Paper Review
Neural Ordinary Differential Equations(2019)

Motivation

In recent times many complex neural networks with billions of parameters are being implemented for LLMs and Image Generation models. Implementing Large Neural Networks is computationally very expensive because of many reasons. During backpropagation while training a neural network the gradient for each layer is stored and which causes the consumption of large amount of space. In the paper Neural Ordinary Differential Equations the authors have come up with a strategy that can overcome the problem of memory consumption by modelling the neural network as an ODE. This new approach can do backpropagation in O(1) extra space and it reduces FLOPs by a considerable amount.

How does it work?

To understand the idea first one should understand the idea of residual neural network , in a residual neural network the layers of the layer is shorted with the output of the previous layer. It was observed that as the complexity of the neural networks was increased the training loss increased as shown in the paper [Deep Residual Learning for Image recognition]. After shorting the layers with the output of previous layer the training loss did not increase on increasing the depth of the network and it also performed much better than the traditional neural network.

The equation for ResNets is ➖

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}.$$

This equation is similar to solving an ODE using Euler's method. In ResNet this is done discreetly. In neural ODEs we consider a continuous evaluation of the following function. We consider taking smaller steps and form an ODE which can be solved using modern differential equation solvers which provide high accuracy and lower computation cost.

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

This ODE can be solved with given initial value at t=0 and we can evaluate the value of h(T) by solving it using modern differential equation solver which will evaluate the function only at the necessary points and will be more efficient and accurate.

Neural ODEs consumes constant space because it does not store the gradients for each layer rather in neural ODE the backpropagation step can be represented by another ODE and we solve that ODE in reverse time using adjoint sensitivity method. Solving this ODE requires to know the value of the function at previous states, instead of storing them, we recompute those during backpropagation thus consuming very less memory. This method can increase the numerical error as in forward pass there would already be some error and

while solving backward that error might increase, but this problem can be tackled by setting up the tolerance limit during backpropagation to reduce the error.

For solving the backpropagation we define a term adjoint as the partial differentiation of loss with z(t). The differential equation for continuous backpropagation can be derived using basic calculus.
The differential equation comes out to be -

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^{\mathsf{T}} \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}}$$

The vector jacobian products can be computed by using automatic differentiation very efficiently.
We get the values of the necessary parameters by a single call to modern differential equation solver.

Thus we can do the backpropagation using constant extra space using continuous states compared to discrete nets where the memory is proportional to the number of layers.

Further Applications

The idea of continuos states can be applied to places where the path of the problem is important like in time series modelling , Normalizing flows.
A normalizing flow is a variable change method that is used to compute the change in the probability of the output if the variables are transformed through a function. In discrete normalizing flow the change in probability is calculated by this formula:

$$\log p(\mathbf{z}_1) = \log p(\mathbf{z}_0) - \log \left| \det \frac{\partial f}{\partial \mathbf{z}_0} \right|$$

Calculation of the determinant of a matrix is cubic in the dimension of the matrix and hence this operation is computationally very expensive. Normalizing flows are used in image generation to new image from other image. Various people have tried to choose the functions of transformations such that the matrix is diagonal, sparse or triangular so that the computation of the determinant is not expensive, but this limits the functions which can be taken. If we use continuous normalizing flows, we don't need to find the determinant anymore.
We model continuous modelling flow by defining a continuous random variable z(t) with time varying probability p(z(t)). And $dz/dt = f(z(t), t)$ a differential equation with continuous in time transformation of $z(t)$ We also assume that the function f is Lipschitz continuous in z and continuous in t. Lipschitz condition ensures that the initial value problem has a solution. Then the change in probability is given by following equation-

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)}\right)$$

Finding trace of the matrix is computationally much cheaper than finding determinant.

Future Works

This paper is solely based on ODEs, we can try to explore how can we use SDEs in a similar fashion to get information about the uncertainties in the output.
Neural ODEs application in Large Language Models can be explored.

Conclusion

The method of neural ODEs is very powerful and alien idea. This method uses traditional implicit differential equation solvers to model neural nets. It decreases the computational expenses by a significant amount. The numerical error decreases as the number of function evaluations are increased. The method in the paper ‘Neural Ordinary Differential Equations’ models the discrete function evaluations of the neural network by a continuous differential equation. The idea is similar to ResNets, wherein one can view it as Euler method of solving differential equation. Many test results were presented in the paper and the Neural ODE performed similar or better than the traditional neural network models at a much less computational cost !!. This method displays that neural networks can be thought of as differential equations and changes ones perspective of seeing neural networks.

The proof of the equations is done using basic calculus and can be found in the paper.
https://arxiv.org/pdf/1806.07366.pdf