by Freepik

WE NEED SIMPLE MODELS!!!!

Review of SHAP for fluid dynamics and heat transfer

Funded by the European Union

European Research Council
Established by the European Commission

Introduction

Index

- What is the best way to understand a model?
  - The model itself:

$$F = m\,a$$

    - An example is second Newton law: the correlation between the force and the acceleration is clear and understandable
  - This cannot be done with difficult models:
    - Navier-Stokes equation
    - Deep learning models

Help me!

Image Designed by Freepik

- Linear models are easy to understand:
  - Additive-feature-attribution methods
    - SHapley Additive exPlanations (SHAP)

$$g \cong f \;\rightarrow\; g(z') = \phi_0 + \sum_{i=1}^{N} \phi_i z_i'$$

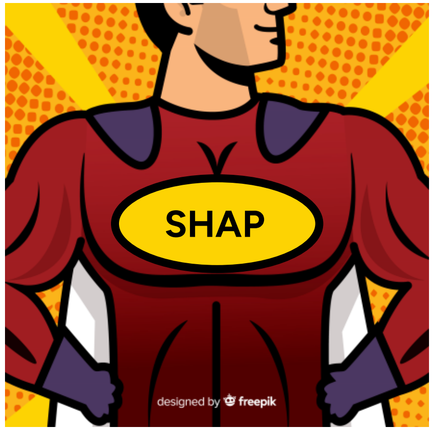


Image Designed by Freepik

Review of SHAP for fluid dynamics and heat transfer

Funded by the European Union
European Research Council
Established by the European Commission

Methodology

Index
- Introduction
- **Methodology**
- Tutorial
- Conclusions

- Additive-feature-attribution methods → Unique solution
  - 3 properties:
    - Local accuracy: $f(x) = g(x') = \phi_0 + \sum_{i=1}^{N} \phi_i x'_i \rightarrow$ if $\rightarrow x = h(x')$
    - Missingness: $\phi_i = 0$ if $x'_i = 0$
    - Consistency: $\phi'_i > \phi_i \rightarrow$ if $\rightarrow g'(x') - g'(x' \setminus i) > g(x') - g(x' \setminus i)$

- Only Shapley values can satisfy the previous conditions:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \left(\frac{|S|!\,(N - |S| - 1)}{N!} \right) \left(f(S \cup i) - f(S) \right)$$

  - Probability of a coallition to happen: $\left(\frac{|S|!(N-|S|-1)}{N!} \right)$
  - Marginal contribution of the feature in the coallition: $\left(f(S \cup i) - f(S) \right)$

Shapley, L. S. (2016). 17. A value for n-person games. In *Contributions to the Theory of Games, Volume II* (pp. 307-318). Princeton University Press.

Review of SHAP for fluid dynamics and heat transfer

Funded by the European Union

European Research Council
Established by the European Commission

Methodology

Index

- Introduction
- **Methodology**
- Tutorial
- Conclusions

- Computational cost increases exponentially with the number of features
  - Shapley values cannot be applied to fluid mechanics.

Valor de Shapley. (2023, October 2). In *Wikipedia*. https://es.wikipedia.org/wiki/Valor_de_Shapley

- We need to simplify the methodology:
  - Option 1: We do not know the model
    - Model agnostic
  - Option 2: Optimized for the architecture
    - Model specific

Review of SHAP for fluid dynamics and heat transfer

Funded by the European Union
European Research Council
Established by the European Commission

Methodology

Index
- Introduction
- **Methodology**
- Tutorial
- Conclusions

- Model Agnostic:
  - Kernel SHAP:
    - Shapley values + LIME (local interpretable model-agnostic explanations)
    - The SHAP values are calculating minimizing a los function $\mathcal{L}$:

$$\xi = argmin_{g \in \mathcal{G}} \left(\mathcal{L}(f, g, \pi_x) + \Omega(g) \right)$$

    - The definition of the loss function for matching the Shapley values:
      - Regularization term: $\Omega(g) = 0$
      - Loss function: $\mathcal{L} = \sum_{z' \in Z} \left(f\left(h(z')\right) - g(z') \right) \pi_x(z')$
      - Kernel: $\pi_x(z') = \frac{N-1}{\binom{N}{|z'|}|z'|(N-|z'|)}$

Lundberg, S. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Review of SHAP for fluid dynamics and heat transfer

Funded by the European Union
European Research Council
Established by the European Commission

Methodology

Index
- Introduction
- **Methodology**
- Tutorial
- Conclusions

- Model Specific:
  - Gradient SHAP:
    - Based on Expected gradients:
      - Expectation of the gradient over a set of trajectories from different references

$$\phi_i = E_{x_r \sim D, \alpha \sim \{0,1\}} \left[(x_i - x_r) \left(\frac{\partial f\left(x_r + \alpha(x - x_r)\right)}{\partial x_i} \right) \right]$$

  - Deep SHAP (Shapley values+DeepLIFT):
    - Backpropagates the gradient at a single step of the model.
    - Approximation of the gradient:

$$\phi_i \approx m_{x_i, f_j}(x_i - E[x_i])$$

Erion, G., Janizek, J. D., Sturmfels, P., Lundberg, S. M., & Lee, S. I. (2021). Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature machine intelligence*, *3*(7), 620-631.

Lundberg, S. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

Review of SHAP for fluid dynamics and heat transfer

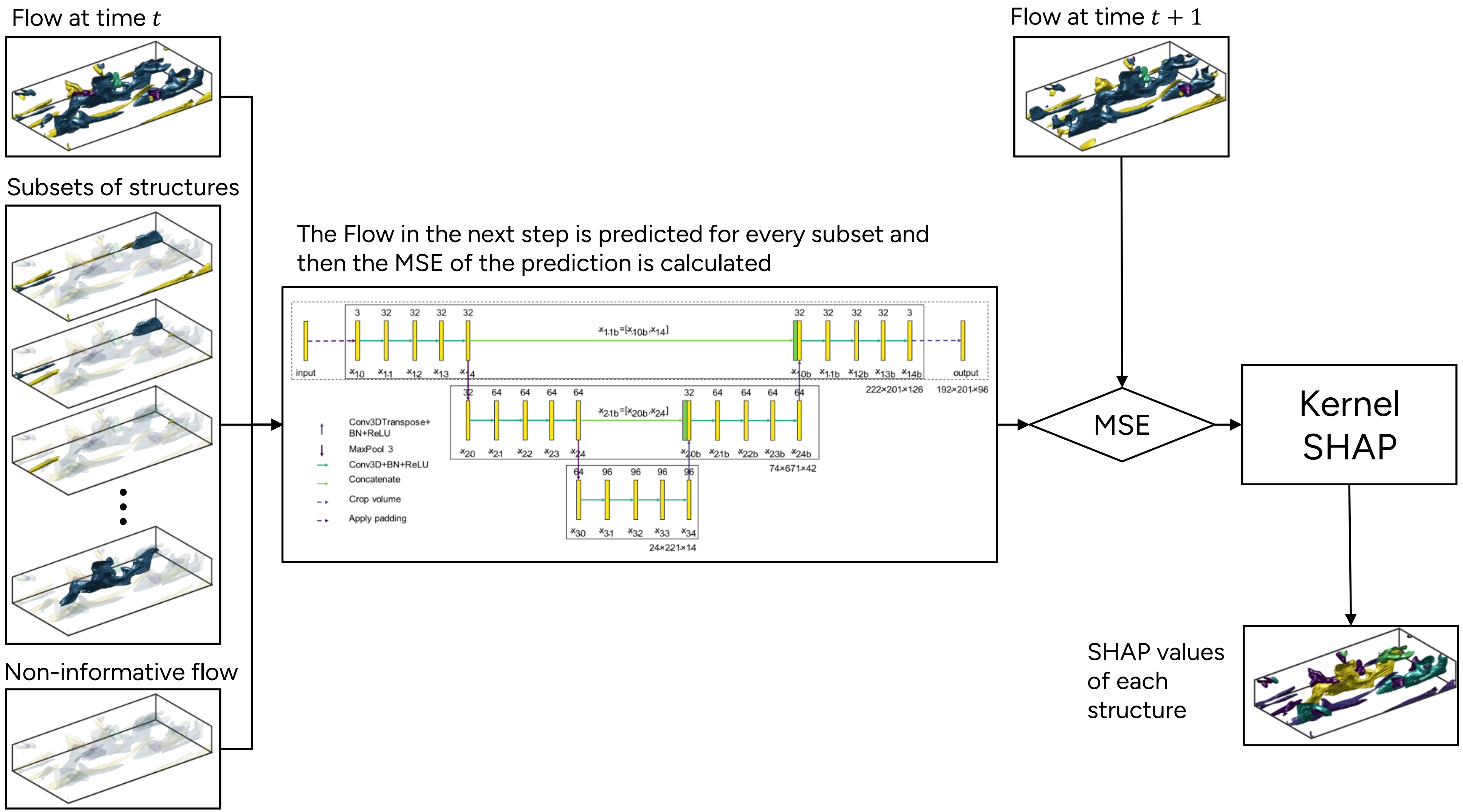
Funded by the European Union

European Research Council
Established by the European Commission

Methodology: Application to fluid mechanics

Index

- Introduction
- **Methodology**
- Tutorial
- Conclusions



Review of SHAP for fluid dynamics and heat transfer

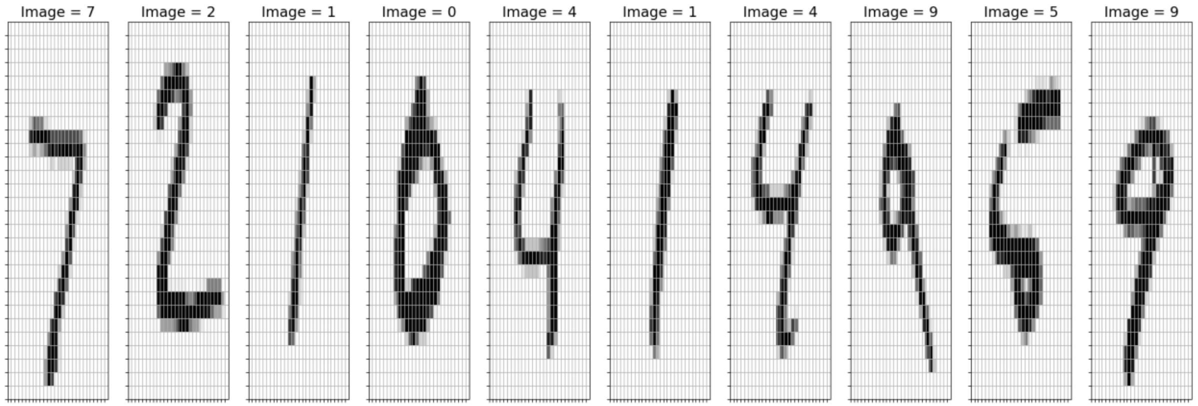
Funded by the European Union

European Research Council
Established by the European Commission

Tutorial

Index
- Introduction
- Methodology
- **Tutorial**
- Conclusions

- How do we calculate the SHAP values?
  - Example using MNIST database:



  - Tensorflow
  - SHAP

  - Modification of the tutorial:

TensorFlow

SHAP

Tutorial

Funded by the European Union

erc European Research Council Established by the European Commission

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions

```python
# -*- coding: utf-8 -*-
"""
Created on Fri Jan 17 11:23:24 2025

@author: andres cremades botella

File for creating a SHAP tutorial - Group meeting Mon Jan 20.

Base example taken from https://shap.readthedocs.io/en/latest/example_notebooks/image_examples/image_classification/Multi-input%20Gradient%20Explainer%20MNIST%20Example.html
"""
#%%
# -----------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Import the packages
# -----------------------------------------------------------------------------------------------------------------------------------------------------------------------
import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.layers import Conv2D, Dense, Dropout, Flatten
import numpy as np
import matplotlib.pyplot as plt
import matplotlib
import shap
```

- The first step is to import all the required packages

Tutorial

Funded by the European Union

European Research Council
Established by the European Commission

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions

Tutorial

```
#%%
# ----------------------------------------------------------------------------------------------------
# Define the size of the images
#     - size_x : size of the picture in x
#     - size_y : size of the picture in y
#     - outcha : number of output channels
#     - fs     : font size of the plots
# ----------------------------------------------------------------------------------------------------
size_x = 28
size_y = 28
outcha = 10
fs     = 14
matplotlib.rc('font',size=fs)

# ----------------------------------------------------------------------------------------------------
# load the MNIST data
#     - x_train : training data for the input
#     - y_train : training data for the output
#     - x_test  : validation data for the input
#     - y_test  : validation data for the output
# ----------------------------------------------------------------------------------------------------
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
x_train, x_test                      = x_train / 255.0, x_test / 255.0
x_train                              = x_train.astype("float32")
x_test                               = x_test.astype("float32")
x_train                              = x_train.reshape(x_train.shape[0], size_x, size_y, 1)
x_test                               = x_test.reshape(x_test.shape[0], size_x, size_y, 1)

# ----------------------------------------------------------------------------------------------------
# Plot the Data
# ----------------------------------------------------------------------------------------------------
fig, axes = plt.subplots(1, 10, figsize=(25, 4))
for ind_i, ax in enumerate(axes.flat):
    ax.set_title("Image = "+str(y_test[ind_i]))
    ax.pcolor(np.flip(x_test[ind_i,:,:,0],axis=(0)),cmap="Greys")
    ax.set_xticks(range(0,size_x,1))
    ax.set_yticks(range(0,size_y,1))
    ax.grid()
    ax.set_xticklabels([])
    ax.set_yticklabels([])
plt.savefig("data.png")
```

Set the dimensions of the problem

Import the data and divide it in training and validation data

Plot the data to check

- Then import the data

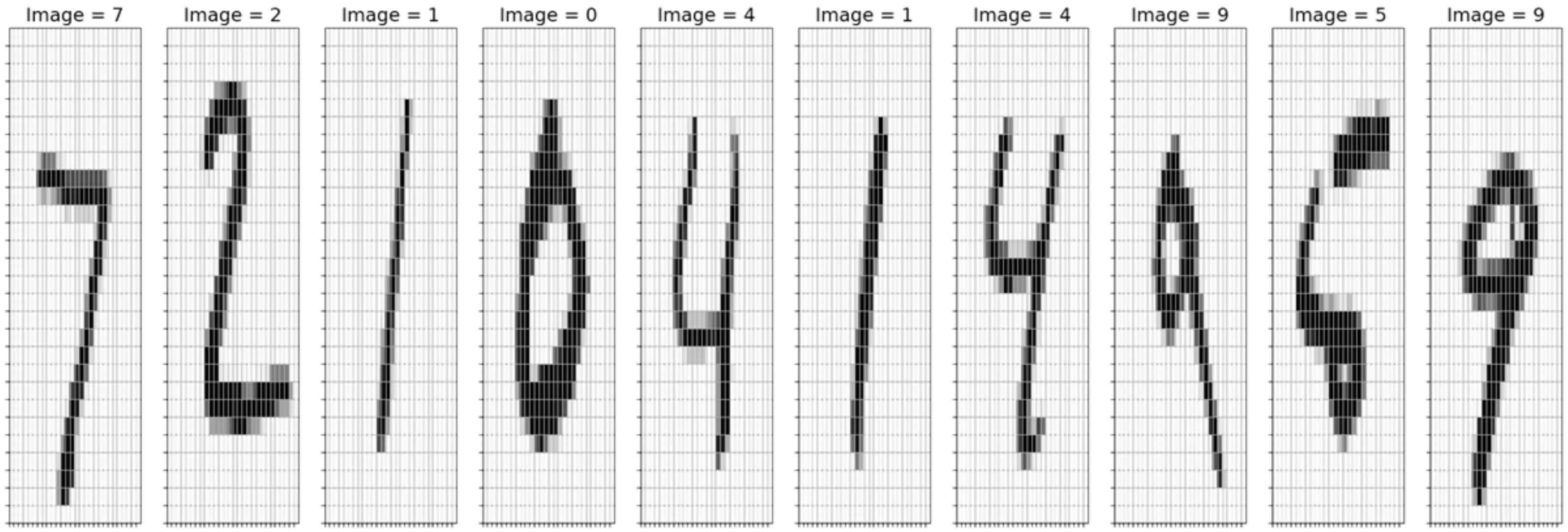
Funded by the European Union

European Research Council
Established by the European Commission

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions



Tutorial

- MNIST data is a collection of hand-writen numbers.

Funded by the European Union

European Research Council
Established by the European Commission

Tutorial

Index


```python
#%%
# ------------------------------------------------------------------------------------------------
# Define our model
# The model takes 2 inputs: input1 and input2 which are the input image. Then a convolutional network is used
# ------------------------------------------------------------------------------------------------
input1  = Input(shape=(size_x, size_y, 1))
input2  = Input(shape=(size_x, size_y, 1))
input2c = Conv2D(32, kernel_size=(3, 3), activation="relu")(input2)
joint   = tf.keras.layers.concatenate([Flatten()(input1), Flatten()(input2c)])
out     = Dense(outcha, activation="softmax")(Dropout(0.2)(Dense(128, activation="relu")(joint)))
model   = tf.keras.models.Model(inputs=[input1, input2], outputs=out)

model.compile(optimizer="adam", loss="sparse_categorical_crossentropy", metrics=["accuracy"])

# ------------------------------------------------------------------------------------------------
# Fit the model
# ------------------------------------------------------------------------------------------------
model.fit([x_train, x_train], y_train, epochs=3)
```

Definition of the model. Note that it duplicates the image and uses it as 2 inputs

Train the model

- Define and train the model.

Tutorial

Funded by the European Union
European Research Council
Established by the European Commission

Tutorial

Index
- Introduction
- Methodology
- **Tutorial**
- Conclusions

Tutorial

Funded by the European Union

European Research Council
Established by the European Commission

- Calculate the Kernel Explainer.
  - The Kernel Explainer is model agnostic.
  - It can be used to work with functions that are generic
    - Not defined in TensorFlow or PyTorch.
  - We will define a mask to cluster the pixel and explain the model by regions instead of individual features.

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions

Tutorial

Funded by the European Union
European Research Council
Established by the European Commission

```
#%%
# ----------------------------------------------------------------------------------------------
# Calculate the SHAP values using the KernelExplainer
# ----------------------------------------------------------------------------------------------

# ----------------------------------------------------------------------------------------------
# Segmentate the domain
# ----------------------------------------------------------------------------------------------
box_size = 7
index    = 0
mask     = np.zeros((size_x, size_y), dtype=int)
for ind_i in range(0, size_x, box_size):
    for ind_j in range(0, size_y, box_size):
        mask[ind_i:ind_i+box_size, ind_j:ind_j+box_size] = index
        index += 1

nmask = np.max(mask)
```

Create a mask to segment the domain in the groups that we are interested to explain

```
# ----------------------------------------------------------------------------------------------
# Create a reference for the SHAP values
# ----------------------------------------------------------------------------------------------
reference = np.zeros((size_x,size_y,1))
```

Create a reference to substitute the absent regions

```
# ----------------------------------------------------------------------------------------------
# Define the input to calculate the SHAP values
# ----------------------------------------------------------------------------------------------
Xin = x_test[0].reshape(1,size_x,size_y,1)
```

Definition of the image to explain

```
# ----------------------------------------------------------------------------------------------
# Function of the model
# ----------------------------------------------------------------------------------------------
def f(zs):
    ii  = 0
    lm  = zs.shape[0]
    out = np.zeros((lm,outcha))
    print("Starting kernel SHAP:",flush=True)

    for ii in np.arange(lm):
        if ii<lm-1:
            print("Calculation "+str(ii)+" of "+str(lm),end='\r',flush=True)
        else:
            print("Calculation "+str(ii)+" of "+str(lm),flush=True)

        zii         = zs[ii]
        model_input = mask_dom(zii)
        out[ii,:]   = model.predict([model_input,model_input])
    return out
```

Function used for the explanation. As Kernel SHAP is model agnostic it is a generic function. This function mask the input image and predicts the output.

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions

```python
# ------------------------------------------------------------------------------------------------------------
# Function to get the index of the blocks
# ------------------------------------------------------------------------------------------------------------
def get_structure_indices():
    struc_indx = []
    for ii in range(nmask):
        indx = np.array(np.where(mask == ii)).transpose()
        struc_indx.append(indx.astype(int))
    array_struc_indx = np.array(struc_indx, dtype=object)
    return array_struc_indx

# ------------------------------------------------------------------------------------------------------------
# Mask the domain
# ------------------------------------------------------------------------------------------------------------
def mask_dom(zs):
    mask_out = Xin.copy()
    if 0 not in zs:
        return mask_out
    struc_selected                       = np.where(zs==0)[0].astype(int)
    indx                                 = np.vstack(array_struc_indx[struc_selected]).astype(int)
    mask_out[:, indx[:,0], indx[:,1], :] = reference[indx[:,0], indx[:,1], :]
    return mask_out

# ------------------------------------------------------------------------------------------------------------
# Calculate the SHAP values
# ------------------------------------------------------------------------------------------------------------
array_struc_indx   = get_structure_indices()
explainer          = shap.KernelExplainer(f, np.zeros((1,nmask)))
kernel_shap_values = explainer.shap_values(np.ones((1,nmask)))
```

This function gets the positions of the pixels inside the groups

Function to mask the input image according to the input coallition

Calculation of the SHAP values

Tutorial

Funded by the European Union
European Research Council
Established by the European Commission

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions

Tutorial

Funded by the European Union

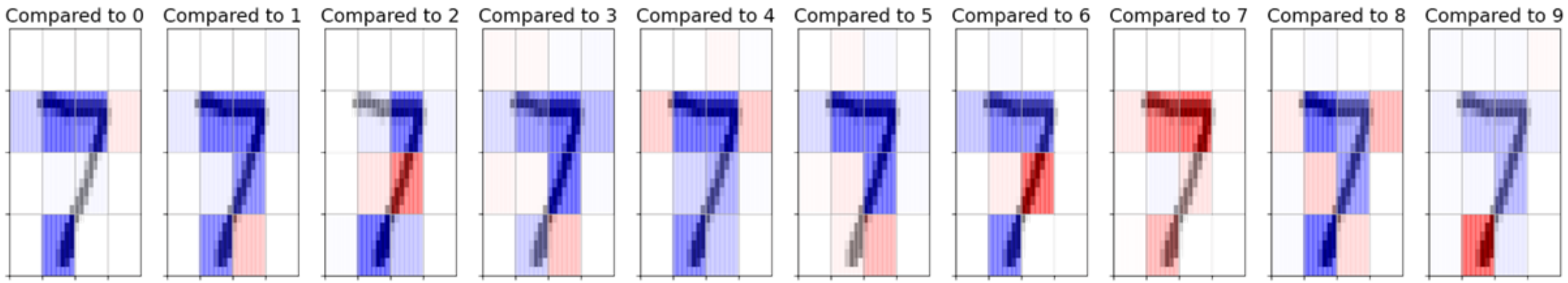
European Research Council
Established by the European Commission

```python
# ------------------------------------------------------------------------------
# Create a matrix of SHAP values
# ------------------------------------------------------------------------------
dim_shap_out    = len(kernel_shap_values)
shap_values_mat = [[] for ind_i in np.arange(dim_shap_out)]
for ind_i in np.arange(len(kernel_shap_values)):
    dim_shap_0             = len(kernel_shap_values[ind_i][:,0])
    dim_shap_1             = len(kernel_shap_values[ind_i][0,:])
    shap_values_mat[ind_i] = np.zeros((size_x,size_y))
    for ind_j in np.arange(dim_shap_0):
        for ind_k in np.arange(dim_shap_1):
            ind_jk                                          = np.vstack(array_struc_indx[ind_k]).astype(int)
            shap_values_mat[ind_i][ind_jk[:,0],ind_jk[:,1]] = kernel_shap_values[ind_i][ind_j,ind_k]

# ------------------------------------------------------------------------------
# Plot the results
# ------------------------------------------------------------------------------
fig, axes = plt.subplots(1, 10, figsize=(25, 4))
for ind_i, ax in enumerate(axes.flat):
    maxshap = np.max(abs(kernel_shap_values[ind_i]))
    ax.set_title("Compared to "+str(ind_i))
    ax.pcolor(np.flip(x_test[0,:,:,0],axis=(0)),cmap="Greys")
    ax.pcolor(np.flip(shap_values_mat[ind_i],axis=(0)),alpha=0.5,vmin=-maxshap,vmax=maxshap,cmap="bwr")
    ax.set_xticks(range(0,size_x,box_size))
    ax.set_yticks(range(0,size_y,box_size))
    ax.grid()
    ax.set_xticklabels([])
    ax.set_yticklabels([])
plt.savefig("kernel_shap.png")
```

Conversion of the SHAP values of the groups to the image pixels

Plot the SHAP values over the image



Red regions make the prediction closer to the objective number and blue makes it more different

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions

Tutorial

Funded by the European Union

European Research Council
Established by the European Commission

- Calculate the Gradient and Deep Explainers.
  - These explainers are model-specific.
  - They work with standard Deep Learning models:
    - Defined in TensorFlow or PyTorch.
  - We will obtain a SHAP value for every pixel of every channel of every input.

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions

Tutorial

Funded by the European Union

European Research Council
Established by the European Commission

```
#%%
# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Calculate the SHAP values using the GradientExplainer
# ------------------------------------------------------------------------------------------------------------------------------------------------------------------

# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Define the input to calculate the SHAP values
# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
Xin = x_test[0].reshape(1,size_x,size_y,1)

# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Since we have two inputs we pass a list of inputs to the explainer. GradientExplainer will calculate a SHAP value for each input feature.
# Any required function should be included in the tensorflow or pytorch model.
# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
explainer          = shap.GradientExplainer(model, [x_train, x_train])
gradient_shap_values = explainer.shap_values([Xin, Xin])

# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Plot the explanations for all classes for the first input (this is the feed forward input)
# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
# Plot the results
# ------------------------------------------------------------------------------------------------------------------------------------------------------------------
fig, axes = plt.subplots(2, 10, figsize=(25, 8))
for ind_i in np.arange(10):
    maxshap = np.max([np.max(abs(gradient_shap_values[ind_i][0][0,:,:,0])),np.max(abs(gradient_shap_values[ind_i][1][0,:,:,0]))])
    axes[0,ind_i].set_title("Compared to "+str(ind_i))
    axes[0,ind_i].pcolor(np.flip(x_test[0,:,:,0],axis=(0)),cmap="Greys")
    axes[0,ind_i].pcolor(np.flip(gradient_shap_values[ind_i][0][0,:,:,0],axis=(0)),alpha=0.5,vmin=-maxshap,vmax=maxshap,cmap="bwr")
    axes[0,ind_i].set_xticks(range(0,size_x,1))
    axes[0,ind_i].set_yticks(range(0,size_y,1))
    axes[0,ind_i].grid()
    axes[0,ind_i].set_xticklabels([])
    axes[0,ind_i].set_yticklabels([])
    axes[1,ind_i].pcolor(np.flip(x_test[0,:,:,0],axis=(0)),cmap="Greys")
    axes[1,ind_i].pcolor(np.flip(gradient_shap_values[ind_i][1][0,:,:,0],axis=(0)),alpha=0.5,vmin=-maxshap,vmax=maxshap,cmap="bwr")
    axes[1,ind_i].set_xticks(range(0,size_x,1))
    axes[1,ind_i].set_yticks(range(0,size_y,1))
    axes[1,ind_i].grid()
    axes[1,ind_i].set_xticklabels([])
    axes[1,ind_i].set_yticklabels([])
    if ind_i == 0:
        axes[0,0].set_ylabel("Input 1")
        axes[1,0].set_ylabel("Input 2")
plt.savefig("gradient_shap.png")
```

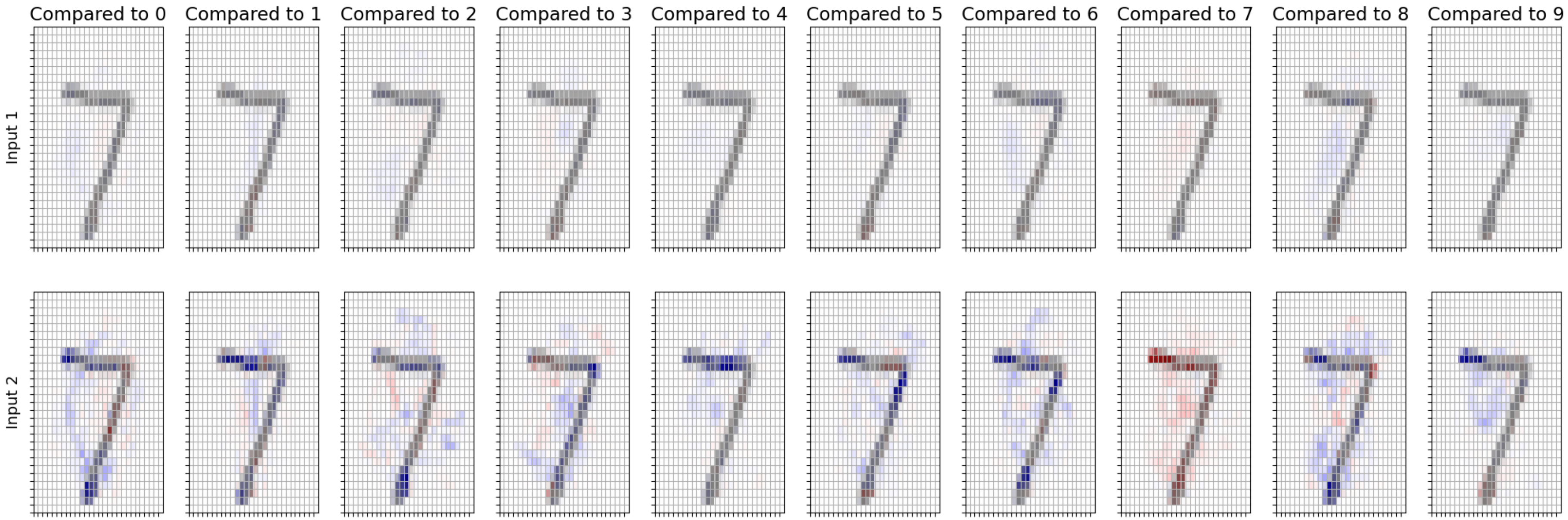
Define the input image

Calculate the SHAP values

Plot the SHAP values of the inputs

Tutorial

Index



- Red regions make the prediction closer to the objective number and blue makes it more different.
- Note that both inputs get different SHAP values even being the same image.

Tutorial

Funded by the European Union

European Research Council
Established by the European Commission

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions

Tutorial

Funded by the European Union

European Research Council
Established by the European Commission

```
#%%
# ----------------------------------------------------------------------------------------------------------------
# Calculate the SHAP values using the DeepExplainer
# ----------------------------------------------------------------------------------------------------------------

# ----------------------------------------------------------------------------------------------------------------
# Define the input to calculate the SHAP values
# ----------------------------------------------------------------------------------------------------------------
Xin = x_test[0].reshape(1,size_x,size_y,1)

# ----------------------------------------------------------------------------------------------------------------
# Since we have two inputs we pass a list of inputs to the explainer. GradientExplainer will calculate a SHAP value for each input feature.
# Any required function should be included in the tensorflow or pytorch model.
# ----------------------------------------------------------------------------------------------------------------
explainer        = shap.DeepExplainer(model, [x_train[:100], x_train[:100]])
deep_shap_values = explainer.shap_values([Xin, Xin])

# ----------------------------------------------------------------------------------------------------------------
# Plot the explanations for all classes for the first input (this is the feed forward input)
# ----------------------------------------------------------------------------------------------------------------
# ----------------------------------------------------------------------------------------------------------------
# Plot the results
# ----------------------------------------------------------------------------------------------------------------
fig, axes = plt.subplots(2, 10, figsize=(25, 8))
for ind_i in np.arange(10):
    maxshap = np.max([np.max(abs(deep_shap_values[ind_i][0][0,:,:,0])),np.max(abs(deep_shap_values[ind_i][1][0,:,:,0]))])
    axes[0,ind_i].set_title("Compared to "+str(ind_i))
    axes[0,ind_i].pcolor(np.flip(x_test[0,:,:,0],axis=(0)),cmap="Greys")
    axes[0,ind_i].pcolor(np.flip(deep_shap_values[ind_i][0][0,:,:,0],axis=(0)),alpha=0.5,vmin=-maxshap,vmax=maxshap,cmap="bwr")
    axes[0,ind_i].set_xticks(range(0,size_x,1))
    axes[0,ind_i].set_yticks(range(0,size_y,1))
    axes[0,ind_i].grid()
    axes[0,ind_i].set_xticklabels([])
    axes[0,ind_i].set_yticklabels([])
    axes[1,ind_i].pcolor(np.flip(x_test[0,:,:,0],axis=(0)),cmap="Greys")
    axes[1,ind_i].pcolor(np.flip(deep_shap_values[ind_i][1][0,:,:,0],axis=(0)),alpha=0.5,vmin=-maxshap,vmax=maxshap,cmap="bwr")
    axes[1,ind_i].set_xticks(range(0,size_x,1))
    axes[1,ind_i].set_yticks(range(0,size_y,1))
    axes[1,ind_i].grid()
    axes[1,ind_i].set_xticklabels([])
    axes[1,ind_i].set_yticklabels([])
    if ind_i == 0:
        axes[0,0].set_ylabel("Input 1")
        axes[1,0].set_ylabel("Input 2")
plt.savefig("deep_shap.png")
```

Define the input image

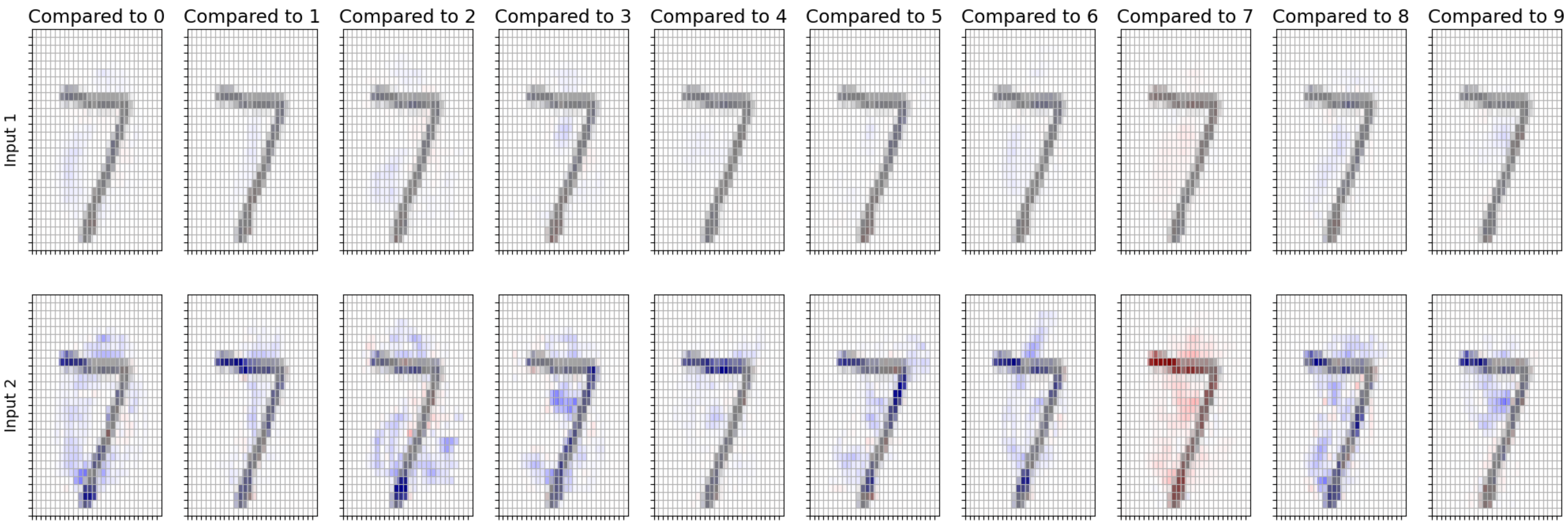
Calculate the SHAP values

Plot the SHAP values of the inputs

Tutorial

Index

- Introduction
- Methodology
- **Tutorial**
- Conclusions



Tutorial

Funded by the European Union

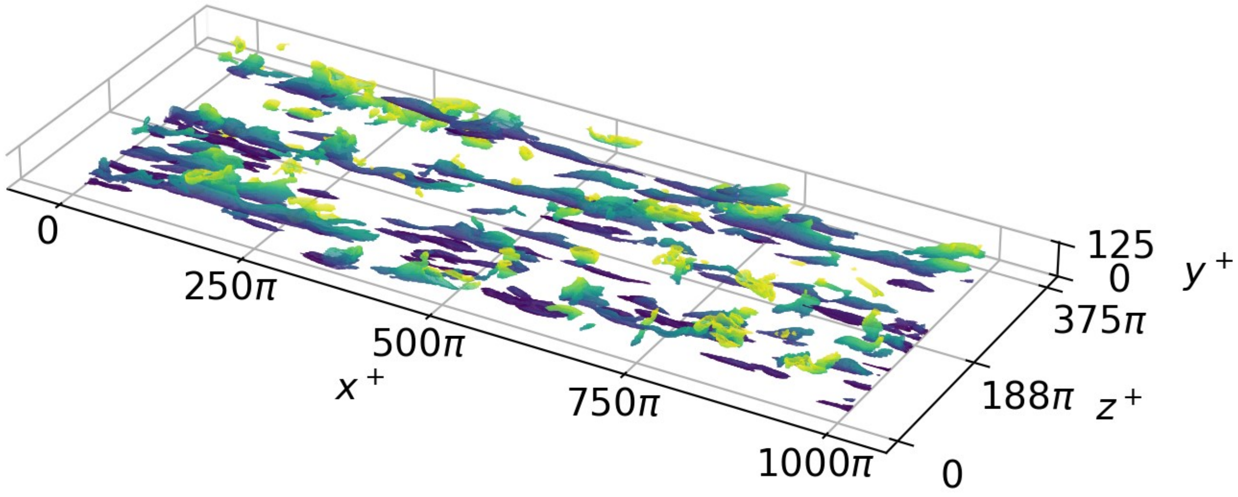
European Research Council
Established by the European Commission

Conclusions

Index

- Introduction
- Methodology
- Tutorial
- **Conclusions**

- Additive-feature-attribution methods:
  - Powerful tool for understanding complex models:
    - FLUID MECHANICS
- Understanding the most important inputs for the predictions:
  - Link models with physics.
    - Calculation of high importance regions in turbulent flows:



Review of SHAP for fluid dynamics and heat transfer

Funded by the European Union

European Research Council
Established by the European Commission

Conclusions

Index

- Introduction
- Methodology
- Tutorial
- **Conclusions**

Review of SHAP for fluid dynamics and heat transfer

Funded by the European Union

European Research Council
Established by the European Commission

- Link to our research articles:

Cremades, A., Hoyas, S., & Vinuesa, R. (2025). Additive-feature-attribution methods: a review on explainable artificial intelligence for fluid dynamics and heat transfer. *International Journal of Heat and Fluid Flow*, *112*, 109662.

Cremades, A., Hoyas, S., Deshpande, R., Quintero, P., Lellep, M., Lee, W. J., ... & Vinuesa, R. (2024). Identifying regions of importance in wall-bounded turbulence through explainable deep learning. *Nature Communications*, *15*(1), 3864.

Cremades, A., Hoyas, S., & Vinuesa, R. (2024). Classically studied coherent structures only paint a partial picture of wall-bounded turbulence. *arXiv preprint arXiv:2410.23189*.