Game (Pygame)
- Play_step (action)
  → reward, game_over, score

Agent
- game
- Model
 Training:
 - state = get-state (game)
 - action = get-move (state)
   - model.predict()
 - reward, game-over, score = game.play-step (action)
 - new_state = get-state (game)
 - remember
 - model.train

Model (Pytorch)
Linear_QNet (DQN)
- model. predict (state)
  → action

Reward	
eat food	+10
game over	-10
else	0

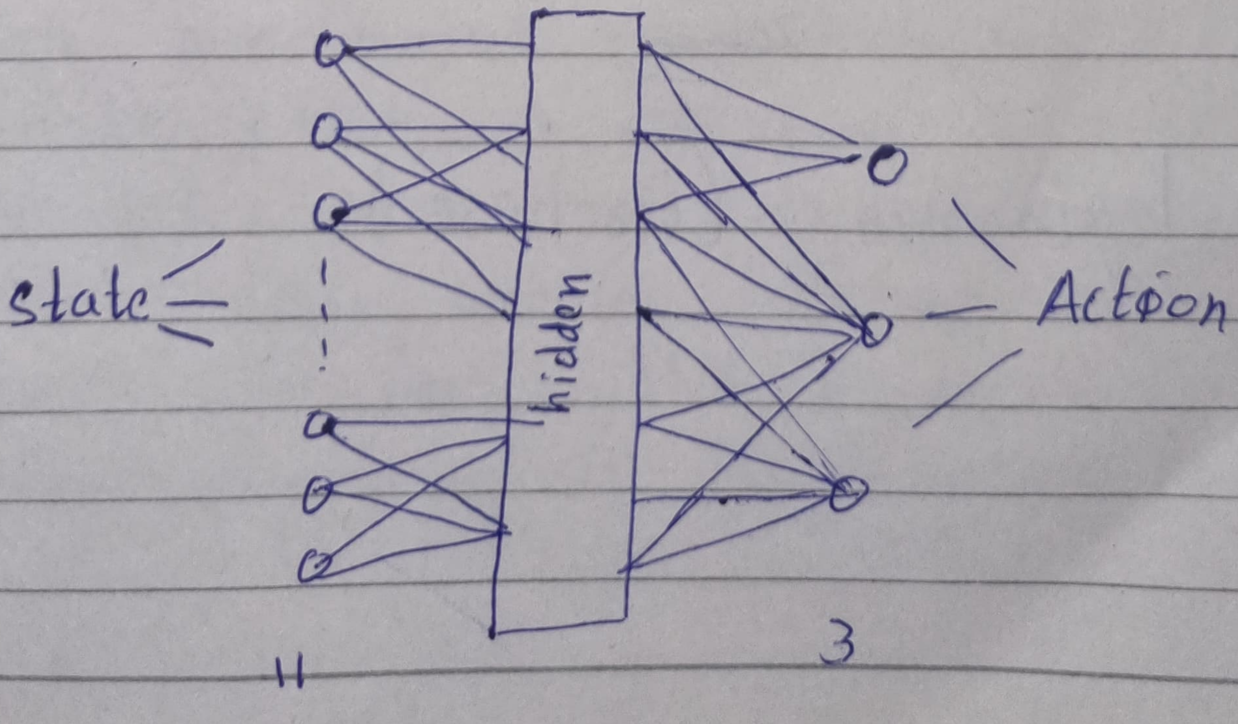
Action	
[1,0,0]	straight
[0 1,0]	right
[0, 0, 1]	left

State

danger straight
danger right
danger left

direction left
direction right
direction up
direction down

food left
food right
food up
food down



(Deep) Q learning

Q value = Quality of action

init Qvalue (= init model)

repeat:

→ Choose action (model.predict(state))

Perform action

Measure reward

← Update Q value (+ train model)

Bellman equation

$$\text{New } Q(s,a) = Q(s,a) + \alpha\left[R(s,a) + \gamma \max Q'(s',a') - Q(s,a)\right]$$

α = learning rate

R = reward

γ = discount rate

Q = model.predict(state$_0$)

$Q_{new} = R + \gamma \times \max(Q(\text{state}_1))$

$\text{loss} = (Q_{new} - Q)^2$