QMIX Based 3v3 Snake Cooperative Agents

Yu Zheng
Shanghai Jiao Tong University
zhengyu0730@sjtu.edu.cn

Abstract

By describing the design and experiments of the cooperative agents using QMIX[2] algorithm in the `3v3 Snake` environment, this paper discusses four main topics about reward shaping, global state encoding, agent network structure design and opponents.

1 Introduction

Multi-agents reinforcement learning (**MARL**) is a widely researched problem. Unlike the traditional reinforcement learning, it more emphasizes the balance of cooperation and competition, which make the learning of great policies more difficult.
In this paper, based on QMIX[2] algorithm, a system is designed to control three "snake" agent to cooperate and compete in a MARL environment named "**3v3 Snake**". To study the main problem of MARL and thus gain a reasonable performance on the environment, some experiments have been designed to research on these main topics:

- reward shaping problem
- state feature embedding
- the opponent selection problem when training
- the agent network structure choice

In Methodology, the basic MARL problem definition, opponent selection problem, the main idea of reward shaping problem,state embedding will be discussed. In Related Work, **QMIX**[2] algorithm will be briefly introduced. In Experiments, the more detail of the experiment design related to the main topic above will be given in addition to the result of my experiments with the discussion towards them.

2 Methodology

2.1 Dec-POMDP

Dec-POMDP[1] (Decentralized Partially Observable Markov Decision Processes) can be described with the following elements and each plays an important role in the model and experiment design.
First is the global state $s \in S$. Global state is usually unseen to the agents, which means what the agents see is partial of it, or to say, a subspace.
Secondly, agent number, $a \in A = \{1, \ldots, n\}$, the number is unique to each agent, which could be used to identify multi agents in the network design.
Thirdly, the action in Dec-POMDP is divided into two main part. One is independent action $u^a \in U$ and the other is the combination of the former, joint action $\mathbf{u} = (u^1 \ldots \ u^n)^T \in \mathbf{U}$. As you can see, the search space of joint action $\mathbf{u}$ is exponentially increasing when more agents are added. So decentralized greedy policy has been widely used in MARL algorithm including QMIX[2], which

make the search space maintain in $O(|U|)$ but not $O(|U|^n)$.
Fourthly, State transition probability $P(s'|s, \mathbf{u})$, the transition of state based on the joint action.
Fifthly, reward $r(s, \mathbf{u})$, in this paper, the reward is gained by current state and the joint action, so we will use total Q value to fit it.
Most importantly, **observation function** $z \in Z, z = O(s, a)$, it's the main difference between the Dec-POMDP and MDP for single agent reinforcement learning. The observation is identified by the number of agent a and each agent gain a different observation.
The next is the observation-action history $\tau^a \in \mathbf{T} \equiv (Z \times U)^*$, which means the accumulation of the history including a combination of which observation the agent has gotten and the corresponding action the agent has taken each step within an episode. To preserve this is due to that to each agent, the policy can not only depend on the current observation because it's partial of the global state. Take the full history of the action and observation will partly reconstruct the global state and thus provide more information for the agent to determine. In practice, we often use RNN to model the history and note that if the agent can directly observe the global state (in some MARL environment), we can just use current observation like MLP to simplify the model.
Lastly, the decentralized policy $\pi^a(u^a|\tau^a)$, each agent based on observation-action history to determine current independent action.

2.2 Opponent Selection

In Dec-POMDP, the opponent policy is not defined. In this paper, the opponent policy is regarded as a part of the environment. In that case, there is a problem that if using random policy when training then test on another policy, so the performance is not promising. It can be indicated as the generalization error just like the supervised learning and the training and testing dataset are not I.I.D if using Dec-POMDP to modeling a competitive MARL problem. Several experiments including using random, greedy and using self as opponent are designed to measure the generalization error using QMIX, which is based on Dec-POMDP model, in "3v3 Snake" environment. More detail about the three opponents can be found in A.2.1.

2.3 Reward Shaping

Reward, which greatly affects the learning efficiency, should be well designed. Although the environment usually provides a reward, and it's exactly the direct metric to examine the trained agent model, but add extra manual or even learnable reward function is still meaningful because it will provide more heuristic knowledge. A simple but useful method to shape reward is divide the reward by several semantics, each of them representing a desired task for agent to finish, and the total reward is exactly their weighted sum. In this work, the reward is divided into three parts, sparse reward, gain reward and distance reward. More detail of these reward will be provided in the Experiments part.

2.4 State Embedding

In this work, the origin global state isn't with a fixed length. The length of snakes varies at different steps, but the position of snake segments are significant feature. In that case, maybe the whole grid map is a feature that contains entire global state information. But such a feature is sparse so an embedding method is required to embed the grid map to a fixed length feature vector. By the way, note that in this work observation function is not embedded due to the reason that embedding requires a learning procedure, which may not work immediately, the agent directly interacts with the environment so a heuristic fixed observation function may make the training more stable, and the policy of cooperation which may need the whole snake segments information will be learnt by the mixing function with the embedding global state.

3 Related Work

QMIX[2] is a classic MARL algorithm. Its main idea is to using a hypernetwork whose parameters is generated by the global state to model a map from each agent's Q value to total Q value. Use absolute function as the activation function to generated weight of hypernetwork to make the mixing function a monotonic increasing function, which make the joint action generated by combination of the independent action determined by decentralized greedy policy, which is to maximize the agent Q

value, will also be a globally greedy policy,that is, make the total Q value the maximum.

$$\frac{\partial Q_{tot}}{\partial Q^a} \geq 0 \quad \forall a \in A \tag{1}$$

$$\arg\max_{\mathbf{u}} Q_{tot} = [\arg\max_{u^a} Q^a]_a \tag{2}$$

To model action-observation history, QMIX uses GRU as the structure of agent network. In "3v3 Snake" environment, the global state is shared among the agent, so MLP can be also be applied to ease the structure. But the design of observation function will map the global state to a more specific subspace so as mentioned in 2.1, so GRU is also used to compare the performance with MLP.

4 Experiments

4.1 Experiment Setup

4.1.1 Network Structure Setup

The network structure using GRU is just as the network structure described in QMIX[2] and the only difference of MLP structure is just the GRU hidden layer is replaced by a normal FC layer with the same hidden layer dimension as GRU. The optimizer is also RMSprop with learning rate 5×10^{-4}. And DQN train strategy is also applied. Replay Buffer contain the history of each episode with fixed length 200 in GRU model while in MLP model it will only contain step-level history. The size of replay buffer in GRU is 5000 and in MLP the size is chosen by experiments.
Target network is also applied. In GRU, the target network is hard updated per 200 episodes, while in MLP soft update method is used, which softly updates the target network parameters by a weighted sum of the old target network parameters and the training network parameters. The weight τ of training network parameters is set as 0.001 in this work.

4.1.2 State and Observation Setup

The observation is a vector combination of four part. The first part is the position (x, y) of the controlled snake head.The second is the (x, y) position of of the 5 generated beans. The third part is the 4 object surround the controlled snake head, the value is the identification number[1] of the object.The fourth is the position (x, y) of the other 5 snakes head. So the dimension of observation is $26 + 3$, where the extra 3 dimension is the one-hot encoding of the controlled due to that the agent network is shared among the three controlled agents.
Just as mentioned above, the global state is a fixed length feature vector of 22 dimension which is the embedding of the whole grid map, the size of the map is 10×20. The encoding mode 1 is one-hot conv. The input state map is $10 \times 20 \times 8$ where the identification number is encoded as 8-dimension one-hot vector. The encoding mode 2 input 10×20 state map the value in each position is exactly the identification number. Mode 1,2 use convolution layer to encode the input state map, exact layer description will be provided in the A.2.2. The encoding mode 3 is to flatten the state map in mode 2 and using FC layer to encode. Mode 4 ,which is not a encoding method to compare with the encoding methods, is a manual state feature vector, which is just as the observation, a combination of 6 snake head position, 5 beans position. The dimension is exactly 22 to control the variable.
Note that all the position and object identification number are Min-Max normalized to avoid the influence of different unit scales among features.

4.1.3 Reward Setup

As mentioned above, reward in this work has been divided into three types. Sparse reward is not exact the sparse reward in GoalRL.The sparse reward there is related to the semantic of win or lose and each step the value of it will be judged by whether the sum of lengths of controlled snakes is more than that of the opponent. And the last step sparse reward will be double of the normal step. The reward of lose is negative and the absolution of it will be less than that of win. The value of sparse reward is exactly its weight. The gain reward is exactly the environment reward which is used to evaluate the model. It is the sum of variety of each snake's length every step. And the distance reward is designed to push the snake to eat beans, which is designed as the negative of the minimum

[1] 0 is none, 1 is bean, 2-7 are corresponding snake agents

of the Manhattan distance between controlled snake head and all the beans. The reason why using Manhattan distance rather than euclidean distance is that Manhattan distance exactly measure the remaining step the snake need to take to eat the closest bean.

4.1.4 Other Setup

During the training, use epsilon greedy policy and ϵ delays from 1 to 0.05 after 50000 steps. γ is set as 0.99. The training batch size is 32.

4.2 Experiments Results and Discussion

4.2.1 Reward Shaping Result

The method of reward shaping is to change the weights of different reward types. If the change gain a fine improvement of performance it will be reserved. So four experiment has been carried out. The result is presented in 1a. As is indicated in the figure, increasing the weight of gain reward, which is exactly the raw reward of the environment, will greatly improve the performance and increasing sparse reward also positively affect the environment reward. Increment or decrement of distance reward will negatively affect the performance so just keep the weight as 1. According to the result of these experiment, the reward has been set as the following.
Win reward is 15 and the last step is 30, while the lose reward is 10 and the last step is 20. The weight of gain reward has been set as 20 and the weight of distance reward is 1.

4.2.2 Agent Structure Experiment Result

As shown in 2, the GRU structure converges slower than the MLP structure and with a better performance than MLP even not converged, which may due to the RNN structure will model the action-observation history which contain more information than only the current observation, when using the fixed length observation. And 1c indicates that in MLP structure, increment of the size of replay buffer is benefit to the agent performance. So the replay buffer of MLP structure has been set as $1,000,000$, which is exactly the size of the GRU structure. 5000×200.

4.2.3 Encoding Method Ablation Experiment Result[2]

As shown in 2a and 2b, the three encoding methods can be better than no embedding mode. Especially in GRU structure with greedy opponent, the no embedding mode doesn't work while the two convolution modes work well.

4.2.4 Opponent Selection Result

As shown in 3a, compare to the normal mode without `self compete`, the method using `self compete` makes a better performance on the test opponent at first but will be worse than the normal mode with more training episodes while the training performance not decreasing. This may result from the overfitting on the competition with itself. As for the greedy and random opponent training, figure 2 indicates that the generalization performance of greedy opponent will be better than the random opponent which may be described as "Stronger enemy will make you stronger" in the common sense. The reason may include that the strong opponent will force the agent facing more trajectories which will lead to a better reward.

5 Conclusion and Discussion

The experiment result fully cover the four topics this paper intending to discuss, which could indicate the follow conclusion and some perspective for further study:

- The reward shaping is important and adding extra manual reward should be based on giving high-weighted to the environment reward.

[2]Considering the generalization error, the following ablation experiments are tested on ddpg16000, which is trained on independent ddpg algorithm 16000 episodes, opponent using Q value greedy policy

- The RNN network modeling the agent history may take more time to train but in MARL environment it could improve the final performance of agents than MLP.
- State embedding method works, which can be used in other environment when the global state can not easily be described by predefined function.
- Dec-POMDP doesn't model the opponent character, which makes the algorithm based on this theory has a generalization error when testing, and the performance varies when training and testing on different opponents. This is a problem remaining to solve. To release the generalization error, zero-shot or few-shot online learning method could be applied when transferred to a new opponent. Another solution is that the opponent should be considered in the MDP model, and a new reward modeling the competition relationship just like Nash equilibrium should be designed.

Acknowledgments

The code is base on https://github.com/jidiai/Competition_3v3snakes and this work's code including qmix training opens source on https://github.com/COMoER/Competition_3v3snakes.

Thanks for the guidance of **Prof. Weinan Zhang** and **TA Zhengbang Zhu**

References

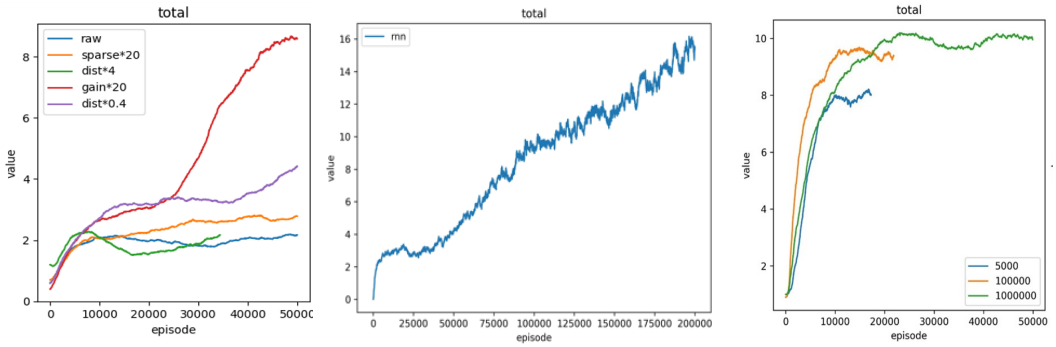
[1] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

[2] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, 2018.

A Appendix

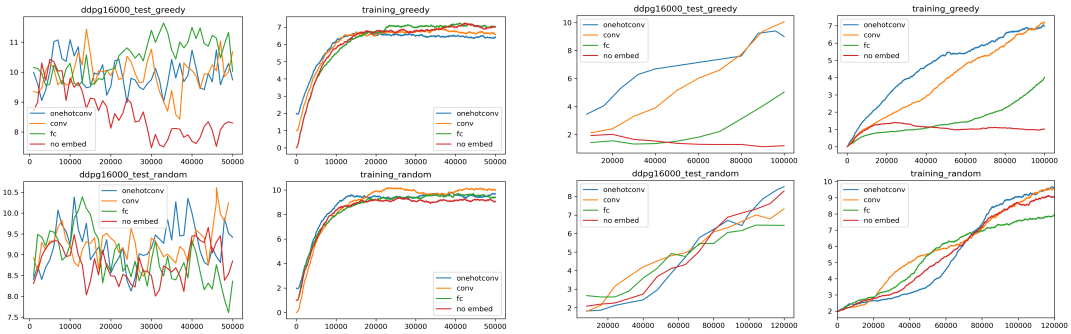
A.1 Experiments Plot

The experiments plot including the figure mentioned in the main content are as following.



(a) reward shaping (b) a training sample of GRU (c) a training sample of MLP

Figure 1: (a) Each is trained on random opponent and GRU structure. The `raw` is with all weight one. `sparse*20` is only raise the weight of sparse reward compared to `raw`. `dist*4` and `gain*20` is based on `sparse*20` setting. `dist*0.4` is based on `gain*20` setting (b) A training sample of GRU structure with random opponent (c) A training sample of MLP and each label of the curves is the size of the replay buffer



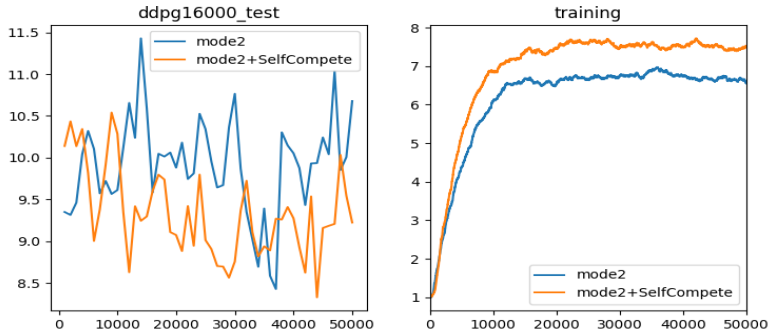
(a) MLP structure (b) GRU structure

Figure 2: The left in the two figures is the test result of the environment reward and the right is the train result.The up of the figures is the result of using greedy opponent while the down is the result of using random opponent

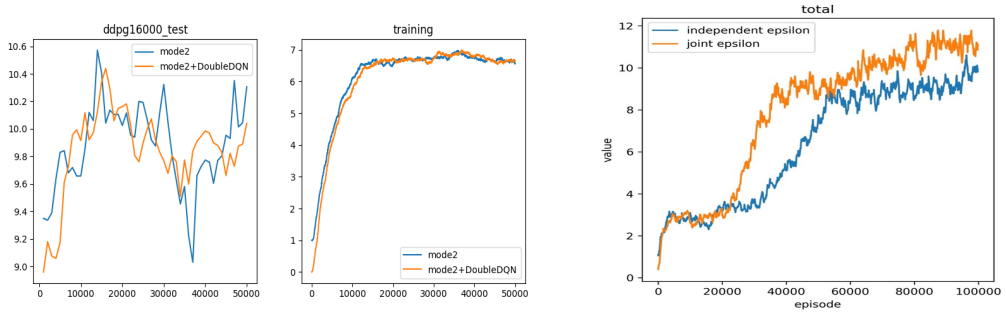
A.1.1 Additional Experiment

As shown in 3b, the ablation experiment between DoubleDQN structure and the normal DQN struture has also been carried out. The result indicated that DoubleDQN will not improve the agent performance and even slow the speed of trainning so in this work DoubleDQN structure is not used in other experiment.

Figure 3c show the result of experiment related to using joint or independent greedy epsilon. When using independent epsilon greedy , each controlled agent will independently judge random policy or greedy policy while when using joint epsilon greedy, the controlled agents jointly judge greedy or random. Independent judging the policy will make the joint action can hardly be the best because one part agents using greedy and the other using random. And the result of experiment also indicates that joint epsilon greedy is better.



(a) Self_Compete



(b) DoubleDQN (c) Epsilon greedy

Figure 3: (a)An ablation experiment between the opponent using greedy and self_compete(b)An additional experiment focusing on the performance using DoubleDQN to compute loss, using MLP and greedy opponent (c)An additional experiment to confirm joint greedy policy is more reasonable, using GRU and random opponent

A.1.2 More Detail about Training

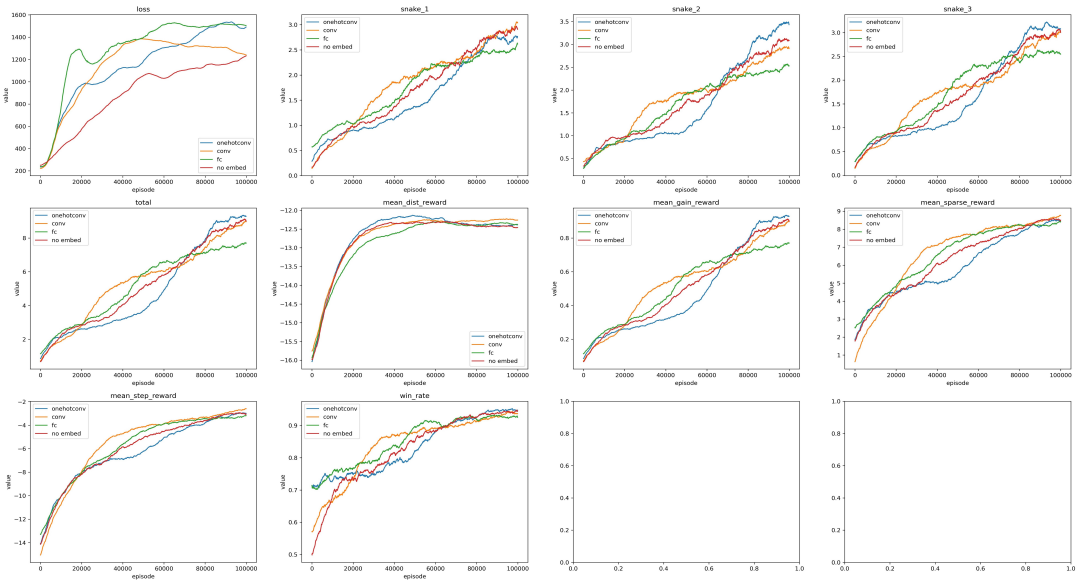


Figure 4: GRU structure Training Detail

The detail about the variety about training loss, each controlled snake environment reward, the total environment reward, each type of reward, the total step reward and the win rate are shown in 5 and 4. The experiment is the ablation among different encoding method in MLP and GRU structure with random opponent. The graph could indicate that

- In MLP structure, the loss will increase to a value and keep value when converged. This may result from the balance of exploring and exploiting. Especially the loss curve of mode 2, the reward decreased when the loss decreased with the decrement of distance reward, and then it came back to converged state when the loss was steady. In the case of GRU structure, because the agent has not been converged, the loss varieties present different modes. But one can be confirmed that the increment of loss indicates the increment of model performance.
- As for the three designed rewards, in the GRU structure when the reward hasn't converged, the distance reward keeps increasing to a value and then varies slightly while the other two rewards keep increasing. This indicated that the agents will adjust its policy to eat beans at the beginning. And in MLP structure, the exploring of mode 2 makes the decrement of distance reward so its reward decreases. This indicates the distance reward also contributes to the environment reward.

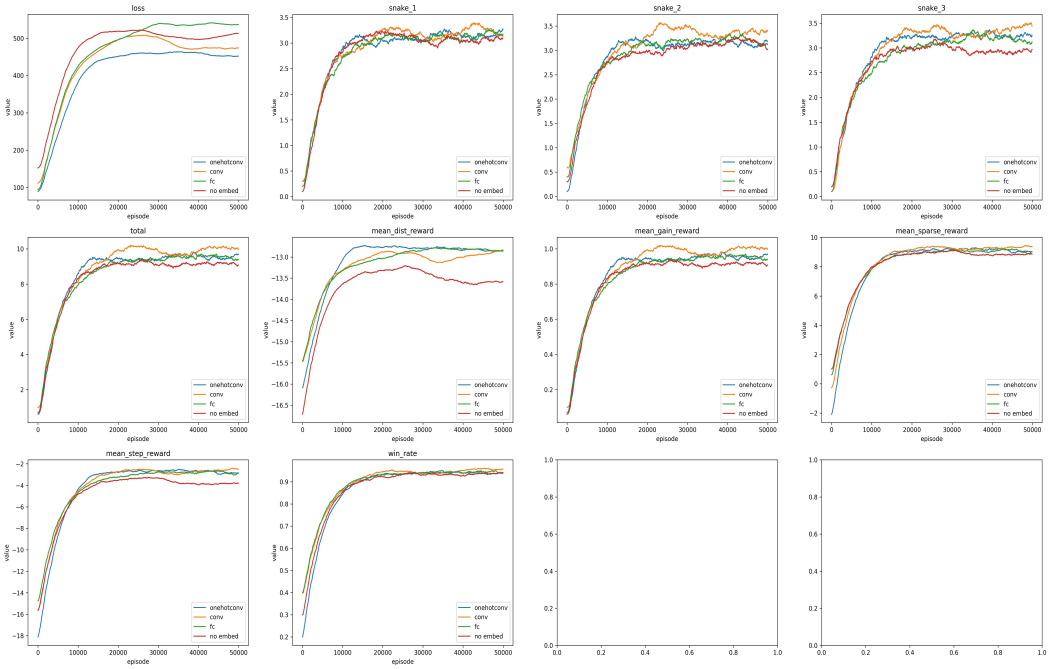


Figure 5: MLP structure Training Detail

A.2 Experiment Detail

A.2.1 Opponent Detail

The detail about each opponent are as following:

- greedy: The greedy policy is naive just find the bean which has the minimal distance towards controlled snake head and then select the best direction which will make the minimal distance from the next step's snake head towards the bean. Of course, when the direction towards other snake's segments it will not be selected.
- random: Due to the bug in baseline code, the observation of origin designed greedy policy are not updated per step and is always the first step's state within an episode, so the heuristic knowledge in greedy policy doesn't work so it's regarded as a random policy in this work, the random policy is just refer to the greedy policy with the observation not updated.

- self_compete: The self compete method is just to use the same network as the opponent. And the trajectory (s, a, r, s', a') in the perspective of opponent will also be keep in the replay buffer.

A.2.2 Encoding Network Detail

The encoding network structure can be found in 6, and the network hyper parameters are as following. The convolution layer of mode1, layer_in channel is 8 and out is 10. The kernel size is 3×3 with no padding. The coluvolution layer of mode2, layer_in channel is 1 and out is 4 , other is the same as mode1. Then the maxpooling layer, kernel size is 2×2 and with no padding and stride 2. The in_channel size of FC layer is corresponding to the input tensor and out_channel is always 22.

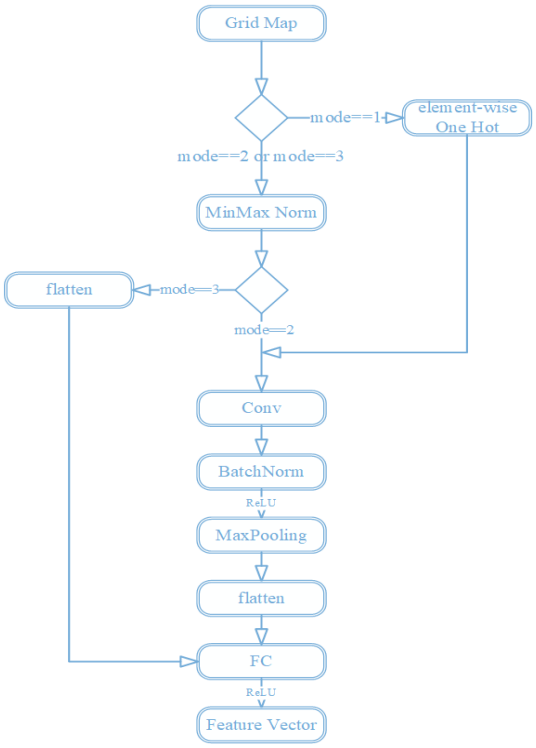


Figure 6: Encoding Network Structure