Deep Learning and Its Application to Fluid Dynamics

Michael Jara

Key words: Machine Learning, Neural Network, Linear/Non-Linear, Turbulence

Abstract

Machine learning and deep learning basics are developed to examine their application to near-wall turbulent flow modeling. This paper takes both a linear and non-linear methodology for reconstructing the velocity, and the results are compared.

Introduction

Machine learning is becoming one of the most popular areas of study and application in the scientific community. It is already applied in many areas such as virtual helpers such as Siri, detected spam, and search engine refinement1. In recent years, machine learning has gotten more attention by physicists. The development of self-improving computational systems has practical application in just about every branch of physics and even other disciplines such as biology and chemistry. This is particularly helpful in situations where solutions to governing physics equations must be approximated. This is where machine learning really shines.

Because of the nature of the aforementioned equations, science has recently turned to using computational techniques such as numerical approximations and machine learning. Equations that govern fluid dynamics such as the Navier-Stokes Equations do not have analytic solutions. Furthermore, fluids have turbulence, which makes reconstructing and forecasting fluid velocity very difficult. This is where supervised deep learning, which is a sub-field of machine learning, is very useful. In the past, there have been limitations for these more complicated computation methods because of the lack of data and computational power, but this issue is quickly disappearing.

This mini review will explain what machine learning is, using supervised deep learning as the primary focus. After the basics of deep learning are reviewed, we will use the example of near-wall turbulent flow to showcase how deep learning can be used to reconstruct velocities using the Burgers equations, and the Naiver-Stokes equations.

Deep Learning and Navier-Stokes

Machine learning is essentially getting a computer to teach itself how to compute something via some training. For the purposes of this paper, if we give it some raw data concerning a fluid’s density, velocity, and other attributes, we would want to get back some program that tells us more about that fluid, and how it will act in the future.

A screenshot of a cell phone

Description automatically generated

Figure 1: A simple visual differentiating traditional programing and ML2. Traditionally, we make programs with a goal to accomplish some task, such as write some line to the console or compute some value, but everything the computer does is a result of your code. With machine learning, the computer takes in data (trains), and later becomes a model. The output in the model is called a program for this reason.

Machine Learning comes in four different varieties. These differentiate the nature of the data they are given; the key here being labeled and unlabeled data, and the ability to have corrective data. These are Supervised, Semi-Supervised, Unsupervised, and Reinforced Learning.2 In Supervised Learning, the data that the model is fed has the desired output. That is to say that the model is aware of what it is trying to calculate and what variables are necessary to complete that task. This is basically a labeling machine, in that the purpose of this method is to put the data in set categories. For example, if we designed a method for detected spam emails, the data we used would be labeled “Spam” or “Not Spam”, and we would continually correct the outputs of our model when feeding it new data. For the more mathematically minded, consider some observable (the email) and the output (Spam or not) . We feed the machine learning algorithm both and , and what we hope to get out is a model that can approximate solutions to , hence its application to fluids.

The first step when working with supervised ML models is to extract the features of the data that are of interest. With raw data, this usually involves cleaning and reformatting, which is nothing more than prepping the data for analysis. After this, we take the data set and split it in two, creating training data and test data. The training data is used to train the chosen ML model. The test data is used after this is done to correct any flaws. Typically, this is where flaws are noticed, and we go back and tweak the model and provide more and more training data until we come up with a model that is consistent with the leftover test data.

Unsupervised Learning involves training data that is not labeled. This is useful for clustering, which is putting things into groups based on common trends. We may not know exactly what those groups mean, but at the very least, this kind of learning can put things into clusters so we may determine that ourselves. Semi-Supervised is somewhere in the middle. We have a few labels in there, but this might be incomplete, or we may not have a way to correct the output from the model. Reinforced learning is a slightly different system that uses the notion of good and bad. These kinds of models interact with the environment and basically score themselves “good” or “bad”. As the model develops, it seeks to maximize how “good” it is. An overview of this is available in figure 2.

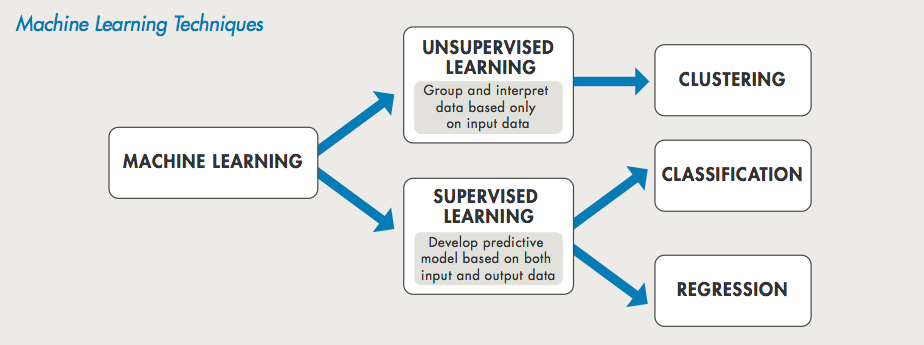


Figure 2: How Machine Learning breaks down3 Generally, these models are either supervised or unsupervised, depending on if the data is labeled or not. Unsupervised data is used for clustering, while supervised learning has application in classification and regression

Supervised Deep learning is the relevant Machine Learning topic for this review, as it is used to model turbulence, something we will get into later. This Supervised Learning topic uses neurons, or nodes, as the basic unit for deep learning, part of the reason it is sometimes referred to as Neural Networks(NN). This special branch of Machine Learning is best used when the data comes in more complicated formats, such as pictures, text, or in our case, fluid dynamics. The result of this model is prediction.

Figure 3 demonstrates this, as well the key difference between the traditional Machine Learning workflow and Deep Learning. Traditionally, the person using the model would be responsible for feature extraction, but the DL model does this all on its own. It just needs raw data. One drawback of this is that it typically requires a very large amount of data to produce reasonable predictions.

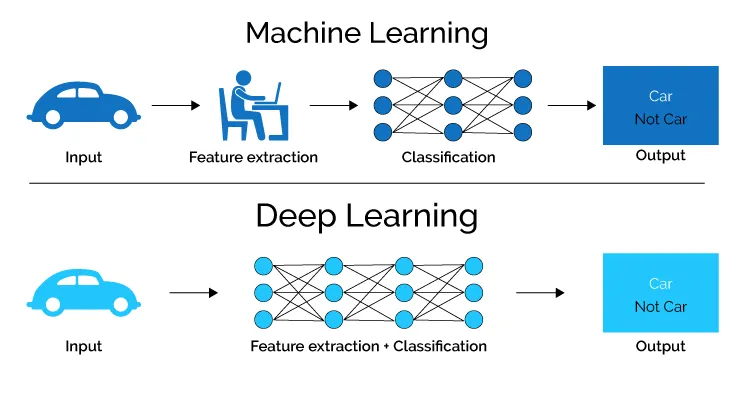


Figure 3: A visual for the key differences between other ML models and DL4

Generally, each Deep Learning Algorithm has multiple layers that each provide a level of complexity to the network. How each node contributes to the final product is determined by the weight, represented by the lines connecting the nodes, as seen in the figure 4

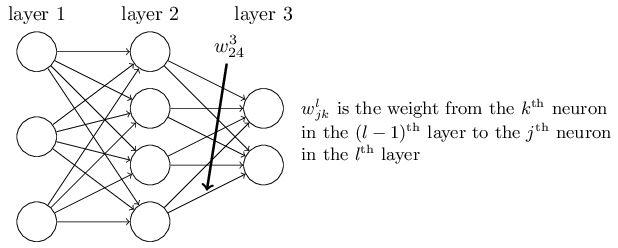


Figure 4: A 3 layer deep learning model 5. A NN is composed of there neurons and weights, in which information is moved from one layer to the next with weight w.

We will use deep learning techniques to model turbulence near a wall as an example for why these techniques are so useful. For now, we’ll quickly shift focus to the Navier-Stokes equations . In general, these are a system of nonlinear partial differential equations which describe fluid behavior with pressure , force, and velocity . is the gradient, and is the Laplacian, also sometimes written as . is the kinematic viscosity, which is essentially the fluid's resistance to flow because of gravity . Note that the bolded letters are vectors (the problem is in 3 dimensions). This creates three equations for the velocities and the pressure, one for each direction. It can be shown that non-dimensionalizing these equations provides the result . is the Reynolds number, which helps characterize turbulence. Re will also depend on the viscosity and the height of the wall (for this near-wall example), given by , being the height of the channel, and standing for.

One way to approximate the flow from NS is via the POD method, standing for proper orthogonal decomposition. This is to break-down the flow velocity into its mean flow and fluctuating components . The fluctuating flow is calculated by = , where are the orthonormal eigenfunctions and are the coefficients, as described by the Karhunen-Loeve theorem. This is proven to be the exact same thing as running a linear neural network over the velocities to reconstruct them. For the rest of the paper, the linear model in question is taken from the approximations governed by this theorem and is equivalent to an actual neural network.

Near-Wall Turbulent Flow

The neural network used to model near-wall turbulent flow is called a multilayer feed-forward neural network, or MFFN for short6. denotes the output of the layer as follows . Note that is the weight matrix

\*

A picture containing mirror, clock

Description automatically generated

Figure 5: The MFFN 6. Each node has two symbols. stands for the sum of all the previous input data, and the small graph next to it denotes whether that process is linear or not.

There are three kinds of layers, the first being the input layer, which simply takes in the input to the network. Then there’s the hidden layer, which receives output from the previous layer, processes it, and then outputs to the next layer. Finally, there’s the output layer which produces the final output of the network. It can be shown that a NN with at least one hidden layer that uses a nondecreasing, bounded nonlinearity can approximate a continuous function, hence the NN’s application to Navier-Stokes. It should be noted that the solutions to Navier-Stokes are assumed to be bounded (real-valued) and continuous.

The neural network is trained such that the sum of the square of the errors, or SSE, is minimized. The SSE is , where the two components are the real and reconstructed velocities. This kind of training is called the back-propagation method, which is outside of the scope of this paper.

It’s here that I’ll distinguish a linear and non-linear approach to the reconstruction of these flows. This is because both a linear and non-linear NN approach is taken in this attempt to model flow. A NN is called linear if the function is linear. That is to say, for equation

If the linear method is used, then we get that for the flow field

Then the SSE becomes as follows

One other important thing is that

The other method used here is the non-linear method. The non-linear method takes the following form. Note that the added carrot on the flow terms denotes approximation.

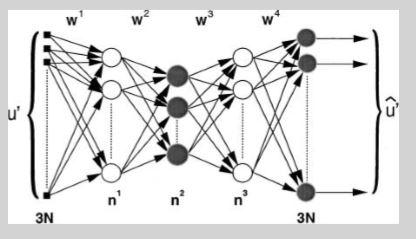


Figure 6: Our specific network for the nonlinear PCA6. The input layer takes in the data, and the output layer produces the final reconstructed velocity. Across the top are the number of weights in that layer, and along the bottom are the number of neurons.

Before reconstruction begins, the model is tested by trying to model the Randomly Forced Burgers Equations. The equations are known to have turbulent-like solutions, as is a benchmark for testing out modeling and the quality of NN’s. Figure 7 depicts the true velocity from the sample vs the reconstructed version. Over 10,00 samples were fed into the simulation.

A close up of a map

Description automatically generated

Figure 7: Reconstructed Velocity comparison. The top graph is the linear case; the bottom is the non-linear.6 The linear relationship in either case shows that the model was successful.

Both reconstruction fared well when compared to the original sample, although there is a notable variance with the linear case, as seen in figure 7. After this is done, the models are now used against Navier-Stokes, using a Reynolds number of 2500 based on the channel height, bulk velocity, and kinematic velocity

A close up of a map

Description automatically generated

Figure 8: Reconstruction Performance on DNS6 . The y component stands for distance from the wall, and it should be noted that these values are time averaged. The top is the overall velocity correlation coefficient, another statistics method. The bottom two are the same, except the velocity is broken down in to its normal and streamwise components. Circles are linear, triangles non-linear

Conclusion

Overall, deep learning methods were very successful in reconstructing turbulent flow. With both the NS and Burger equations, the outputted velocities and the constructions very closely resembled the actual flow. The only notable drawback in the computational power required to accomplish this, since the nonlinear PCA is an iterative process, but as technology continues to evolve, this will become less of an issue.

Sources

1 D. Software, Medium (2017).

2 J. Brownlee, Mach. Learn. Mastery (2015).

# 3Luis B, "An introduction to machine learning and data science for developers" (2018)

4Lawtomated (2019).

5 M.A. Nielsen, (2015).

6 M. Milano and P. Koumoutsakos, J. Comput. Phys. **182**, 1 (2002).