Name : Sushant Sreeram Swamy

Introduction :
Inspired by the assignment, the topic on which I am going to model based on the paper is “Optimization Using Reinforcement Learning”.

Step 1 : Formulate the question:
Deep learning has always faced the problem of the gradient descent algorithm getting stuck at a local minima, plateau or a saddle point. The problem statement which I would like to put forward is : *How well would RL agents trained to optimize, perform compared to traditional optimizers.* The way I plan to evaluate its performance is by monitoring the number of steps the optimizer took to converge to the global minima.

Step 2 : Understanding the State of the Art
There are two papers which I found related to this, Learning to Optimize (Li & Malik, 2016) and Learning to learn by gradient descent by gradient descent (Andrychowicz et al., 2016). The second paper tackled this problem using supervised learning and the first paper used reinforcement learning. The problem with modelling the optimization task as a supervised learning task is that supervised learning assumes that the data it receives is from an independent and identically distributed (*i.i.d*.) examples. But while doing gradient descent, the step which is taken in the current time instance would most likely affect the function in the upcoming iterations. Hence the authors in the first paper modelled this using a reinforcement learning problem.

Step 3 : Determining the Basic Ingredients
Most of the deep learning models use gradient descent as the optimization algorithm. The way in which these models differ is the optimizer that the model uses. To model the optimizer we need a policy function which the reinforcement learning agent will learn. This policy might differ from one objective function to the other. So in order to generalize all functions, what

we can do is to make the policy a function of the gradient. This will subsume all the optimization functions.

For the agent, the inputs will be the current state. Given the current state as the input, the output will be the action to take. This action would represent the amount to update the current state along each dimension.

Step 4 : Formulating specific, mathematically defined hypotheses

For this problem statement, intuitively we know that the rewards which we assign to the agent should be higher when the action the agent takes moves it closer to the global optimum.

Step 5: Selecting the toolkit

One of the best open source toolkits is PyTorch which can be used to generate the model. There are various ways in which we can change the parameters and PyTorch provides a flexible framework. The other toolkit which we can see is PyTorch-Lightning. It is built over PyTorch and provides a way to organize PyTorch code into proper parts. The advantage of PyTorch Lightning is that it gives all the functionalities that PyTorch gives and it also helps to reproduce code easily. The code can be organized in a better way.

So once the network is trained, we can save the model and its weights. After this, given an optimization function, we can directly use this trained model to give the optimization steps in the optimization loop of any deep learning algorithm.

Step 6 : Planning the Model :

For the policy function, since the environment involves continuous action spaces using algorithms such as DDPG would be helpful since it performs better in continuous action spaces.

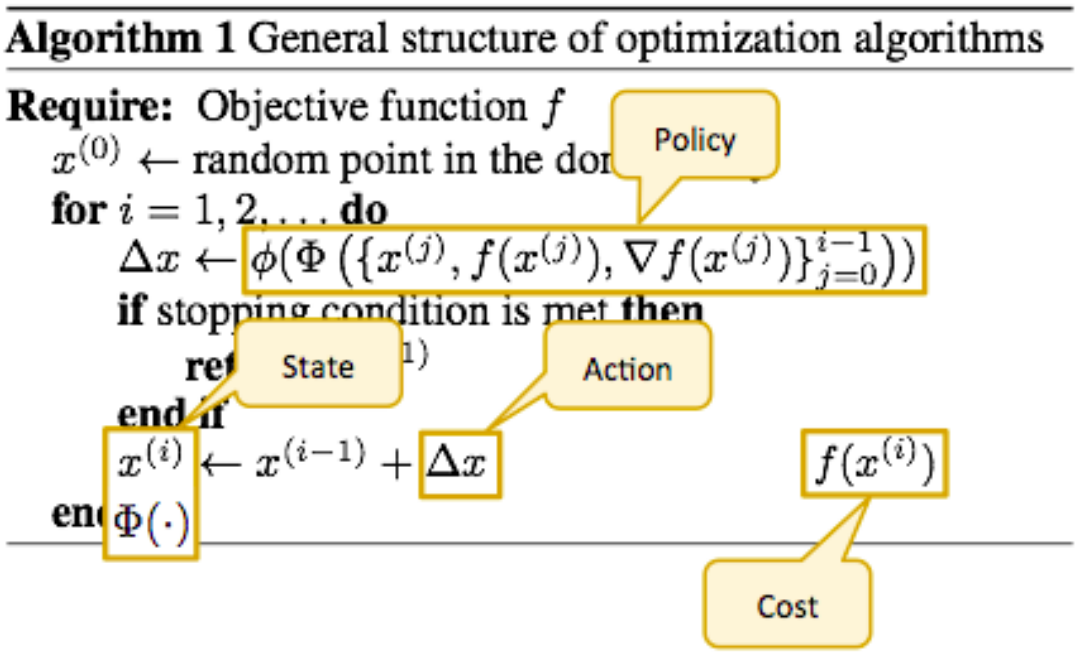

Image taken from this reference

This can be a general structure of the algorithm

Step 9 : Evaluate the model

To test how the model works with different other optimizers, we can monitor the rate of convergence of the optimizers. If the rate of convergence is faster than the traditional algorithms, then we can conclude that our learning based optimizer can do a better job in deep learning tasks.

References :

https://bair.berkeley.edu/blog/2017/09/12/learning-to-optimize-with-rl/
https://arxiv.org/abs/1606.01885