Final Report: Strategic Navigation in Complex Maps

Nikunj Gupta, Raj Talan
{nikunj,talan}@usc.edu

April 22, 2024

1 Problem description

Strategic navigation in complex maps presents several significant challenges. **(Sparse rewards)** where meaningful feedback is rare, making it tough for agents to determine effective strategies. **(High-dimensional state spaces)** which can lead to computational inefficiencies and slow learning processes. **(Delayed rewards)** it can be challenging to determine which actions are responsible for eventual success. **(Exploration vs. exploitation trade-off)** agents must balance the need to explore unknown parts of the environment with the need to exploit known paths to maximize rewards. These challenges require advanced algorithms and robust strategies for effective learning and navigation in diverse scenarios. We implemented an algorithm based on reinforcement learning capable of efficiently (within 2000 episodes of training) and consistently (multiple independent trials/evaluations) navigating all the test maps by making intelligent decisions over substate goals.

2 Subgoal-condtioned reinforcement learning

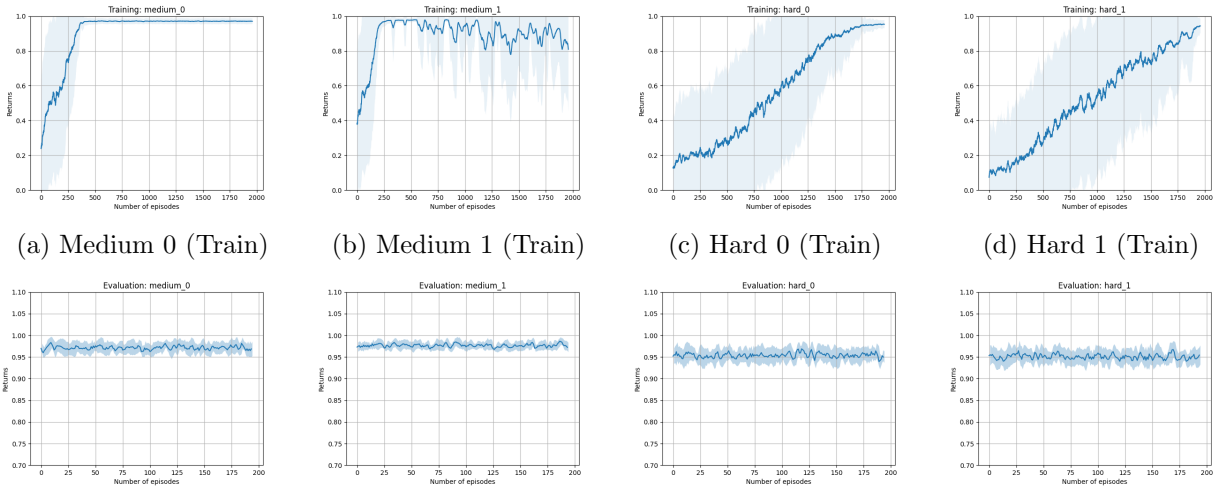
We proposed dividing the environment into sub-states with identified subgoals $S_{sub} = \{s_1, s_2, \ldots, s_n\}$, simplifying the navigation challenge into smaller, manageable tasks. The agent is expected to learn to navigate through these subgoals sequentially towards the final goal S_{goal}. To determine the subgoals, we model the environment as a graph $G = (V, E)$, where nodes V represent the agent, goal, and exit gates, and edges E signify connections between exits across different rooms. We also introduce distance-based edge weights to the graph to accurately reflect the cost or effort required to traverse one node to another. This is particularly useful when determining the agent's shortest or most efficient paths.

Then, we implemented an RL algorithm in which our agent accumulates a bonus for each time it achieves a given subgoal. Specifically, given a list of subgoals $S_{sub} = \{s_1, s_2, \ldots, s_n\}$, for achievement of each subgoal, we accumulate a $r_{subgoal\_bonus}$. We augment these accumulated bonuses to the extrinsic/environmental reward only when the final goal is achieved. In this way, we are in a modified 0-1 reward scheme, where the agent continues to receive a reward of zero when it is not able to achieve its goal; however, receives $r_{aug} = r_{environment} + \beta * r_{subgoal\_bonus}$ only when the final goal is achieved. $r_{environment}$ here equals 1, provided by the environment.

Our approach introduced additional hyperparameters: weight β and subgoal bonus $r_{subgoal\_bonus}$.

- **Extrinsic Bonus Weight** (β): Controls the impact of the extrinsic bonuses on the total reward. We have $\beta = 1$. We tried {0.1, 0.5, 0.01, 0.001, 1}.
- **Subgoal Achievement Bonus** $r_{subgoal\_bonus}$: Determines the intrinsic reward for achieving subgoals. We have $r_{subgoal\_bonus} = 1$. We tried {0.1, 0.3, 0.5, 0.01, 1}.

3 Final results



(a) Medium 0 (Train) (b) Medium 1 (Train) (c) Hard 0 (Train) (d) Hard 1 (Train)

(e) Medium 0 (Eval) (f) Medium 1 (Eval) (g) Hard 0 (Eval) (h) Hard 1 (Eval)

Figure 1: Our results. Training is run for 2000 episodes on 20 independent trials. Evaluation is run for 200 episodes on 5 independent trials. Our approach works on all test maps.

4 Limitations

(1) **Pre-/manual determining of subgoals:** We determine the subgoals for given maps manually. This may be okay for these or similar tasks; however, it is clearly less generalizable to other tasks. Such predetermination may not always be possible. What if our RL agent is also able to determine subgoals? Or is there a way to automatically design subgoals given any state from any environment? Such *dynamic subgoal identification* can help improve this in the future.

(2) **Additional hyperparameters:** We introduce additional hyperparameters ($\{\beta, r_{subgoal\_bonus}\}$) that need to be tuned to optimize the learning process. Tuning these parameters can be tedious and computationally expensive. Using *hyperparameter optimization tools* such as Hyperopt, Optuna, and Ray Tune can help automate the hyperparameter tuning process.

Intermediate Report 2: Strategic Navigation in Complex Maps

Nikunj Gupta, Raj Talan
{nikunj,talan}@usc.edu

April 9, 2024

1 Problem description

We realized that hyperparameter tuning seems absolutely necessary from the other group presentations from Intermediate Phase 1. So, in this phase, we decided to first work on "*Model Selection and Fine-Tuning*" before trying out more complicated ideas such as "*Rewarding by punishments*" or "*Dynamic subgoal identification*" mentioned earlier in our plan. To reiterate, we aim to solve at least one of the hard maps, using our previously discussed approach of subgoal-conditioned reinforcement learning.

2 Proposed solution: Subgoal-condtioned RL

Previously, we combined subgoal-conditioned learning with an incentive system for learning the given subgoals. For this, we provided intrinsic rewards for each time our agent achieved a subgoal from the predetermined set of subgoals (using depth-first search on a distance-based edge-weighted graph of exit gates from agent cell to goal cell), represented as $S_{sub} = \{s_1, s_2, \ldots, s_n\}$. This approach ensures that the agent's exploratory actions are guided not only by the immediate environment but also by the strategic achievement of these subgoals, enabling a more structured exploration pattern. We also note here that we applied these subgoal rewards as "extrinsic bonuses", unlike the stereotypical intrinsic reward system where bonuses are immediately applied upon each subgoal achievement. In our case, the extrinsic bonus is accumulated through the episode and applied in a singular instance when the final goal, S_{goal}, is reached (hence, $r_{aug} = r_{environment} + \beta * r_{subgoal\_bonus}$). This reinforces the optimal path to the final goal that encompasses our strategic subgoals.

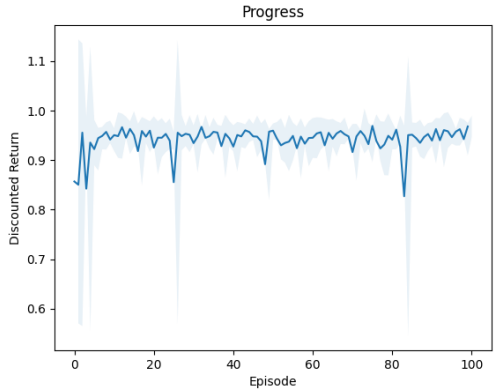
As realized from the previous group meeting, hyperparameter tuning seems to play a pivotal role in refining our reinforcement learning models, especially when utilizing subgoal-conditioned learning and extrinsic bonuses (which introduce additional hyperparameters, for instance, the weighting factor β for the bonuses). The primary hyperparameters that required readjustment include:

- **Exploration Rate** and **Exploration Decay** in ϵ-greedy strategy: For both the hard maps, we experimented with initial $\epsilon = 1$ and decay of 0.0001 and 0.00001. The gradual decay ensures that the agent initially explores a wide range of actions and progressively focuses more on exploiting the learned policy as it becomes more confident in its decisions.
- **Learning Rate** (α): We have $\alpha = 0.1$ now. A smaller learning rate requires more training epochs but can lead to better convergence, whereas a larger learning rate can expedite convergence but may overshoot the minimum.
- **Extrinsic Bonus Weight** (β): Controls the impact of the extrinsic bonuses on the total reward. Required tuning to ensure that the extrinsic rewards effectively motivate the agent towards achieving subgoals without overshadowing the primary task's rewards. We have $\beta = 1$ now. We tried {0.1, 0.5, 0.01, 0.001, 1}.

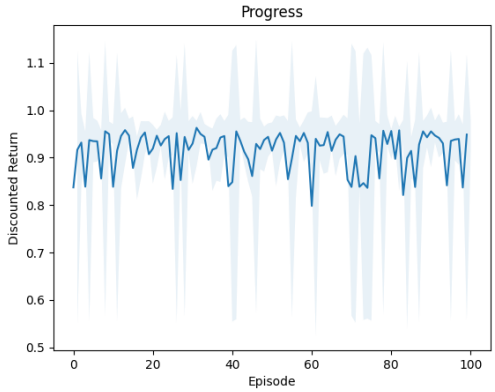
- **Subgoal Achievement Bonus** $r_{subgoal\_bonus}$: Determines the intrinsic reward for achieving subgoals. Must be balanced to ensure subgoals are enticing but not so rewarding that they detract from the final goal. We have $r_{subgoal\_bonus} = 1$ now. We tried {0.1, 0.3, 0.5, 0.01, 1}.
- **Model selection**: Additionally, the choice of the RL algorithm itself played a critical role in the algorithm's success. Our current model learns using Q-learning. We implemented and experimented with {Q-learning, SARSA, TD(0) Learning, TD(1) Learning}.

Tuning these hyperparameters involved iterative experimentation by random (intuitive) search (experiments/evaluations from other groups helped make this faster). The goal was to find a set of hyperparameters that maximizes the efficiency and effectiveness of the agent in navigating hard maps toward the final goal, taking into account the structured subgoals using extrinsic bonuses.

3 Results



(a) Hard 0 (Eval)



(b) Hard 1 (Eval)

Figure 1: We declare success on *Hard 0* map. Our results are decent on *Hard 1* map (standard deviation seems higher). We evaluate for 100 episodes on 10 independent trials (random seeds).

4 Plan

Week	**Activity**	**Milestone/Outcome**	**Responsibility**
Done	Setup Implementation	Setup Code Base	Nikunj, Raj
Done	Algorithm Implementation	Algorithm Code Base	Nikunj, Raj
Done	Initial Evaluation	Positive Results on Easy Maps	Nikunj, Raj
Done	Designing heuristic weight function (e.g., distance) for edge weights	Unweighted to weighted subgoal graph	Raj
Done	Testing new exploration bonuses	Enhanced exploration bonuses	Nikunj
Done	Integration of Approach 2.1 and 2.2	Merged single approach	Nikunj, Raj
Postponed	Rewarding by punishments	Testing with penalties when agent does not do as expected	Nikunj
Postponed	Training on local rooms	Solving subgoals independently first then using them for original map	Nikunj, Raj
Postponed	Dynamic subgoal identification	Heuristic to Learnable Subgoals	Nikunj, Raj
Done	Model Selection and Fine-Tuning	Optimal models/hyperparameters	Nikunj, Raj
Week 13	Preparing for final submission	Final documentation, cleaning of code, final evaluations	Nikunj, Raj

Table 1: Activities and their corresponding milestones or outcomes in the development process.

Intermediate Report 1: Strategic Navigation in Complex Maps

Nikunj Gupta, Raj Talan
{nikunj,talan}@usc.edu

March 28, 2024

1 Problem description

The major challenges that we are addressing to solve the problem of strategic navigation in complex maps are **high state dimensionality** and **reward sparsity**. For this, we came up with ideas under subgoal-conditioned reinforcement learning, where we want to find appropriate, numerous subgoals for our agent to recognize, learn to solve, and then combine the smaller solutions in a way to solve the original problem (reaching the final goal). As we progress in developing such an algorithm based on reinforcement learning, we realize that the hard maps are a better testbed for its evaluation. There are multiple rooms in the hard maps, and their exits can become tangible subgoals for our agent to recognize and learn.

2 Proposed solution: Subgoal-condtioned RL

We proposed two strategies to enhance reinforcement learning in complex navigation tasks. The first employed subgoal-conditioned learning, where the environment is divided into sub-states with identified subgoals $S_{sub} = \{s_1, s_2, \ldots, s_n\}$, simplifying the navigation challenge into smaller, manageable tasks. The agent was expected to learn to navigate through these subgoals sequentially towards the final goal S_{goal}. The second strategy introduced exploration bonuses, $r_{aug} = r_{environment} + \beta * r_{bonus}$, to mitigate reward sparsity by rewarding the agent for exploring novel states, thereby encouraging comprehensive environment exploration.

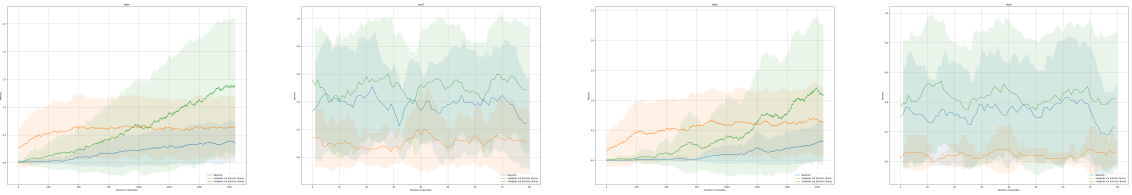
To determine the subgoals, we model the environment as a graph $G = (V, E)$, where nodes V represent the agent, goal, and exit gates, and edges E signify connections between exits across different rooms. In enhancing our graphical representation of the environment, we introduce distance-based edge weights to the graph. By assigning such weights to the edges, we can more accurately reflect the cost or effort required to traverse from one node to another. This is particularly useful when determining the shortest or most efficient paths for the agent to take.

Moreover, with the motivation of merging ideas from the aforementioned two approaches, we have implemented an RL algorithm in which our agent is given a bonus (intrinsic) reward for each time it achieves a given subgoal. This way, we have aligned its intrinsic motivation to explore with subgoal-finding. Specifically, given a list of subgoals $S_{sub} = \{s_1, s_2, \ldots, s_n\}$, for achievement of each subgoal, we accumulate a $r_{subgoal\_bonus}$.

Now, we try two different ways of using this accumulated $r_{subgoal\_bonus}$ for reinforcement learning. One way is to add a fixed bonus (weighted), $r_{subgoal\_bonus}$, to the $r_{environment}$ using $r_{aug} = r_{environment} + \beta * r_{subgoal\_bonus}$ as soon as a subgoal is achieved; we call this "**intrinsic bonus**". In this way, we address the fact that whenever the agent passes through a necessary subgoal, the agent is rewarded to let it know that it is following an optimal path. So in this approach, if it passes through 3 sub-goals, then for each subgoal attainment, $r_{subgoal\_bonus}$ gets added three times. However, this way of reward interference is not empirically working well for us (results in the next

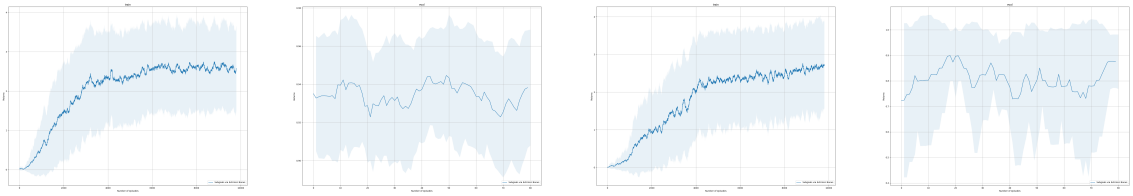
section). So, we try another way in which we augment the accumulated bonuses for achieving sub-goals to the extrinsic reward only when the final goal was achieved; we call this "**extrinsic bonus**". In this way, we are in a modified 0-1 reward scheme, where the agent continues to receive a reward of zero when it is not able to achieve its goal; however, receives $r_{aug} = r_{environment} + \beta * r_{subgoal\_bonus}$ only when the final goal is achieved. $r_{environment}$ here equals 1, provided by the environment. So, in this case, if the agent reaches the goal while passing through 3 sub-goals during its learning, taking an optimal path, then the final reward for the agent becomes an environmental reward plus three times the extrinsic bonus. This gives us better empirical results.

3 Results



(a) Hard 0 (Train) (b) Hard 0 (Eval) (c) Hard 1 (Train) (d) Hard 1 (Eval)

Figure 1: Current status of our results: Blue: Baseline (SARSA with UCB1 exploration), Orange: Subgoal-conditioned RL via intrinsic bonus, Green: Subgoal-conditioned RL via extrinsic bonus.



(a) Hard 0 (Train) (b) Hard 0 (Eval) (c) Hard 1 (Train) (d) Hard 1 (Eval)

Figure 2: Feasibility Study: Results with longer training (10K episodes)

4 Plan

Week	**Activity**	**Milestone/Outcome**	**Responsibility**
Done	Setup Implementation	Setup Code Base	Nikunj, Raj
Done	Algorithm Implementation	Algorithm Code Base	Nikunj, Raj
Done	Initial Evaluation	Positive Results on Easy Maps	Nikunj, Raj
Done	Designing heuristic weight function (e.g., distance) for edge weights	Unweighted to weighted subgoal graph	Raj
Done	Testing new exploration bonuses	Enhanced exploration bonuses	Nikunj
Done	Integration of Approach 2.1 and 2.2	Merged single approach	Nikunj, Raj
Week 12	Rewarding by punishments	Testing with penalties when agent does not do as expected	Nikunj
Week 12	Training on local rooms	Solving subgoals independently first then using them for original map	Nikunj, Raj
Week 13	Dynamic subgoal identification	Heuristic to Learnable Subgoals	Nikunj, Raj
Week 13	Model Selection and Fine-Tuning	Optimal models/hyperparameters	Nikunj, Raj

Table 1: Activities and their corresponding milestones or outcomes in the development process.

Proposal: Strategic Navigation in Complex Maps

Nikunj Gupta, Raj Talan
{nikunj,talan}@usc.edu

March 20, 2024

1 Problem description

Strategic navigation in complex environments presents several challenges that algorithms must overcome to succeed. In given maps, rewards are few and far between, making it challenging to understand which actions lead to positive outcomes (**reward sparsity**). Complexity increases in medium and hard maps as the number of possible states exponentially grows. **Exploration** is daunting under these circumstances. We aim to develop an algorithm based on reinforcement learning capable of efficiently and consistently navigating at least two medium maps by making intelligent decisions over **substate goals** and learning from its environment to maximize **motivational** rewards over multiple attempts.

2 Proposed solutions

2.1 Approach 1: Subgoal-conditioned reinforcement learning

We propose a subgoal-based method for automatically creating useful skills in reinforcement learning. Our method identifies subgoals by partitioning the states into substates, and then the agent learns to navigate within each substate using reinforcement learning. Through this, we aim to decompose the given complex navigation problem into multiple simpler subproblems, solve them in turn, and compose their solutions to navigate the entire map. We construct a graphical representation of the environment, $G = (V, E)$ where V represents the set of nodes corresponding to the positions of agent, goal, and all the exit gates in the state/map, and E represents the set of edges denoting connections between the exit gates which connect different rooms. The agent and goal node get connected to the corresponding room's exit nodes. From this graph, we extract the shortest path of exits from the agent's initial position S_{init} to the goal S_{goal}. These 'exits' identify as the sub-goals $S_{sub} = \{s_1, s_2, \ldots, s_n\}$ for our agent where n is the number of sub-goals, and the last subgoal is the final goal of the map. We then train an RL agent to sequentially learn to achieve these subgoals.

2.2 Approach 2: Exploration-based bonuses

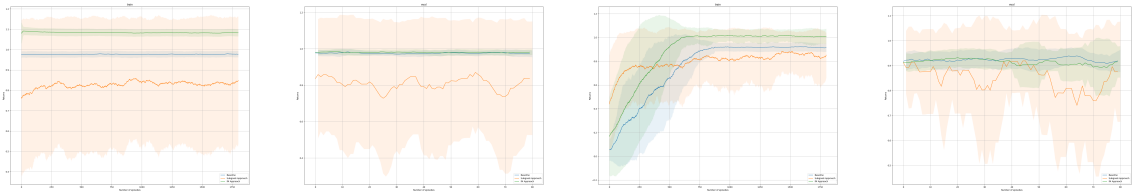
Alternatively, we also test an approach to address reward sparsity in the given maps. We propose "exploration bonuses" as additional rewards given to the agent for taking actions that lead to less-visited or novel states or actions within the environment to incentivize exploration of the environment more thoroughly rather than solely focusing on exploiting known paths to immediate rewards. We translate the concept of such exploration-based bonus, r_{bonus}, to RL by combining it with the extrinsic reward provided by the environment

$$r_{aug} = r_{environment} + \beta\ r_{bonus}$$

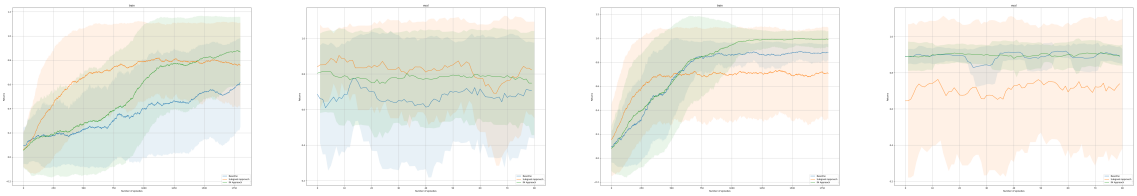
through a weighting factor β. We begin by adding two types of bonus rewards: (1) incentivizing visits to as many distinct states as possible in one episode using $\frac{N_{\text{distinct}}}{N_{\text{total}}}$ (2) Modeling long-term

exploration behavior using $1/\sqrt{\mathcal{N}(s)}$. We note here that these bonuses are given to the agent **only** when it successfully reaches the goal. In other words, the reward is 0 when the goal is not reached and r_{aug} when the goal is achieved.

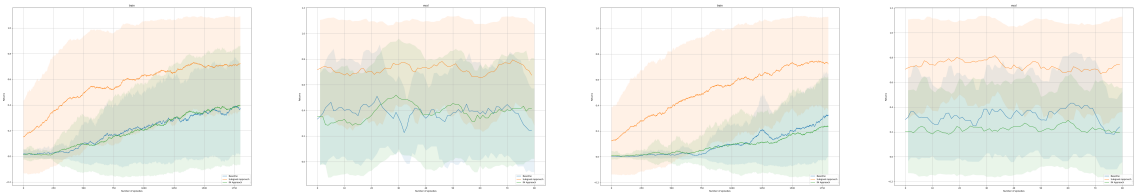
3 Feasibility study



(a) Easy 0 (Train) (b) Easy 0 (Eval) (c) Easy 1 (Train) (d) Easy 1 (Eval)



(e) Medium 0 (Train) (f) Medium 0 (Eval) (g) Medium 1 (Train) (h) Medium 1 (Eval)



(i) Hard 0 (Train) (j) Hard 0 (Eval) (k) Hard 1 (Train) (l) Hard 1 (Eval)

Figure 1: Potential feasibility of both approaches. Blue: Baseline (SARSA agent with UCB1 exploration), Orange: Subgoal-conditioned RL, Green: Exploration-based bonuses).

4 Plan

Week	**Activity**	**Milestone/Outcome**	**Responsibility**
Done	Setup Implementation	Setup Code Base	Nikunj, Raj
Done	Algorithm Implementation	Algorithm Code Base	Nikunj, Raj
Done	Initial Evaluation	Positive Results on Easy Maps	Nikunj, Raj
Week 11	Designing heuristic weight function (e.g., distance) for edge weights	Unweighted to weighted subgoal graph	Raj
Week 11	Designing/using new exploration bonuses	Enhanced exploration bonuses	Nikunj
Week 12	Dynamic subgoal identification	Heuristic to Learnable Subgoals	Nikunj, Raj
Week 12	Integration of Approach 2.1 and 2.2	Merged single approach	Nikunj, Raj
Week 13	Selection of optimal base model and hyperparameters	Model Selection and Fine-Tuning	Nikunj, Raj

Table 1: Activities and their corresponding milestones or outcomes in the development process.