# Transfer Learning in Turbulent Systems

*Thesis to be submitted in partial fulfillment of the requirements for the degree*

*of*

**Mtech in Artificial Intelligence/Machine Learning**
**Foundations and Applications**

*by*

**Sachin Prasad**
**19MF3AI23**

Under the guidance of

**Dr. Rajaram Lakkaraju, Dr. Adway Mitra**


**CENTRE OF EXCELLENCE IN ARTIFICIAL INTELLIGENCE**

**INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR**

Department of Centre of Excellence in Artificial Intelligence
Indian Institute of Technology,
Kharagpur
India - 721302

---

# CERTIFICATE

This is to certify that we have examined the thesis entitled **Transfer Learning in Turbulent Systems**, submitted by **Sachin Prasad**(Roll Number: *19MF3AI23*) a postgraduate student of **Department of Centre of Excellence in Artificial Intelligence** in partial fulfillment for the award of degree of Mtech in Artificial Intelligence/Machine Learning Foundations and Applications. We hereby accord our approval of it as a study carried out and presented in a manner required for its acceptance in partial fulfillment for the Post Graduate Degree for which it has been submitted. The thesis has fulfilled all the requirements as per the regulations of the Institute and has reached the standard needed for submission.

---

## Supervisors

**Department of Centre of Excellence in Artificial Intelligence**
Indian Institute of Technology,
Kharagpur

**Place: Kharagpur**
**Date:**

# ABSTRACT

Transfer learning is a technique which enables neural networks to generalize out of distribution via targeted re-training. It is becoming a powerful tool in scientific machine learning applications such as weather, climate prediction and turbulence modeling.

Turbulent flows are difficult to simulate numerically because they contain eddies of various sizes and energies and the smaller eddies may not be represented if the mesh isnt small enough. Large Eddy Simulation (LES) is a technique where we filter out the smaller eddies and use a sub-grid model to represent their effects on the flow. The biggest challenge of LES simulations is developing an accurate sub-grid scale model. We use neural networks to create data-driven sub-grid scale models and use transfer learning to make them generalizable to different flows of higher turbulence.

In my previous project, I had worked on creating data-driven sub-grid scale models for the problem of 2 dimensional turbulence. In this project I have used this framework for the problem of Rayleigh Benard Convection.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background and MTP 1 work

There is a growing interest in developing data-driven subgrid-scale (SGS) models for large eddy simulation (LES) using deep learning techniques. While they are promising, inabilities of these data-driven models to generalize to a different flow (like with a higher Reynolds number) remain as major obstacles in broadening the applications of such data-driven SGS models. We use transfer learning to re-train the layers of our network with a small amount of data from the new flow and show that this method can be used to create accurate and generalizable sub-grid scale models [5, 1].

Previously, I had worked on creating data-driven SGS models for the problem of 2D turbulence. In this project, I have used this framework for the problem of Rayleigh-Benard convection which is a buoyancy driven flow in a container with a temperature gradient.

We show that transfer learning can be applied to create SGS models for Rayleigh-Benard convection that can generalize to higher Rayleigh numbers.

## 1.2 Layout of the Report

The report is organized as follows:

- **Chapter 2 - Turbulence:** In this chapter, we introduce the Navier-Stokes equations and the Reynolds number. Then we explain what is turbulence and why its difficult to simulate.
- **Chapter 3 - Approaches to model turbulence:** Here, we discuss a few methods to simulate turbulence flows and compare the pros and cons of each method. We show how neural networks can be used to improve Large Eddy Simulations.
- **Chapter 4 - Pseudo-Spectral Methods:** In this chapter, we discuss how to numerically solve the partial differential equations using pseudo-spectral methods, whether its 2D turbulence or Rayleigh Benard convection. We compare it to finite difference techniques and show why its more accurate.
- **Chapter 5 - Neural Networks:** Here, we introduce Neural networks as a function approximation tool. We show why Convolutional Neural Networks are more useful for our particular use case.
- **Chapter 6 - Rayleigh Benard Convection:** In this chapter, we introduce the problem of Rayleigh Benard convection. We explain the problem setup and show the corresponding partial differential equations. We then derive the LES equations and the sub-grid stress terms.
- **Chapter 7 - Model Training:** In this chapter we show the parameters we have chosen for the base and target system. We explain how to filter the velocity and temperature fields and calculate the sub-grid terms to create our training data.
- **Chapter 8 - Results:** In this chapter, we compare the effect of re-training different layers of our network and explain which layers are the best to re-train.
- **Chapter 9 - Future Scope:** Here we discuss improvements we can make to our framework.

# Chapter 2

# Turbulence

## 2.1 Introduction

In this chapter, we introduce the Navier-Stokes equation and an important parameter which is Reynolds Number ($Re$), which helps us to characterize turbulence. We explain why turbulent flows are difficult to simulate numerically and how neural networks can help us build sub-grid models which can solve this problem.

## 2.2 The Navier-Stokes Equation

The Navier–Stokes equations are partial differential equations which describe the motion of viscous fluid substances. They mathematically express momentum balance for Newtonian fluids and making use of conservation of mass. They are non linear and do not have an analytical solution. The equations are:

$\frac{\partial v}{\partial t} + (v.\nabla)v = -\frac{1}{\rho}\nabla p + \nu\nabla^2 v + f$

$\nabla.v = 0$

where:

$\rho$ represents the density

$v$ represents the fluid velocity vector

$\nabla$ is the divergence operator

$p$ is the pressure field

$t$ is time

$f$ represents body accelerations like gravity, inertial accelerations etc.

## 2.3 The Reynolds Number

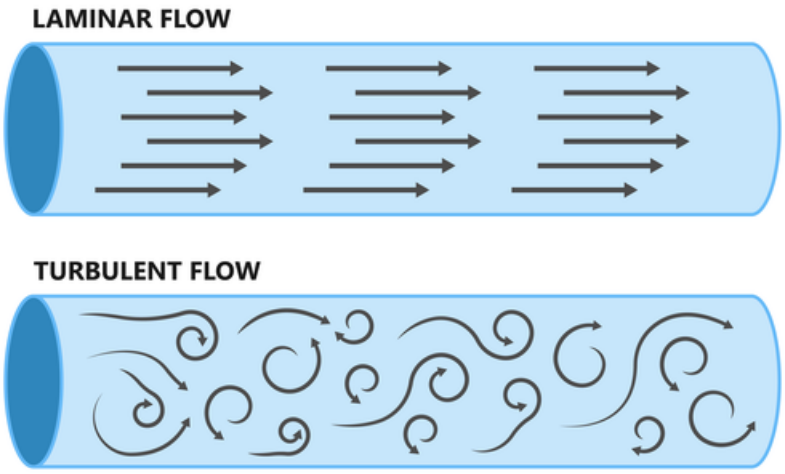
The Reynolds Number is defined as the ratio of the inertial force to the viscous force, it is calculated as:

$Re = \frac{|(v.\nabla)v|}{|\nu\nabla^2 v|}$

where:

$L$ is the length scale $v$ is the velocity $\nu$ is the kinematic viscosity

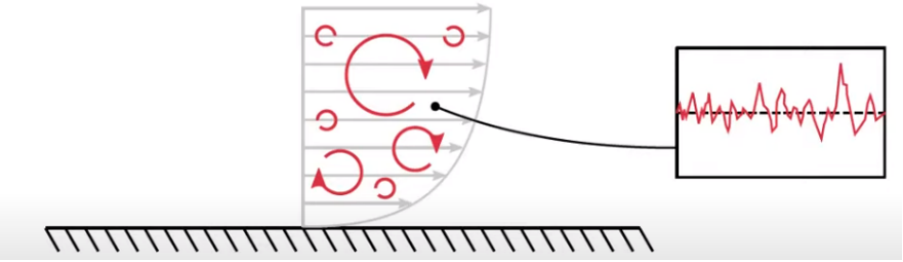
At low Reynolds numbers, flows tend to be dominated by laminar (sheet-like) flow, while at high Reynolds numbers, flows tend to be turbulent. The turbulence results from differences in the fluid's speed and direction, which may sometimes intersect or even move counter to the overall direction of the flow which are eddy currents.




Turbulence is caused by excessive kinetic energy in parts of a fluid flow, which overcomes the damping effect of the fluid's viscosity. For this reason turbulence is commonly realized in low viscosity fluids. In turbulent flow, unsteady vortices appear of many sizes which interact with each other, consequently drag due to friction effects increases. Engineering flows are mostly of turbulent nature and we need to solve for them.

## 2.4 Eddies

An eddy is the swirling of a fluid and the reverse current created when the fluid is in a turbulent flow regime. It is a movement of fluid that deviates from the general flow of the fluid. An example for an eddy is a vortex which produces such deviation. Turbulent flows contain eddies with a range of sizes and energies.



## 2.5 Meshing in CFD

The Navier-Stokes equation is a non-linear partial differential equation which does not have an analytical solution. Computational Fluid Dynamics (CFD) is the process of mathematically predicting physical fluid flow by solving the governing equations using computational power.

Meshing refers to the process of dividing a physical domain into smaller, interconnected elements to facilitate the numerical simulation of fluid flow and other physical phenomena. The resulting mesh, or grid, is used to discretize the governing equations of fluid dynamics, allowing for the solution of complex problems through numerical methods.

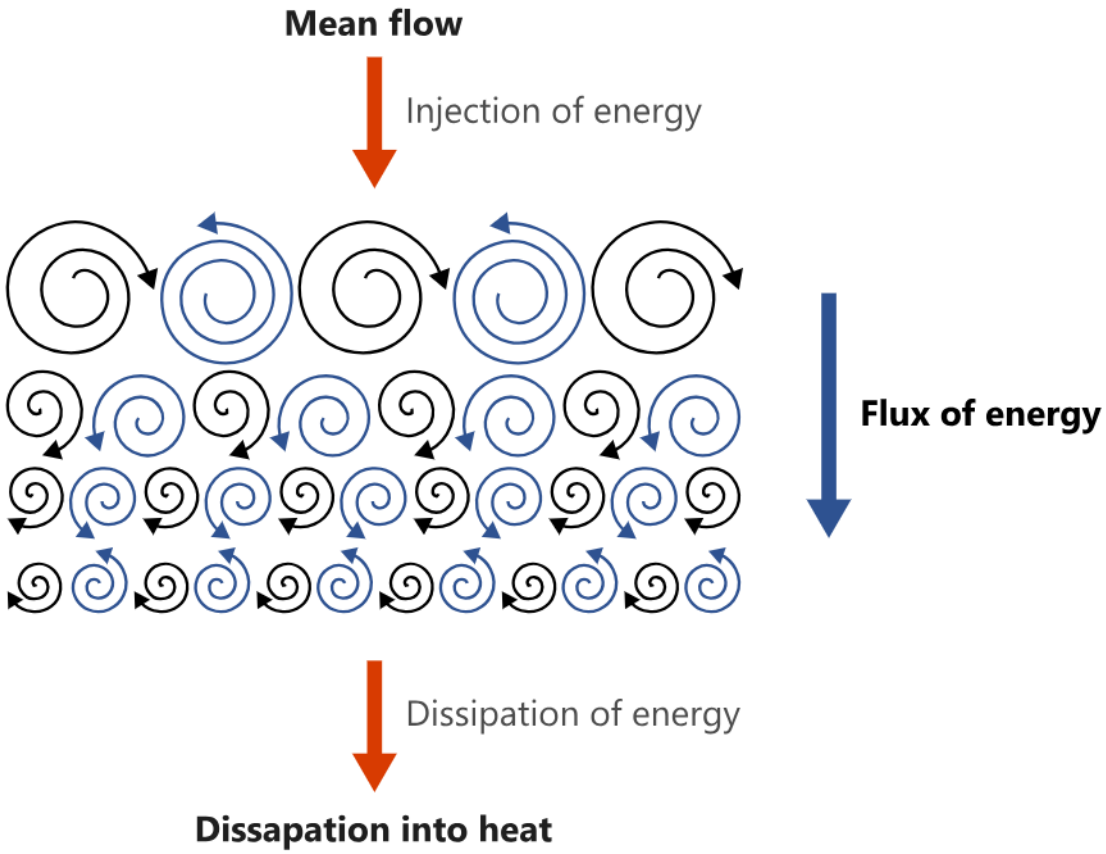
There is a trade-off between grid density, solution accuracy, and computation time. CFD simulations need to balance solution accuracy and computation time by adjusting the mesh density.

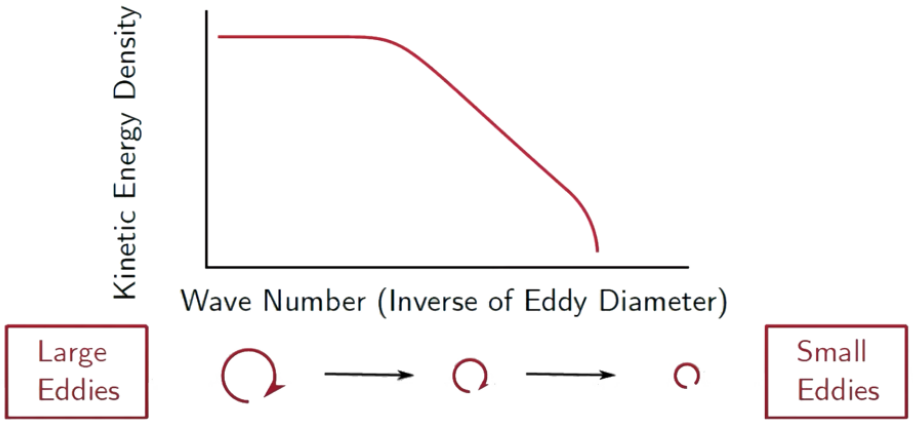
## 2.6 Turbulence Energy Cascade

Turbulence is composed of turbulent eddies of different sizes. At high Reynolds numbers, a scale separation exists between the largest eddies and smallest eddies.

The largest eddies extract kinetic energy from the mean flow as the energy production. The length scale is comparable to the flow dimensions. These processes are highly anisotropic and mostly are not influenced by viscosity. Most transport and mixing happen in this range.

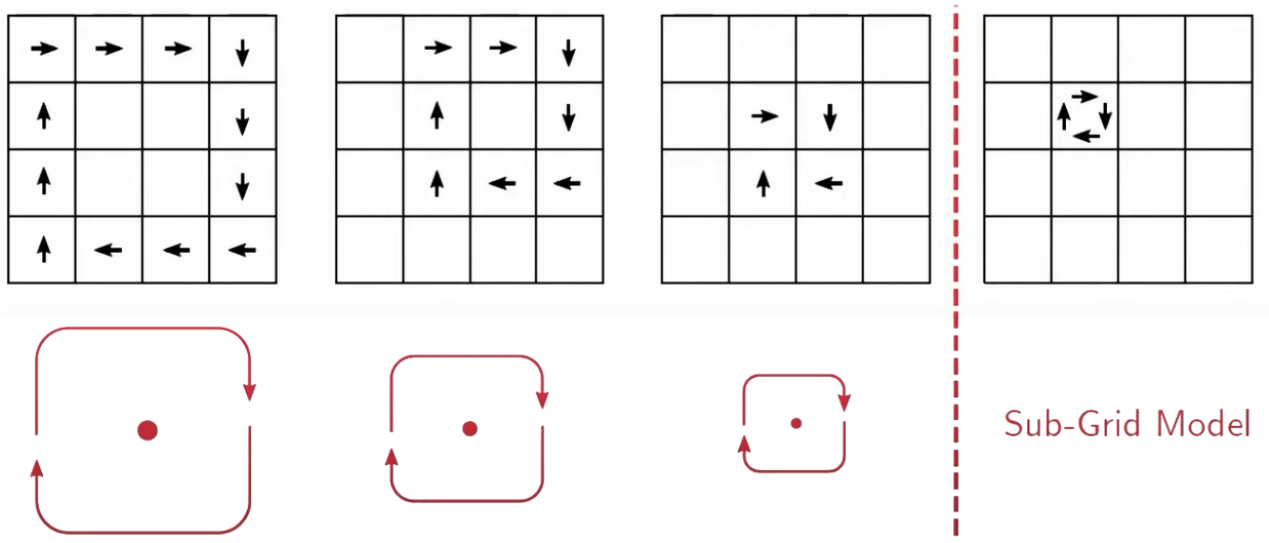
The smallest eddies have universal characters independent of the flow geometry and conditions. Those eddies in this range usually receive energy from the larger eddies and dissipate their energy into heat through the fluid's molecular viscosity. These eddies are isotropic with length scales as described by Kolmogorov scales.






## 2.7 Difficulty in Simulating Turbulence

Since Turbulent flows contain eddies, the mesh cannot resolve eddies smaller than the mesh size itself because we only know the velocity at the cell centroids. We either need to make the mesh fine enough so that it can resolve eddies of all sizes, or model the effect of the eddies which cannot be resolved.



# Chapter 3

# Approaches to Model Turbulence

## 3.1 Introduction

In this chapter, we discuss different methods to model turbulent flows and how to model the effect of small-scale eddies.

## 3.2 Direct Numerical Simulation

In Direct Numerical Simulation (DNS), we solve the extensive range of temporal and spatial scales of a turbulent flow, from very large to very small, down to the Kolmogorov length scale.

It can be estimated that the mesh resolution and time steps required to correctly solve the complexity of the fluid structures scales approximately with the cube of Reynolds number. This makes the DNS approach virtually impossible for engineering applications.

DNS is indeed almost exclusively used in academia and research institutions to model simple flows and, along with experiments, it is used to improve the understanding of turbulence and to develop simplified turbulence models that are less expensive to calculate but still useful to predict the main contribution of turbulence on the flow.

## 3.3 Reynolds Averaged Navier-Stokes

An averaging operation can be applied to the Navier-Stokes equations to obtain the mean equations of fluid flows called Reynolds Averaged Navier-Stokes (RANS) equations. These are very similar to the original equations but contains some additional terms in the momentum equations called Reynolds stress terms that are unknown and need to be modelled.

Turbulence models aim to represent the effect of turbulence via the closure of unknown Reynolds stress terms. All RANS models have some limitations due to the modelling assumptions used to derive the mathematical formulation of the model.

## 3.4 Large Eddy Simulation

In Large Eddy Simulations (LES), the smallest scales of turbulence are spatially filtered out while the largest, most energy containing scales are resolved directly.

Due to the nature of turbulence, at a very small scale, the flow structures tend to be similar to each other even in different applications. This allows the use of simpler turbulence models that tend to be more universal and can be applied to several applications with a reduced requirement of model tuning.

Similarly to RANS modelling, in LES turbulence models aim at resolving the unknown terms in the filtered Navier-Stokes equations, called the Sub-grid Scale stresses. The term comes from the fact that in most LES models, the filtering of the equations is obtained at mesh size level, relegating the modelling to flow scales smaller than the grid size.

## 3.5 Filtering operation in LES

This filtering process is crucial for LES because it simplifies these complex Navier-Stokes equations.

The kernel works by focusing on the larger eddies in the fluid, which are the main drivers of flow patterns in turbulence, and averages out the effects of the smaller

eddies. This is done through a mathematical operation known as convolution, which blends the fluid's characteristics over a particular scale.

The filtering operation in Large Eddy Simulation (LES) is generally expressed in the following integral form:

$\bar{\phi}(x,t) = \int_{-\infty}^{\infty} \phi(r,\tau) G(x-r, t-\tau) dr d\tau$

where:

$\bar{\phi}(x,t)$ represents the filtered field variable, which could represent quantities like velocity or pressure.

$\phi(r,\tau)$ represents the unfiltered field variable.

$G(x-r,t-\tau)$ is the filtering kernel applied to the field variable. Usually, Gaussian filters are used.

After filtering, the variable is split into 2 parts: The filtered component ($\bar{\phi}$) and the sub-grid scale component ($\phi'$).

**The Filtered Component ($\bar{\phi}$)** is the larger-scale part of the flow, which the LES filter keeps. It represents the averaged effect of the flow across larger eddies and structures.

**The Sub-grid Scale Component ($\phi'$)** is the smaller-scale part that the LES filter does not directly compute. It includes all the tiny eddies and fluctuations that are too small to be captured by the filter.

Together, the total field variable is described as the sum of both the filtered and sub-grid components: $\phi = \bar{\phi} + \phi'$

## 3.6 DNS vs RANS vs LES

All turbulence models have strengths and limitations due to the nature of the modelling assumptions.

| Feature | DNS | LES | RANS |
|---|---|---|---|
| Applications | Simple flow, low Re number | Simple flow, low Re number | Wide range |
| Computational Power | Very high demand | Less demand than DNS, higher than RANS | Least demand |
| Accuracy | Most accurate | More accurate than RANS, less than DNS | Least accurate |
| Complexity | Can't handle complex systems due to computational load | Handles more complexity than RANS | Can handle complex systems |
| Turbulence | Resolves all scales of turbulence | Resolves large scales, models small scales | Models all scales of turbulence |
| Suitability | Academic and theoretical studies | Flows with significant large-scale turbulence | Industrial and practical applications where efficiency is key |
| Cost | Prohibitively high for most practical applications | High but more feasible than DNS | Economical and practical |
| Typical Use Case | Fundamental research, low Reynolds number flows | Transitional and complex turbulence not suited for RANS | Flows where turbulence is well-behaved and predictable |

The main challenge in LES simulations is having an accurate sub-grid model which can accurately model the effect of the sub-grid scale effects as a function of the filtered variables. Deep Learning techniques are extremely effective for approximating functions when there is sufficient data available. Here, we train neural networks to accurately predict the sub-grid scale terms.

# Chapter 4

# Pseudo-spectral Methods

## 4.1 Introduction

In this chapter, we see how to solve the partial differential equations using numerical methods. We discuss Pseudo-spectral methods and compare them to finite difference methods.

## 4.2 Pseudo-Spectral Methods

Pseudo-spectral methods are a class of numerical methods used in applied mathematics and scientific computing for the solution of partial differential equations. They are closely related to spectral methods, but complement the basis by an additional pseudo-spectral basis, which allows representation of functions on a quadrature grid.

This simplifies the evaluation of certain operators, and can considerably speed up the calculation when using fast algorithms such as the fast Fourier transform [2].

We define the forward Fourier transform of a 2 dimensional discrete field $u$ (transforming from Fourier to physical space) as:

$$u_{i,j} = \sum_{m=-N_x/2}^{N_x/2-1} \sum_{n=-N_y/2}^{N_y/2-1} \tilde{u}_{m,n} e^{i(\frac{2\pi m}{L_x} x_i + \frac{2\pi n}{L_y} y_j)}$$

We then define the backward Fourier transform (transforming from physical to Fourier space) as:

$$\tilde{u}_{m,n} = \frac{1}{N_x N_y} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} u_{i,j} e^{-i(\frac{2\pi m}{L_x} x_i + \frac{2\pi n}{L_y} y_j)}$$

The transforms are normalised, a forward, followed by a backward transform recover the initial discrete field. In this case we have normalised the backward transform. The wave numbers are defined as:

$k_x = \frac{2\pi m}{L_x}$ , $k_y = \frac{2\pi n}{L_y}$

The $N_x$ discrete collocation grid points are uniformly spaced in our domain $[L_x, L_y]$ as:

$x_i = \frac{iL_x}{N_x}$, $y_i = \frac{iL_y}{N_y}$,

Because the problem is periodic, $x_0 = x_{N_x}$, we do not include the last grid point in the simulations, as collocation methods using periodic basis functions automatically impose the periodicity on the problem.

By transforming a discrete physical field to Fourier-space, what would otherwise be done using a differential scheme, derivatives are evaluated by multiplying the Fourier coefficients by the corresponding complex wave number. The *nth* derivative of a discrete field $u$ with respect to $x$ is given by:

$$\frac{\partial^{(n)} u_{i,j}}{\partial x^{(n)}} = \sum_{m=-N_x/2}^{N_x/2-1} \sum_{n=-N_y/2}^{N_y/2-1} \tilde{u}_{m,n} (ik_x)^{(n)} e^{i(\frac{2\pi m}{L_x} x_i + \frac{2\pi n}{L_y} y_j)}$$

The only errors associated with this operation are the interpolation and truncation errors, this is why spectral methods are very powerful.

## 4.3 Time Integration

Using spectral operators for the spatial derivatives ensures that the discretization error of spatial terms is kept to a minimum. In order to keep the global discretization errors low, high-order temporal integrator are required.

The Runge–Kutta methods are a family of implicit and explicit iterative methods which include the Euler method, used in temporal discretization for the approximate solutions of simultaneous nonlinear equations.

Consider the general ordinary differential equation as:
$y'(t) = f(t, y(t))$
The 4th order Runge-Kutta Method is defined as:

$$y_{n+1} = y_n + h(k_1/6 + k_2/3 + k_3/3 + k_4/6)$$

where:

$$k_1 = f(t_n, y_n)$$
$$k_2 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_1)$$
$$k_3 = f(t_n + \tfrac{h}{2}, y_n + \tfrac{h}{2}k_2)$$
$$k_4 = f(t_n + h, y_n + hk_3)$$

The local truncation error for this method is $O(h^5)$

# Chapter 5

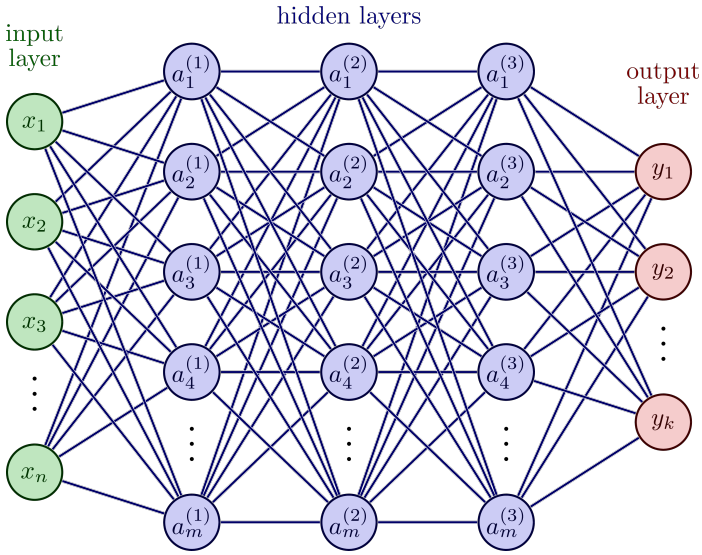
# Neural Networks

## 5.1 Introduction

In Deep Learning learning, a neural network is a model inspired by the structure and function of biological neural networks in animal brains.

An Neural Network consists of connected units or nodes called artificial neurons, which loosely model the neurons in a brain. These are connected by edges, which model the synapses in a brain.

Each artificial neuron receives signals from connected neurons, then processes them and sends a signal to other connected neurons. The signal is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs, called the activation function. The strength of the signal at each connection is determined by a weight, which adjusts during the learning process.



## 5.2 Neural Networks vs Traditional ML Models

Deep Learning out perform other techniques if the data size is large. When there is lack of domain understanding for feature introspection, Deep Learning techniques outshines others as we have to focus less on feature engineering, and it really shines when it comes to complex problems like image classification, natural language processing and speech recognition, etc.

## 5.3 Fully Connected Neural Networks

In a fully connected neural network (FNN), every neuron has a connection with the node of the previous layer output.

The equation for the neural network is a linear combination of the independent variables and their respective weights and bias term for each neuron. The neural network equation is:

$a_j^{(l)} = f(\Sigma_{i=1}^{N_{l-1}} w_{ji} a_i^{(l-1)} + b_j)$

where:

$a_j^{(l)}$ is the $j$th activation in layer $l$

$f$ is the activation function

$w$ are the weights

$b$ is the bias

## 5.4 Activation Functions

Activation functions introduce non-linearity in the network. There are several types of functions we can use, the most common are the ReLU, Sigmoid and the Hyperbolic tangent function.

The ReLU function is: $ReLU(x) = max(x, 0)$

The Sigmoid function is: $\sigma(x) = \frac{1}{1+e^{-x}}$

The Hyperbolic tangent function is $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

Usually the ReLU activation is used in hidden layers and the sigmoid activation in the output layer.

## 5.5 Problems with Fully Connected Networks

Using a fully connected network for a task like image classification can be impractical due to the large number of parameters involved.

In a fully connected network, each neuron in one layer is connected to every neuron in the next layer, leading to a very high number of connections and parameters.

This can result in a computationally expensive and memory-intensive model, making it difficult to train and prone to overfitting. Additionally, fully connected networks may not effectively capture the spatial structure and local dependencies present in images, which are important for accurate classification. For these kind of tasks, we use a Convolutional Neural Network.

## 5.6 Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of deep neural network architecture commonly used in Computer Vision. The main feature of a CNN is the convolutional layer. Unlike an artificial neuron in a fully-connected layer, a neuron in a convolutional layer is not connected to the entire input but just some section of the input data.

Consider an input $x$ which is an $m \times n$ matrix, now consider a convolutional kernel $H$ which is an $l \times l$ matrix, $l$ is usually taken to be a small integer like $3, 5, 7$ etc. The 2D convolution of the input $x$ with the kernel $H$ is another matrix $y$ which is given by the equation:

$y_{ij} = \sum_{u=1}^{l} \sum_{s=1}^{l} H_{us} x_{i+u,j+s}$

The output of such a two dimensional convolution at index $(i, j)$ covers a box in the input matrix $x$ at the neighbourhood of $(i, j)$ with the kernel $H$.

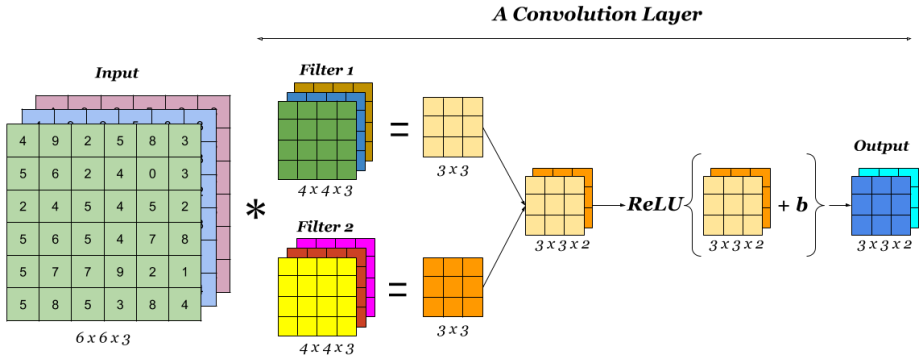



Figure 5.1: Convolutional Layer

Since our data involves velocity and temperature fields which are spatio-temporal in nature, we use CNNs to train our sub-grid models.

## 5.7 Out-of-distribution generalization

Neural Networks have been used to improve simulations or predictions of nonlinear, multi-scale, high-dimensional systems. For example, in thermo-fluid sciences and in weather/climate modeling, a number of different approaches using neural networks have shown significant promise for fully data-driven forecasting, subgrid-scale closure modeling, and ways of solving partial differential equations.

One major challenge is the inability of neural networks, to generalize out-of-distribution, and to perform equally well when tested on a dataset whose distribution is different from the training set. Some degree of such out-of-distribution generalization is essential for neural networks to be practically useful in many applications.

## 5.8 Use of transfer learning

Transfer learning is a technique in machine learning where a model trained on one task is used as the starting point for a model on a second task. This can be useful when the second task is similar to the first task, or when there is limited data available for the second task. By using the learned features from the first task as a starting point, the model can learn more quickly and effectively on the second task.

Transfer learning provides a powerful and flexible framework for improving the out-of-distribution generalization of neural networks, and has shown success in various Machine Learning applications. Here, we try to create sub-grid models and then extrapolate them to more turbulent systems using transfer learning.

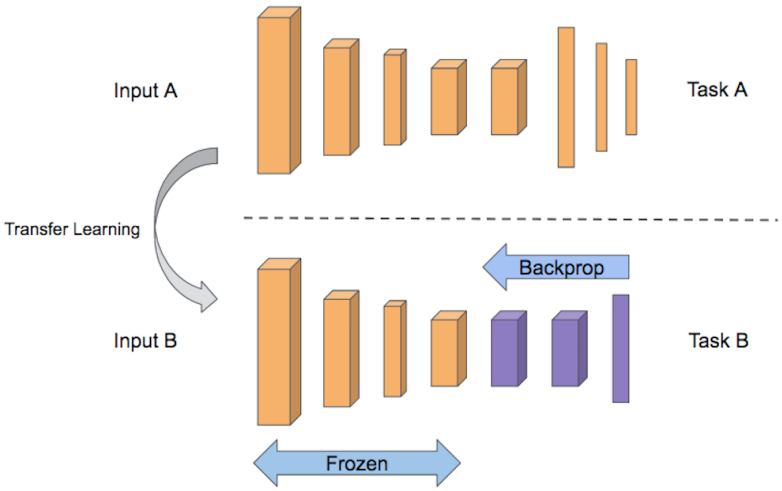



Figure 5.2: Transfer Learning process

# Chapter 6

# Rayleigh Benard Convection

## 6.1 Introduction

Convective flows are frequently encountered in natural processes like atmospheric and planetary flows, as well as in technological applications like electronic and industrial cooling systems. Rayleigh-Benard convection is an idealized setup used to study thermal convection.

Here, a layer of fluid is confined between two plates which are heated at the bottom and cooled at the top. This causes the hot fluid to rise due to buoyancy while the cold fluid falls. The two governing parameters of Rayleigh Benard Convection are Rayleigh number (Ra), which is the ratio of buoyancy over dissipation, and Prandtl number (Pr), which is the ratio of kinematic and thermal diffusivities.

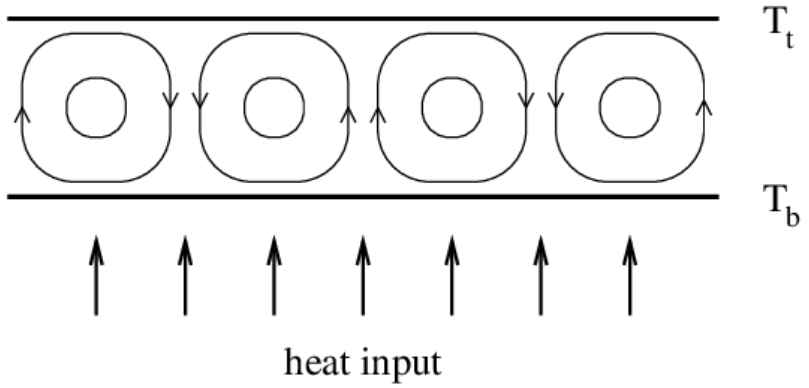



Figure 6.1: A scehmatic of Rayleigh Benard Convection

## 6.2 Equations of Rayleigh-Benard Convection

We solve the incompressible Navier-Stokes equation along with the temperature equation to compute the velocity field $u$ and the temperature field $T$ respectively. The two equations are coupled by the Oberbeck-Boussinesq approximation, where density variations are ignored everywhere except to account for the effects of buoyancy, which manifests as a forcing term in the momentum equation. Non-Boussinesq effects are normally expected for $Ra > 10^{15}$. Thus, the full set of equations are:

$$\frac{\partial u}{\partial t} + u.\nabla u = -\frac{1}{\rho_0}\nabla p + \alpha g T\hat{z} + \nu\nabla^2 u$$
$$\frac{\partial T}{\partial t} + u.\nabla T = \kappa\nabla^2 T$$
$$\nabla.u = 0$$

Here, $p$ is the pressure field, $\alpha$ is the thermal expansion coefficient, $g$ is the gravitational acceleration, $\nu$ and $\kappa$ are the kinematic viscosity and thermal diffusivity respectively, and $\rho_0$ is the constant value of density within the Boussinesq approximation. The equations are non-dimensionalized by the free-fall velocity $u_f = \sqrt{\alpha g \triangle H}$ , imposed temperature difference $\triangle$, and domain height $H$, so that we get:

$$\frac{\partial u}{\partial t} + u.\nabla u = -\nabla p + T\hat{z} + \sqrt{\frac{Pr}{Ra}}\nabla^2 u$$
$$\frac{\partial T}{\partial t} + u.\nabla T = \frac{1}{\sqrt{RaPr}}\nabla^2 T$$

We now have the two dimensionless parameters, Rayleigh number, $Ra = \frac{\alpha g \triangle H^3}{\kappa\nu}$, and Prandtl number, $Pr = \frac{\nu}{\kappa}$.

## 6.3 Deriving the LES Equations

For direct numerical simulations (DNS), the equations are solved in 2 dimensions using a pseudo-spectral solver with high resolution [3], resolving all relevant spatio-temporal scales.

Filtering the DNS equations gives us the Large-eddy simulation (LES) equations [4]. In LES, we solve the DNS equations on a coarse grid which does not resolve all

the scales of the flow. Hence, the velocity, pressure and temperature field are implicitly filtered by the grid spacing, and decomposed into resolved and unresolved parts as, $u_i = \bar{u}_i + u_i'$ , $p_i = \bar{p}_i + p_i'$ and $T_i = \bar{T}_i + T_i'$. We then obtain the filtered governing equations on the LES grid as:

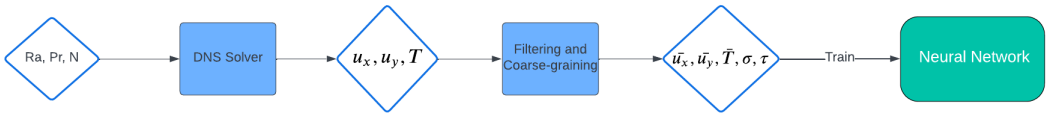
$$\frac{\partial \bar{u}_i}{\partial t} + \bar{u}_j \frac{\partial \bar{u}_i}{x_j} = -\frac{\partial \bar{p}}{\partial x_i} + \bar{T}\delta_{i3} + \sqrt{\frac{Pr}{Ra}} \frac{\partial^2 \bar{u}_i}{\partial x_j^2} - \frac{\partial \tau_{ij}}{\partial x_j}$$
$$\frac{\partial \bar{T}}{\partial t} + \bar{u}_j \frac{\partial \bar{T}}{\partial x_j} = \sqrt{\frac{1}{RaPr}} \frac{\partial^2 \bar{T}}{\partial x_j^2} - \frac{\partial \sigma_j}{\partial x_j}$$

Here $\tau_{ij} = u_i\bar{u}_j - \bar{u}_i\bar{u}_j$ is the subgrid stress tensor, and $\sigma_j = T\bar{u}_j - \bar{T}\bar{u}_j$ is the subgrid scalar flux.

The sub grid terms are to be explicitly represented in terms of the resolved flow variables $(\bar{u}, \bar{T})$. While the LES solver is computationally much cheaper, it requires an accurate closure of $\tau(\bar{u}, \bar{T})$ and $\sigma(\bar{u}, \bar{T})$, which is a major challenge when dealing with turbulent flows.

To build data-driven closures, we train Convolutional Neural Networks (CNN) on filtered and coarse-grained DNS (FDNS) data. The input of the CNNs is $(\bar{u}, \bar{T})$ and the output is $(\tau, \sigma)$ .

After training the network on one system (base system), we can apply transfer learning and re-train the layers of the network using data from another system (target system), so that the network can be used for higher $Ra$ number systems.



## 6.4 Training Setup

For the base and target systems, we consider the following parameters:

| System | $Ra$ | $Pr$ | $N_{DNS}$ | $N_{LES}$ |
|---|---|---|---|---|
| Base | $10^4$ | 0.7 | $128X128$ | $64X64$ |
| Target | $10^6$ | 0.7 | $256X256$ | $64X64$ |

We solve the DNS equations on a square domain $x \in [-\pi, \pi]$, $y \in [0, 2\pi]$ using a pseudo-spectral solver to obtain snapshots of the velocity and temperature fields. For the base system, we run the solver with the following parameters: $Ra = 10^4$, $Pr = 0.7$, $N_x = N_y = 128$.

The dimensions of the variables obtained are as follows:
$u_x \in \mathbb{R}^{tX128X128}$ $u_y \in \mathbb{R}^{tX128X128}$ $T \in \mathbb{R}^{tX128X128}$
where $t$ is the number of timesteps the solver is run for. We then apply filtering and coarse-graining on the velocity and temperature fields to generate filtered DNS (FDNS) data.

## 6.5 Filtering and Coarse-Graining

Using temperature as an example, we first transform the DNS temperature field $T(x, y)$ into the spectral space $\hat{T}(k_x, k_y)$. Then, we apply the Gaussian filter in the spectral space:

$\tilde{\hat{T}}(k_x, k_y) = G(k_x, k_y) \odot \hat{T}(k_x, k_y)$

where the $\odot$ operator denotes element-wise multiplication of matrices. The transfer function of the Gaussian filter is $G(k_{DNS}) = e^{-|k_{DNS}|^2 \triangle_F^2/24}$ where $k_{DNS} = (k_x, k_y)$ and $\triangle_F$ is the filter size. After the filtering operation, coarse-graining is performed to transform the filtered solution from the DNS to LES grid.

$\bar{\tilde{T}}(k_{LES}) = \tilde{\hat{T}}(|k_x| < |k_c|, |k_y| < |k_c|)$ where $k_c$ is the cutoff wavenumber in the spectral space. The dimensions of the filtered variables are as follows:

$\bar{u_x} \in \mathbb{R}^{tX64X64}$ $\bar{u_y} \in \mathbb{R}^{tX64X64}$ $\bar{T} \in \mathbb{R}^{tX64X64}$

## 6.6 Calculating Sub-grid terms

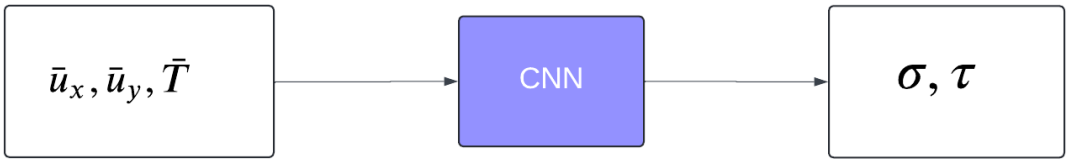
The subgrid terms are calculated using:

$\tau_{ij} = u_i\bar{u}_j - \bar{u}_i\bar{u}_j$ and $\sigma_j = T\bar{u}_j - \bar{T}\bar{u}_j$

The dimensions of the sub grid terms are:

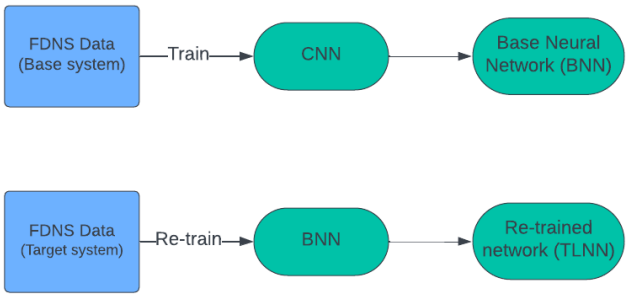
$\tau \in \mathbb{R}^{tX64X64X3}$ $\sigma \in \mathbb{R}^{tX64X64X2}$

We then train our network to predict the sub-grid terms $\tau$ and $\sigma$ using $\bar{u_x}, \bar{u_y}$ and $\bar{T}$ as the inputs. We use a fully convolutional neural network with 11 layers.




## 6.7 Transfer Learning

We then repeat the data generation process using the parameters of the target system. Once we have the filtered variables and the sub-grid terms, we apply transfer learning and re-train the layers of base neural network. We perform this for each layer separately and compare the accuracy for re-training each layer.



## 6.8 Impact of re-training layers

We first train our network on the base system with around $t = 1500$ snapshots. Then we apply transfer learning and re-train each layer on the target system with significantly fewer samples $t = 150$. We use correlation coefficient as the metric and compare the effect of re-training each layer.

Figure 6.2: Temperature profiles of the base and target systems

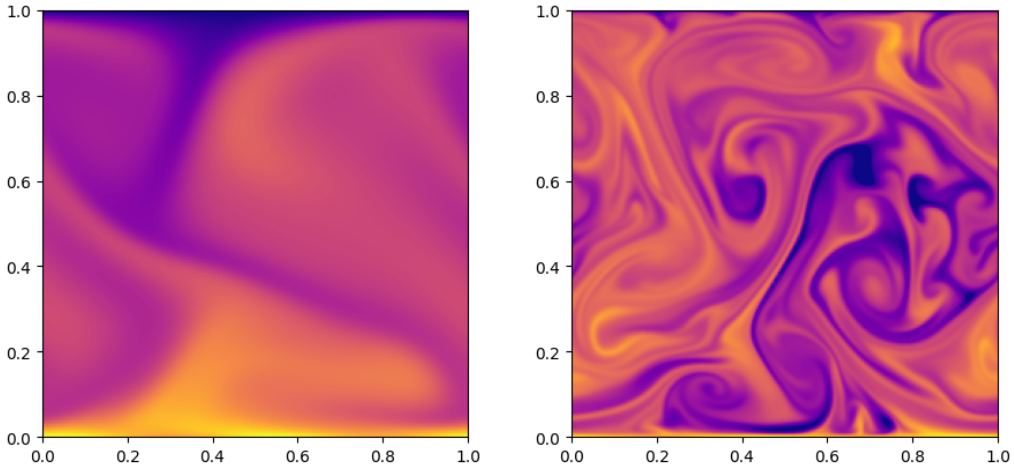
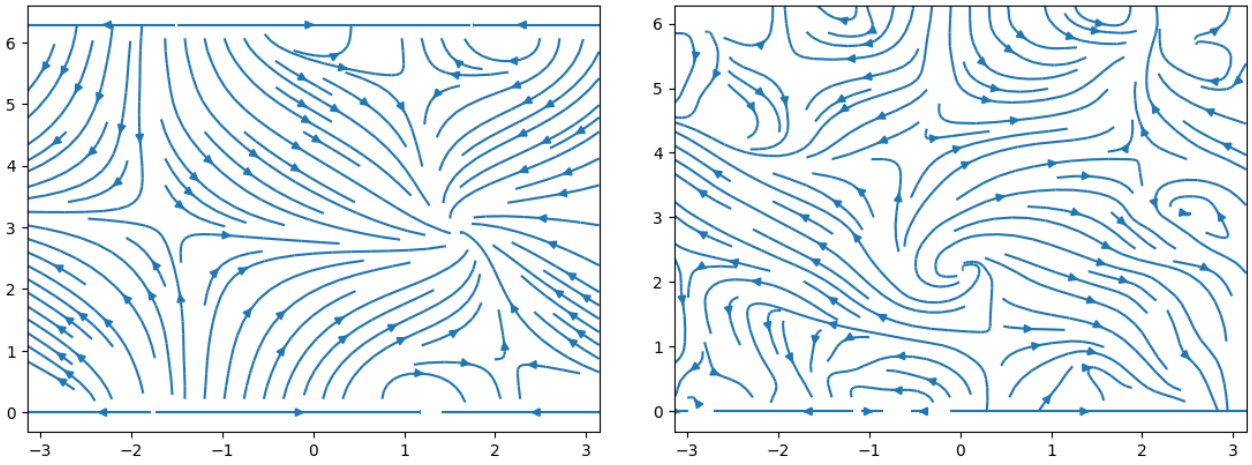
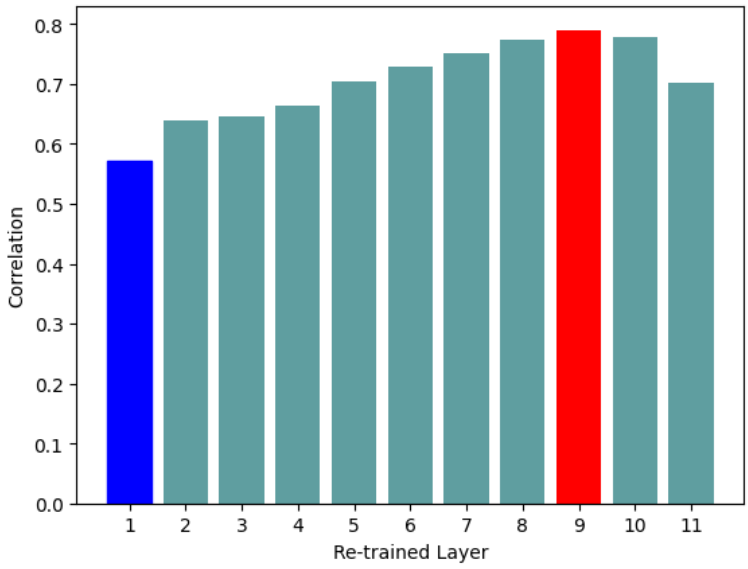


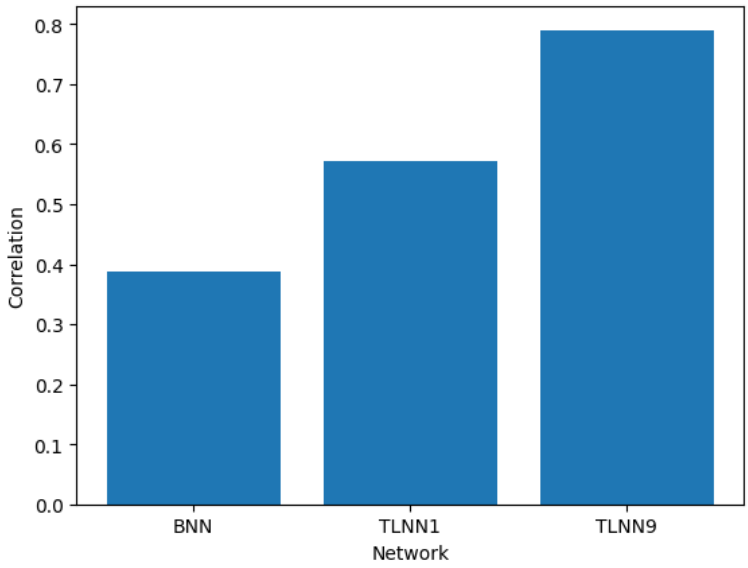

Figure 6.3: Velocity streamlines of the base and target systems



## 6.9 Summary and Conclusion

We can see that re-training layer 9 gives us the best performance, and re-training layer 1 gives us the worst performance. The deeper layers in general gives us better performance than re-training the shallower layers. However, regardless of which layer we re-train, the performance drastically improves compared to the base network. Therefore, transfer learning can be a useful method to train sub-grid models for a system where large amounts of training data are not available.



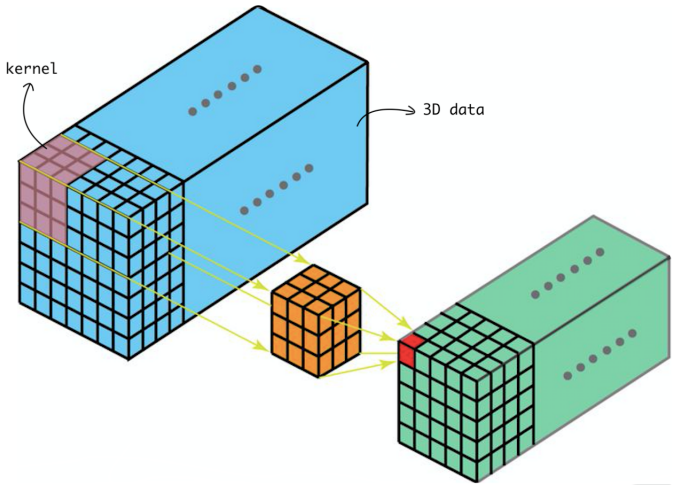



# Chapter 7

# Future Scope

## 7.1 3D turbulence

We can extend this framework to 3 dimensions to create more accurate sub-grid models which can be used in weather/climate prediction models. We can solve the equations of Rayleigh Benard Convections in 3 dimensions and obtain snapshots of the velocity and temperature fields. For creating sub-grid models, we can use 3d convolutional layers.




## 7.2 Multi-phase Flow

We can also try to model multi-phase flows, where we have 2 or more phases in our fluid. Multiphase flow occurs regularly in many natural phenomena, for example sediment transport in rivers is subject to multiphase flow, in which the suspended

particles are treated as a disperse second phase which interacts with the continuous fluid phase. We can use neural networks to create data-driven closures for this type of flow.

## 7.3 Hybrid RANS-LES simulations

The biggest limitation of LES simulations is the application to wall-bounded flows. In regions very close to the wall small-scale turbulence is damped and does not need to be resolved. However, the viscous sublayer thickness is a function of the Reynolds number $Re$ of the flow. At higher $Re$ numbers, the viscous sublayer becomes thinner, thereby allowing the survival of smaller eddies that need to be resolved.

This problem can be solved by blending a RANS model near the wall with a LES model away from the wall.

# Bibliography

[1] Yifei Guan, Ashesh Chattopadhyay, Adam Subel, and Pedram Hassanzadeh. Stable a posteriori les of 2d turbulence using convolutional neural networks: Backscattering analysis and generalization to higher re via transfer learning. *Journal of Computational Physics*, 458:111090, 2022.

[2] Marin Lauber. Pseudo-spectral method, 2021.

[3] Mikael Mortensen. Pseudo-spectral solver, 2019.

[4] Roshan Samuel, Ravi Samtaney, and Mahendra K Verma. Large-eddy simulation of rayleigh–bénard convection at extreme rayleigh numbers. *Physics of Fluids*, 34(7), 2022.

[5] Adam Subel, Yifei Guan, Ashesh Chattopadhyay, and Pedram Hassanzadeh. Explaining the physics of transfer learning in data-driven turbulence modeling. *PNAS nexus*, 2(3):pgad015, 2023.