

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC & KỸ THUẬT MÁY TÍNH



BÁO CÁO LUẬN VĂN TỐT NGHIỆP

**Xây dựng hệ thống điều khiển đèn giao thông
bằng Deep Reinforcement Learning**

Hội đồng: Khoa học Máy tính
GVHD: TS. Nguyễn Đức Dũng
GVPB: PGS. TS. Huỳnh Tường Nguyên

Sinh viên thực hiện	MSSV
Đỗ Lê Duy	1510443
Nguyễn Đỗ Đức Anh	1510062

Tp. Hồ Chí Minh, 05/2019

Lời cam đoan

Chúng tôi xin cam đoan đây là công trình nghiên cứu của riêng chúng tôi dưới sự hướng dẫn của TS. Nguyễn Đức Dũng. Nội dung nghiên cứu và các kết quả đều là trung thực và chưa từng được công bố trước đây. Các số liệu được sử dụng cho quá trình phân tích, nhận xét được chính chúng tôi thu thập từ nhiều nguồn khác nhau và sẽ được ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, chúng tôi cũng có sử dụng một số nhận xét, đánh giá và số liệu của các tác giả khác, cơ quan tổ chức khác. Tất cả đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kì sự gian lận nào, chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung thực tập tốt nghiệp của mình. Trường đại học Bách Khoa thành phố Hồ Chí Minh không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện.

Lời cảm ơn

Để hoàn thành luận văn tốt nghiệp này, chúng tôi tỏ lòng biết ơn sâu sắc đến TS. Nguyễn Đức Dũng đã hướng dẫn tận tình trong suốt quá trình nghiên cứu.

Chúng tôi chân thành cảm ơn quý thầy, cô trong Khoa Khoa Học và Kỹ Thuật Máy Tính, phòng thí nghiệm hiệu năng cao, các giảng viên Lab Trường Đại học Bách Khoa Thành phố Hồ Chí Minh đã tận tình truyền đạt kiến thức trong những năm chúng tôi học tập ở trường. Với vốn kiến thức tích lũy được trong suốt quá trình học tập không chỉ là nền tảng cho quá trình nghiên cứu mà còn là hành trang để bước vào đời một cách tự tin.

Cuối cùng, chúng tôi xin chúc quý thầy, cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý.

Tóm tắt nội dung

Báo cáo luận văn tốt nghiệp của nhóm bao gồm phần cơ sở lý thuyết về: học tăng cường (Reinforcement Learning), các giải thuật nổi bật trong học tăng cường như: Deep Q-learning, Policy Gradient, Actor-Critic,... Từ cơ sở lý thuyết đó, nhóm chúng tôi tiến hành tìm hiểu các mô hình cho bài toán điều khiển đèn giao thông qua các bài báo. Nhóm chọn ra một số mô hình từ quá trình nghiên cứu, đánh giá sơ lược ưu nhược điểm và chọn ra mô hình phù hợp để tiến hành hiện thực.

Qua quá trình nghiên cứu, nhóm sẽ trình bày về cách sử dụng công cụ mô phỏng SUMO để tạo ra bản đồ, thiết lập luật di chuyển, chọn loại xe và số lượng phương tiện, mô phỏng sao cho gần giống với thực tế. Ngôn ngữ lập trình Python, thư viện Keras, Tensor sẽ được sử dụng để xây dựng mô hình huấn luyện học tăng cường. Mô hình này sẽ được thử nghiệm và đánh giá bằng những độ đo khác nhau so với hệ thống giao thông cố định hiện nay và giữa các mô hình với nhau để tìm ra chiến lược xây dựng tốt nhất và tìm ra mô hình đem lại hiệu quả.

Cuối cùng là kết luận những kết quả đã được, những hạn chế và kế hoạch trong tương lai.

Mục lục

1	Giới thiệu	1
2	Học tăng cường	4
2.1	Nền tảng của học tăng cường	4
2.1.1	Học tăng cường là gì?	4
2.1.2	Các cách tiếp cận trong học tăng cường	5
2.2	Giải thuật Q-learning và Deep-Q-learning	6
2.2.1	Q-learning	6
2.2.2	Deep-Q-learning và Experience Replay	7
2.3	Target Network	8
2.4	Double DQN	9
2.5	Dueling DQN	9
2.6	Prioritized Experience Replay	10
2.7	Convolution neural network	11
2.7.1	Lớp tích chập (Convolutional layer)	11
2.7.2	Lớp Pooling	12
2.7.3	Lớp Rectified Linear Units (ReLU layer)	13
2.7.4	Fully-connected layers (các lớp được kết nối đầy đủ)	13
2.7.5	Lan truyền ngược (Backpropagation)	13
2.7.6	Siêu tham số (Hyperparameters)	14
2.8	Policy Gradient	14
2.9	Actor-Critic [6]	15
3	Các mô hình điều khiển	17
3.1	Traffic Light Control Using Deep Policy-Gradient and Value-Function Based Reinforcement Learning	17
3.1.1	Trạng thái	17
3.1.2	Hành động	18
3.1.3	Phần thưởng	18
3.1.4	Hàm mục tiêu và huấn luyện hệ thống	19
3.1.5	Kiến trúc tổng quan	20
3.2	Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks	21
3.2.1	Trạng thái	21
3.2.2	Hành động	23
3.2.3	Phần thưởng	24
3.2.4	Những kỹ thuật được áp dụng	25
3.2.5	Kiến trúc tổng quan	28

4 Phân tích và lựa chọn mô hình	30
4.1 Trạng thái	30
4.2 Hành động	30
4.3 Phần thưởng	32
4.4 Mạng nơ-ron và các kỹ thuật được áp dụng	32
5 Hiện thực mô hình	35
5.1 Đề xuất công cụ sử dụng	35
5.2 Xây dựng mô phỏng giao thông bằng SUMO	35
5.2.1 Xây dựng bản đồ mô phỏng	36
5.2.2 Các loại phương tiện	38
5.2.3 Xây dựng kịch bản giao thông	38
5.3 Mô hình bài toán	39
5.3.1 Trạng thái	39
5.3.2 Hành động	41
5.3.3 Phần thưởng	43
5.4 Mạng chính Deep-Q-Learning	44
5.5 Mạng mục tiêu Deep-Q-Learning	45
5.6 Thông số liên quan	45
5.7 Chiến lược huấn luyện	45
6 Đánh giá và kết quả đạt được	47
6.1 Phương pháp đánh giá	47
6.2 Mô phỏng hệ thống giao thông tĩnh	48
6.3 Quá trình thử nghiệm và mô hình đề xuất	49
6.4 Kết quả	50
6.4.1 Chiến lược 1	50
6.4.2 Chiến lược 2	54
6.4.3 Nhận xét	56
6.4.4 Kết quả huấn luyện ngã ba	57
7 Tổng kết	59
7.1 Kết quả đạt được	59
7.2 Hạn chế	60
7.3 Hướng phát triển	60

Danh sách bảng

5.1	Thông số mô hình hành động cách 1	46
5.2	Thông số mô hình hành động cách 2	46
6.1	Kí hiệu cho chương 6	47
6.2	Thời gian cố định cho hệ thống STL (33,33)	48
6.3	Thời gian cố định cho hệ thống STL (40,40)	48
6.4	Kết quả đánh giá của hệ thống STL (33,33) trên ngã tư Lý Thường Kiệt - 3/2 . .	49
6.5	Kết quả đánh giá của hệ thống STL (40,40) trên ngã tư Lý Thường Kiệt - 3/2 . .	49
6.6	So sánh AWT của 2 hệ thống STL với Agent 1	52
6.7	So sánh AWT của 2 hệ thống STL với Agent 1 trên 5 kịch bản ngẫu nhiên . . .	53
6.8	So sánh AWT của 2 hệ thống STL với Agent 2	56
6.9	So sánh AWT của 2 hệ thống STL với Agent 2 trên 5 kịch bản ngẫu nhiên . . .	56
6.10	So sánh AWT của 2 hệ thống STL kịch bản ngẫu nhiên giao lộ ngã ba	57
6.11	So sánh AWT của 2 hệ thống STL kịch bản ngẫu nhiên giao lộ ngã ba	58

Danh sách hình vẽ

2.1	Ý tưởng cơ bản và những thành phần liên quan trong học tăng cường	5
2.2	Ví dụ minh họa cách tiếp cận dựa trên giá trị	6
2.3	Quá trình huấn luyện bằng giải thuật Q-learning	7
2.4	So sánh Q-learning và Deep-Q-learning	8
2.5	Minh họa kĩ thuật Experience Replay	8
2.6	Minh họa kĩ thuật Dueling DQN	10
2.7	Minh họa kĩ thuật Prioritized experience replay.	10
2.8	Minh họa đầu vào và đầu ra của Conv Layer	12
2.9	Minh họa đầu vào và đầu ra của lớp pooling sử dụng max pooling	12
2.10	Minh họa cách hoạt động của lớp Fully-connected	13
2.11	Actor-critic kết hợp giữa value-based và policy-based	16
2.12	Kiến trúc tổng quan của Actor-critic	16
3.1	Sơ lược kiến trúc mô hình	17
3.2	Snapshot từ một ngã tư vào một thời điểm	22
3.3	Ma trận biểu diễn vị trí xe của ngã tư	22
3.5	Một ví dụ về không gian hành động trên giao lộ Hình 3.2	24
3.6	Kiến trúc của mạng CNN để xấp xỉ Q value	26
3.7	Kiến trúc tổng quan của mô hình.	28
4.1	Chính sách hành động của mô hình theo cách 1	31
4.3	Mạng nơ-ron	33
4.4	Kiến trúc tổng quan mô hình hiện thực	33
5.1	Giao lộ mô phỏng ngã tư ban đầu trong SUMO	36
5.2	Bản đồ mô phỏng sau khi được chuyển đổi	36
5.4	Ngã tư mô phỏng với xe cộ	38
5.5	Ngã tư kẹt xe	39
5.6	Chia làn đường thành những ô nhỏ để ánh xạ sang ma trận	40
5.7	Biểu diễn state dưới dạng ma trận	41
5.8	Chuyển đổi sang ma trận trạng thái map 1-1 bằng thư viện traci	42
5.9	Chuyển đổi sang ma trận trạng thái map 1-N bằng thư viện traci	42
6.1	Thời gian chờ trung bình khi thử nghiệm một mô hình đã hiện thực	49
6.2	AWT trong quá trình huấn luyện theo chiến lược 1 - LOW	50
6.3	AWT trong quá trình huấn luyện theo chiến lược 1 - HIGH	51
6.4	AWT trong quá trình huấn luyện theo chiến lược 1 - NS	51
6.5	AWT trong quá trình huấn luyện theo chiến lược 1 - EW	52
6.7	AWT trong quá trình huấn luyện theo chiến lược 2 - LOW	54
6.8	AWT trong quá trình huấn luyện theo chiến lược 2 - HIGH	55

6.9 AWT trong quá trình huấn luyện theo chiến lược 2 - NS	55
6.10 AWT trong quá trình huấn luyện theo chiến lược 2 - EW	56

Chương 1

Giới thiệu

Cùng với sự phát triển mạnh mẽ của các hệ thống tính toán, những hệ thống ứng dụng công nghệ trí tuệ nhân tạo (AI) ngày càng nhiều. Có nhiều bài toán được đặt ra cần giải quyết nhằm phục vụ nhu cầu cuộc sống với mong muốn không cần đến tác động của con người, giảm thiểu nguồn nhân lực mà lại cực kì hiệu quả. AI, do đó, đã trở thành một công cụ vô cùng quan trọng trong các giải pháp được đề xuất, từ góc độ ứng dụng đến thực tiễn.

Một trong những bài toán phải kể đến đó là bài toán nghiên cứu và phát triển hệ thống AI để giải quyết vấn đề giao thông mà không cần sự trợ giúp từ con người cũng đã được thực hiện từ rất lâu và rất nhiều nơi trên thế giới đã lắp đặt hệ thống này như Mỹ và các nước Châu Âu, kết quả của nó đã đóng góp một phần không nhỏ giải quyết vấn đề ùn tắc, giảm thiểu ô nhiễm môi trường, thời gian chờ của con người, năng lượng tiêu hao. Đây cũng chính là bài toán mà chúng tôi sẽ đề cập đến và tiến hành nghiên cứu.

Để làm được điều này, các cảm biến sẽ được lắp đặt khắp nơi để thu thập thông tin về tình hình phương tiện, lưu lượng giao thông cũng như cập nhật các vấn đề thời tiết. Những thông tin này sau đó sẽ được tổng hợp và gửi đến hệ thống, từ đó hệ thống sẽ tính toán và điều chỉnh thời gian của các đèn giao thông với mục đích là cải thiện tình hình giao thông một cách tối ưu, làm giảm thời gian chờ của các phương tiện di chuyển, giảm ùn tắc và ô nhiễm môi trường. Hệ thống trở nên mạnh mẽ nhờ sự vận dụng kiến thức của Machine Learning và Deep Learning đòi hỏi phải hiểu biết những kiến thức nền tảng mới có thể xây dựng.

Ở Việt Nam hiện nay chưa có nơi nào sử dụng hệ thống như vậy, ngay cả những thành phố đông dân như thành phố Hồ Chí Minh hay Hà Nội bởi sự thiếu hụt chi phí từ việc nghiên cứu cho đến lắp đặt công nghệ này hay mua lại từ nước ngoài.

Hiện nay, với tốc độ tăng trưởng dân số, sự phổ biến của phương tiện giao thông cá nhân tại nhiều thành phố lớn như TP.HCM, Hà Nội,... nhưng cơ sở hạ tầng giao thông lại không phát triển kịp thời dẫn đến tình trạng ùn tắc và quá tải giao thông diễn ra liên tục. Mặc dù có rất nhiều hệ thống điều khiển đèn giao thông được bố trí khắp nơi tại các giao lộ, tuy nhiên những hệ thống hiện tại chưa tối ưu vì khoảng thời gian của các tín hiệu: đèn xanh, đèn đỏ, đèn vàng là cố định. Do đó, vào những giờ cao điểm, chúng ta thường thấy những chú công an giao thông có nhiệm vụ quan sát mật độ và điều khiển giao thông thay cho hệ thống đèn. Chúng tôi nhận thấy việc điều khiển giao thông như vậy gây ra rất nhiều khó khăn: kẹt xe, tai nạn, lãng phí nhân lực (điều tiết giao thông), ô nhiễm môi trường,...

Những vấn đề đó cho thấy sự cần thiết phải có một hệ thống hỗ trợ điều phối giao thông thật sự thông minh. Hệ thống đó phải có khả năng đáp ứng và điều phối trong các điều kiện khác nhau một cách linh hoạt với dữ liệu giao thông có được từ các hệ thống giám sát tự động. Cùng với sự cải thiện dần trong ý thức của người dân, việc tuân thủ các quy tắc và sự điều phối của hệ thống sẽ đóng vai trò vô cùng quan trọng trong giảm thiểu ùn tắc của thành phố. Đây chính là

động lực lớn thôi thúc nhóm thực hiện nghiên cứu này.

Có thể nhận thấy kết quả của việc nghiên cứu đề tài này có thể đem lại một phần đóng góp về nền móng xây dựng một hệ thống điều khiển đèn giao thông dựa trên dữ liệu thời gian thực bằng Deep Reinforcement Learning, giúp giảm thiểu ùn tắc giao thông, tai nạn cũng như thời gian chờ của người tham gia giao thông.

Bên cạnh đó, thông qua việc nghiên cứu, nhóm còn có thể học thêm được các kiến thức chuyên nghành, thông qua các bài báo, các sách chuyên môn mà các thành viên chưa từng được tiếp cận trước đó. Đây là cơ hội tốt để nhóm phát triển các kỹ năng nghiên cứu khoa học, không chỉ cho việc nghiên cứu mà còn cho việc ứng dụng vào thực tế.

Mục tiêu đề tài

Mục tiêu quan trọng của việc nghiên cứu là xây dựng được hệ thống điều khiển đèn giao thông tại giao lộ (ngã tư, ngã 3) bằng Deep Reinforcement Learning và kiểm thử mô hình bằng công cụ mô phỏng Simulation of Urban MObility (SUMO). Cụ thể hệ thống xây dựng phải đạt được những mục tiêu sau:

- Có khả năng ra quyết định giúp điều hướng phương tiện di chuyển một cách hiệu quả, giảm thời gian chờ và ùn tắc giao thông.
- Mô hình linh hoạt, hoạt động tốt trong các tình huống bất thường, có khả năng mở rộng lên các giao lộ phức tạp (ngã 5, ngã 6, vòng xoay,...).
- Hiểu được trạng thái giao lộ dựa vào dữ liệu thu thập từ mạng lưới giao thông.
- Việc đánh giá, kiểm chứng phải cho ra kết quả là tốt nhất khi so với các mô hình đèn giao thông hiện tại, với các mô hình khác.

Phương pháp nghiên cứu và hướng tiếp cận

Ở thời điểm hiện tại, đã có rất nhiều bài báo khoa học liên quan đến việc xây dựng hệ thống điều khiển đèn giao thông với các mô hình khác nhau và hiệu quả khác nhau. Vì thế, việc nghiên cứu xây dựng mô hình cần xác định rõ ràng phương pháp đem lại hiệu quả tốt nhất. Mọi phương pháp đều dựa trên mô hình học tăng cường.

Để đạt được kết quả khả quan khi bắt đầu, chúng tôi đã phân tích bài toán, nghiên cứu các mô hình từ các bài báo nổi tiếng. Qua bài báo, chúng tôi tiến hành liệt kê và tìm hiểu những kiến thức nền về lý thuyết cũng như ứng dụng các kĩ thuật trong Machine Learning và Deep Learning.

Sau khi nghiên cứu các mô hình qua các bài báo, nhiệm vụ tiếp đó là đánh giá, so sánh từng mô hình và lựa chọn ra mô hình phù hợp. Cuối cùng là huấn luyện theo mô hình đã chọn và kiểm chứng bằng SUMO.

Tóm lược

Trong chương 2 tới đây, chúng tôi giới thiệu về kiến thức nền tảng của học tăng cường, các giải thuật để xây dựng nên mô hình học tăng cường như Q-learning, Deep-Q-learning, Policy Gradient, Actor-Critic và những kĩ thuật tiên tiến gồm Double Deep-Q-learning, Dueling Deep-Qlearning, Prioritized Experience Replay được áp dụng nhằm cải thiện hiệu suất trong quá trình huấn luyện.

Lý thuyết và kiến thức nền tảng mà chúng tôi tìm tòi, tìm hiểu đều được rút ra từ những bài báo viết liên quan đến mục tiêu chúng tôi nghiên cứu. Chi tiết và nội dung về cách hiện thực trạng thái, hành động, phần thưởng và kiến trúc tổng quan mô hình của từng bài báo sẽ được chúng tôi trình bày cụ thể ở chương 3.

Ở chương 4, nhóm tiến hành phân tích, lựa chọn để đưa ra mô hình sẽ xây dựng và đánh giá hiệu suất. Bắt đầu từ những thứ cơ bản cần phải có trong một mô hình học tăng cường là trạng thái, hành động và phần thưởng và kết thúc là kiến trúc mạng nơ-ron mà nhóm sử dụng.

Với mô hình được trình bày ở chương 4 thì ở chương 5, chúng tôi trình bày cụ thể cách hiện thực mô hình trạng thái, hành động, phần thưởng và xây dựng mạng Deep-Q-Learning gồm mạng chính và mạng mục tiêu. Ngoài ra, chương này nhóm cũng sẽ trình bày công cụ mô phỏng SUMO, từ cách tạo bản đồ, chỉnh sửa con đường, giao lộ đến việc thiết lập phương tiện, loại xe, kích thước và cuối cùng là định ra kịch bản giao thông với các chiều xe, thời gian xuất hiện và số lượng xe cộ.

Sau khi hoàn thành xây dựng mô hình, công việc đánh giá hiệu quả nằm ở chương 6. Chương này sẽ nêu ra phương pháp mà chúng tôi đánh giá, giới thiệu về hệ thống giao thông tĩnh hiện nay và cách mô phỏng, đánh giá các mô hình đã lựa chọn và đưa ra mô hình có kết quả tốt nhất, trình bày kết quả so sánh giữa mô hình và hệ thống giao thông tĩnh với độ đo là thời gian chờ trung bình.

Và cuối cùng, quan trọng và không thể thiếu đó là chương tổng kết (chương 7). Những kết quả mà trong quá trình nghiên cứu chúng tôi đạt được cũng như mặt hạn chế và khó khăn gặp phải sẽ được nêu rõ ở chương này. Ngoài ra, với các vấn đề như vậy, nhóm cũng đề xuất cách giải quyết và hướng phát triển, định hướng trong tương lai.

Chương 2

Học tăng cường

Xử lý điều khiển tín hiệu đèn giao thông đã được nghiên cứu khá nhiều năm, từ khi các thành phố hình thành. Tuy nhiên, chỉ những năm gần đây khi AI chứng minh được vai trò nổi trội của nó trong cách lĩnh vực thì chúng ta mới thấy một nhánh nghiên cứu về mảng này sử dụng công nghệ học máy.

Trong số các nghiên cứu về hướng này, có 2 nghiên cứu tiêu biểu liên quan đến kỹ thuật học tăng cường là:

- Traffic Light Control Using Deep Policy-Gradient and Value-Function Based Reinforcement Learning (DePGVF) in 2017 [9]
- Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks (DeTLC) in 2018 [8]

Hai mô hình này đều dựa trên giải thuật học tăng cường. Trong nghiên cứu **DePGVF**, các tác giả giới thiệu hai cách tiếp cận để xây dựng mô hình là **value-based** và **policy-based**. Với bài báo **DeTLC**, ta sẽ thấy được một mô hình chi tiết hơn với sự kết hợp giữa kiến trúc học tăng cường và những kỹ thuật tiên tiến áp dụng để cải thiện hiệu suất, sự ổn định và an toàn. Việc so sánh và lựa chọn mô hình sẽ được trình bày ở chương 4.

Trong phần này, chúng tôi trình bày chi tiết các khái niệm của học tăng cường (Reinforcement Learning), Convolutional Neural Network (CNN), Deep-Q-learning, Policy gradient,... Tất cả đều là những kỹ thuật quan trọng trong Reinforcement Learning.

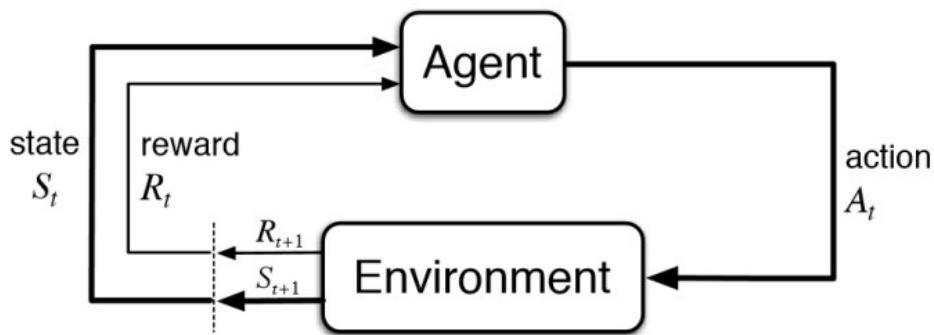
2.1 Nền tảng của học tăng cường

2.1.1 Học tăng cường là gì?

Trong ngành khoa học máy tính, học tăng cường (reinforcement learning) [1] là một lĩnh vực con của học máy, khác với học có giám sát và học không giám sát. Trong học tăng cường, Agent (robot, program,...) sẽ tương tác với môi trường (environment) bằng hành động khi đang ở trạng thái hiện tại: (action, current state), sau đó nhận phần thưởng và trạng thái kế tiếp: (reward, next state) từ môi trường. Mục tiêu của Agent là cực đại tổng phần thưởng tích lũy được về lâu dài.

Mô hình học tăng cường có thể kí hiệu bởi 4 thành phần $\langle S, A, R, T \rangle$:

- S : không gian các trạng thái có thể, s là 1 trạng thái cụ thể ($s \in S$)
- A : không gian các hành động có thể, a là 1 action cụ thể ($a \in A$)
- R : không gian phần thưởng, $r_{s,a}$ nghĩa là phần thưởng nhận được khi thực hiện hành động a tại trạng thái s .
- T : hàm chuyển đổi, thể hiện xác suất chuyển từ trạng thái này sang trạng thái khác.



Hình 2.1: Ý tưởng cơ bản và những thành phần liên quan trong học tăng cường

Các thuật toán học tăng cường cố gắng tìm một chiến lược ánh xạ các trạng thái s của môi trường thành các hành động a mà Agent nên chọn khi ở các trạng thái đó, chiến lược ánh xạ đó được kí hiệu là π hay còn gọi là chính sách (policy).

Khác với học có giám sát, trong học tăng cường không có các cặp: dữ liệu vào (input) và kết quả đúng (label), thay vào đó Agent trong học tăng cường chỉ tương tác trực tiếp với môi trường và thử sai (trial and error) để có được dữ liệu huấn luyện. Qua đó, mục tiêu cuối cùng của Agent là tìm ra được chính sách tối ưu (optimal policy π^*) để cực đại tổng phần thưởng về lâu dài từ trạng thái hiện tại.

2.1.2 Các cách tiếp cận trong học tăng cường [15]

Đối với những vấn đề về học tăng cường được đặt ra, chúng tôi sẽ giới thiệu 2 cách tiếp cận để giải quyết đó là: **dựa trên giá trị** (value-based), **dựa trên chính sách** (policy-based).

1. Value-based:

Trong các giải thuật dựa trên giá trị: mục tiêu của chúng ta là cực đại **hàm giá trị** $V(s)$, đây là hàm cho chúng ta biết được cực đại phần thưởng kỳ vọng trong tương lai (maximum expected future reward) của Agent khi đang ở trạng thái s . Giá trị trả về của hàm $V(s)$, hay còn gọi là giá trị trạng thái là tổng phần thưởng mà Agent có thể nhận được trong tương lai bắt đầu từ trạng thái s . $V(s)$ được tính theo công thức:

$$v_\pi(s) = E_\pi[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots | s_t = s] \quad (2.1)$$

Trong công thức trên: γ là hệ số giảm (discount factor), $\gamma \in [0, 1]$, nghĩa là những phần thưởng sẽ giảm dần giá trị trong tương lai tránh việc giá trị $V(s)$ tiến về vô cực. Chú ý: γ càng nhỏ thì mức độ quan trọng của phần thưởng trong tương lai càng giảm.

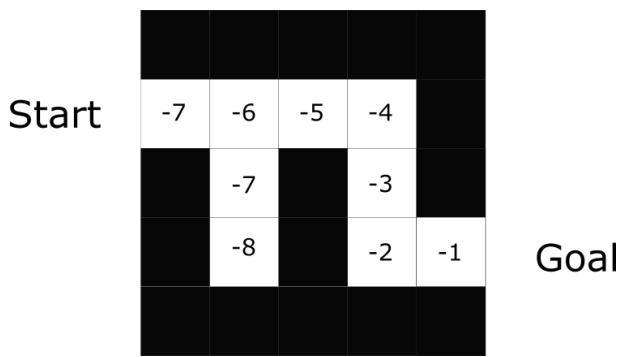
Agent sẽ sử dụng hàm giá trị này để chọn trạng thái tiếp theo ở mỗi bước bằng cách chọn trạng thái tiếp theo có giá trị lớn nhất.

Trong ví dụ Hình 2.2, Agent sẽ dựa vào hàm giá trị để xác định đường đi từ **Start** đến **Goal** bằng cách chọn giá trị lớn nhất trong các trạng thái kế tiếp ở mỗi bước. Kết quả: $(-7) \rightarrow (-6) \rightarrow (-5) \rightarrow (-4) \rightarrow (-3) \rightarrow (-2) \rightarrow (-1)$.

2. Policy-based:

Trong các giải thuật dựa trên chính sách, mục tiêu của chúng ta là tìm ra 1 hàm chính sách tối ưu (policy function) $\pi_\theta(s)$ trực tiếp mà không cần sử dụng **hàm giá trị** như đã giới thiệu. Hàm chính sách này sẽ ánh xạ ra hành động a tốt nhất dựa trên đầu vào là trạng thái s :

$$a = \pi_\theta(s)$$



Hình 2.2: Ví dụ minh họa cách tiếp cận dựa trên giá trị

Có 2 loại chính sách:

- **Chính sách xác định (deterministic):** khi đầu vào là 1 trạng thái s thì đầu ra luôn luôn là hành động a . Chính sách xác định được dùng khi môi trường là xác định, đây là môi trường khi thực hiện hành động a khi ở trạng thái s thì chắc chắn trạng thái kế tiếp là s' .
- **Chính sách ngẫu nhiên (stochastic):** khi đầu vào là trạng 1 trạng thái s thì đầu ra là phân phối xác suất của các hành động. Nghĩa là thay vì trả về 1 hành động duy nhất, chính sách ngẫu nhiên trả về xác suất của từng hành động đối với trạng thái đầu vào. Chính sách ngẫu nhiên được sử dụng khi môi trường không xác định. Nghĩa là khi thực hiện hành động a khi đang ở trạng thái s thì trạng thái tiếp theo chưa chắc là s' .

$$\pi_\theta(a|s) = P(A_t = a|S_t = s)$$

Đây là công thức xác định xác suất thực hiện hành động a khi đang ở trạng thái s .

2.2 Giải thuật Q-learning và Deep-Q-learning

2.2.1 Q-learning

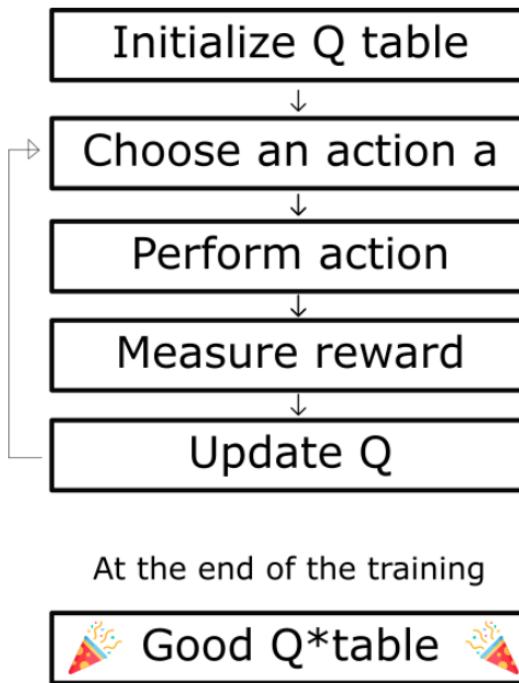
Q-learning [12] là 1 giải thuật dựa trên giá trị, tức là Agent sẽ dựa trên *hàm giá trị* để quyết định hành động tốt nhất. Để xây dựng được hàm giá trị đó, chúng ta sẽ thiết lập 1 bảng gọi là: **Q-table**. Một **Q-table** bao gồm:

- Các **cột**: thể hiện cho từng hành động a trong không gian hành động A .
- Các **hàng**: thể hiện cho từng trạng thái s trong không gian trạng thái S .
- Giá trị mỗi ô: trong Q-table hay còn gọi là **Q value** là giá trị phần thưởng mong đợi cực đại của Agent khi ở trạng thái s và thực hiện hành động a .

Như vậy, không cần hiện thực *hàm chính sách* chúng ta vẫn có thể chọn ra hành động a tốt nhất cho trạng thái s bằng cách chọn ra ô có giá trị lớn nhất tại hàng s trong Q-table.

Tính Q value cho mỗi ô: Để làm được điều này, chúng ta sẽ xây dựng 1 hàm gọi là: **hàm giá trị hành động (action value function)** hay còn gọi là **Q function**, với đầu vào là trạng thái s , hành động a và đầu ra là phần thưởng mong đợi cực đại Q value. Công thức để xác định Q value là:

$$\begin{aligned} Q^\pi(s, a) &= E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots | s_t = s, a_t = a, \pi] \\ &= \sum_{k=0}^{\infty} [\gamma^k r_{t+k} | s_t = s, a_t = a, \pi] \end{aligned} \tag{2.2}$$



Hình 2.3: Quá trình huấn luyện bằng giải thuật Q-learning

Quá trình huấn luyện sử dụng giải thuật **Q-learning**:

Trong hình 2.3, ở bước cập nhật **Q-table** chúng ta dùng **phương trình Bellman**:

$$\begin{aligned}
 Q_{New}(s, a) &= Q(s, a) + \alpha \left[R(s, a) + \gamma \max_a Q'(s', a') - Q(s, a) \right] \\
 &= Q(s, a) + \alpha \Delta Q(s, a)
 \end{aligned} \tag{2.3}$$

Trong đó:

- α : là hệ số học.
- $Q(s, a)$: là Q value hiện tại.
- $R(s, a)$: là phần thưởng nhận được khi Agent đang ở trạng thái s và thực hiện hành động a.
- $\max Q'(s', a')$: là giá trị Q value khi mà Agent ở trạng thái s' thì a' là hành động đem lại Q value lớn nhất.

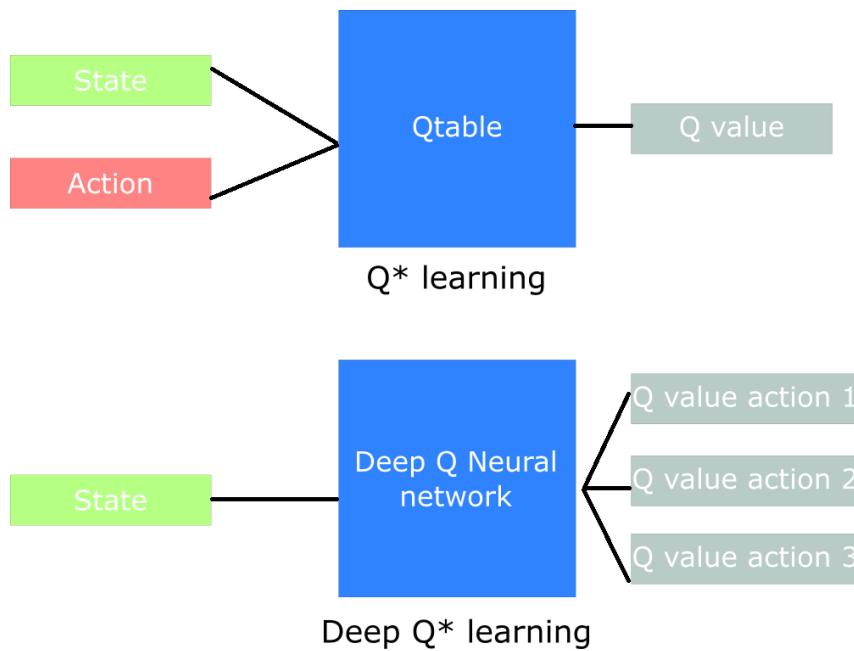
2.2.2 Deep-Q-learning và Experience Replay

Vấn đề với Q-learning: sau quá trình huấn luyện, Agent có thể sử dụng Q-table để xác định hành động có Q value tốt nhất và thực hiện. Tuy nhiên, đối với những bài toán có không gian trạng thái vô cùng lớn thì việc xây dựng Q-table như vậy là bất khả thi. Do đó, 1 giải thuật mới được thay thế để giải quyết vấn đề này gọi là: **Deep-Q-learning** [14].

Giải thuật **Deep-Q-learning** không xây dựng **Q-table**, mà thay vào đó sẽ xây dựng 1 **mạng nơ-ron (Deep Q Neural Network)** nhằm xấp xỉ các **Q-values** ứng với mỗi hành động (Hình 2.4). Từ đó, chọn ra hành động có Q value lớn nhất để thực hiện.

Đối với giải thuật **Deep-Q-learning** [5], thay vì cập nhật Q-table như Q-learning thì chúng ta sẽ cập nhật trọng số (θ) của mạng nơ-ron bằng cách cực tiểu hàm mất mát:

$$J(\theta) = E [(y_t - Q(s_t, a_t; \theta))^2]. \tag{2.4}$$



Hình 2.4: So sánh Q-learning và Deep-Q-learning



Hình 2.5: Minh họa kĩ thuật Experience Replay

Trong đó: hàm mất mát là tổng bình phương sai số của Q value dự đoán và Q value (Q target), $y_t = r_t + \gamma \max_a Q(s', a; \theta)$ hay **Q target** là tổng của: phần thưởng tức thời khi thực hiện hành động a và phần thưởng cực đại ở trạng thái kế tiếp s' .

Như vậy, quá trình huấn luyện mạng nơ-ron diễn ra liên tục bằng cách: Agent thực hiện hành động lên môi trường ở mỗi timestep, và nhận về một tuple hay kinh nghiệm (experience): (s, a, r, s') sau đó cập nhật trọng số θ bằng gradient descent.

$$\Delta\theta_t = \alpha[(r_t + \gamma \max_{a_{t+1}}(Q(s_{t+1}, a_{t+1}, \theta))) - Q(s_t, a_t, \theta)] \nabla_\theta Q(s_t, a_t, \theta) \quad (2.5)$$

Cách huấn luyện Agent ở đây nảy sinh ra một vấn đề là: sẽ có những tuple hiếm gặp do đó Agent được học rất ít lần dẫn đến không đưa ra hành động chính xác trong tương lai, hoặc là những quan sát đã được học từ lâu thường dễ bị “quên” do nó học những tuple mới.

Để giải quyết vấn đề này, một kĩ thuật được đưa ra là: **Experience Replay**. Ý tưởng của Experience Replay là sử dụng 1 bộ đệm (Replaybuffer) để lưu lại những kinh nghiệm trong khi tương tác với môi trường, sau đó lấy ra từng batch các kinh nghiệm để huấn luyện mạng nơ-ron.

2.3 Target Network

Vấn đề: Đối với Deep Q Network (DQN) trong giải thuật Deep-Q-learning chúng ta xây dựng, giá trị Q target liên tục thay đổi do chúng ta liên tục cập nhật trọng số của Q Network

trong lúc huấn luyện, có nghĩa là chúng ta đang cố gắng đạt tới 1 mục tiêu mà nó liên tục thay đổi. Do đó, quá trình huấn luyện trở nên không được ổn định.

Target network [13] là một kỹ thuật để giải quyết vấn đề này bằng cách tạo thêm một network thứ hai (target network) giống hệt DQN ban đầu (primary network). Target Network được dùng để sinh ra Q target để tính độ mất mát (loss) cho mỗi hành động lúc huấn luyện, primary network được cập nhật bằng Mean Square Error (MSE) theo hàm mất mát J:

$$J = \sum_s [Q_{target}(s, a) - Q(s, a; \theta)]^2. \quad (2.6)$$

Trong đó: $Q_{target}(s, a) = r_t + \gamma \max_a Q(s, a; \theta^-)$. Tham số của primary network kí hiệu là θ cập nhật liên tục trong lúc huấn luyện. Tham số của target network được kí hiệu là θ^- được cập nhật theo trọng số α :

$$\theta^- = \alpha \theta^- + (1 - \alpha) \theta. \quad (2.7)$$

Trong đó, α là tỉ lệ cập nhật, biểu thị mức độ ảnh hưởng của tham số mới nhất đối với những thành phần trong target network.

2.4 Double DQN

Double DQN [13], [4] là một kỹ thuật dùng để giải quyết vấn đề overestimation trong hàm giá trị hành động (action value function) và cải thiện hiệu suất của giải thuật. Như đã giới thiệu, giá trị Q target sẽ được tính bởi target network theo công thức:

$$Q_{target}(s, a) = r + \gamma \max_{a'} Q(s', a'; \theta^-). \quad (2.8)$$

Nếu giá trị cực đại Q value của s' bị ước lượng quá mức thì sẽ dẫn đến Q target cũng bị ước lượng quá mức. Để giải quyết vấn đề này, chúng ta sẽ dùng primary network để chọn ra hành động a' có Q value lớn nhất khi ở s' và target network để tính ra Q value cho trạng thái s' với hành động a' vừa chọn.

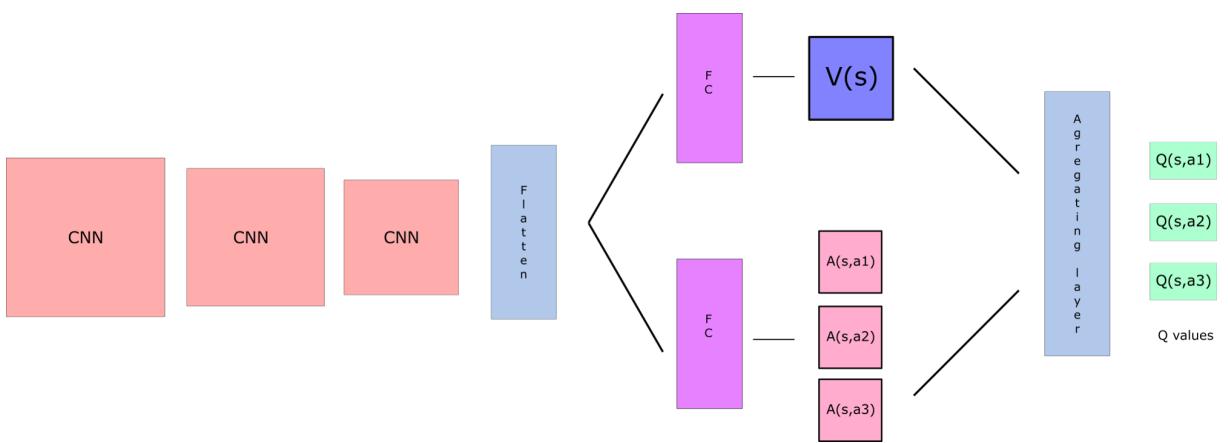
$$Q_{target}(s, a) = r + \gamma Q(s', \arg \max_{a'} (Q(s', a'; \theta)); \theta^-). \quad (2.9)$$

2.5 Dueling DQN

Như chúng tôi đã giới thiệu, Q value là giá trị phần thưởng mong đợi cực đại của Agent khi ở trạng thái s và thực hiện hành động a. Kí hiệu là $Q(s, a)$. Do đó, $Q(s, a)$ có thể được chia thành 2 phần:

- $V(s)$: giá trị của trạng thái s.
- $A(s, a)$: độ thuận lợi của việc thực hiện hành động a khi đang ở trạng thái s.

Do đó, thay vì ước lượng trực tiếp $Q(s, a)$, Dueling DQN [13], [19] sẽ chia ra ước lượng giá trị của trạng thái và độ thuận lợi của hành động. Sau đó kết hợp lại bằng lớp **aggregation layer** để có được các giá trị ước lượng Q values.



Hình 2.6: Minh họa kĩ thuật Dueling DQN

Trong cách tính Q value thông thường, chúng ta chỉ tính giá trị của trạng thái s khi thực hiện hành động a mà không thể biết được trạng thái đó là tốt hay xấu (không xét đến hành động tiếp theo).

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + (A(s, a) - \frac{1}{A} \sum_{a'} A(s, a'; \theta, \alpha)). \quad (2.10)$$

2.6 Prioritized Experience Replay [11]

Ý tưởng của Prioritized Experience Replay [13]: khi sử dụng kĩ thuật Experience Replay, có vài kinh nghiệm có thể sẽ quan trọng hơn những kinh nghiệm khác trong quá trình huấn luyện, nhưng chúng có thể không xảy ra thường xuyên. Khi chúng ta lấy 1 tập nhỏ các kinh nghiệm từ bộ nhớ một cách ngẫu nhiên, dẫn đến tình trạng những kinh nghiệm quan trọng hiếm khi được chọn để huấn luyện. Đó là lí do tại sao chúng ta cần sử dụng độ ưu tiên lên các kinh nghiệm để thay đổi xác suất lấy mẫu từ bộ nhớ.

Vậy làm thế nào để tính được độ ưu tiên của các kinh nghiệm trong bộ nhớ, và qua đó tính được xác suất được chọn ra để huấn luyện của 1 kinh nghiệm? 1 phương pháp phổ biến đó là: **rank-based** để tính độ ưu tiên cho các kinh nghiệm. Trong đó:

Độ ưu tiên (priority): của một kinh nghiệm được tính bằng độ sai số giữa giá trị dự đoán (**Q value**) và giá trị mục tiêu (**Q target**):

$$p_t = |\delta_t| = |r_t + \gamma \max_a Q(s', a; \theta) - Q(s_t, a_t; \theta)|. \quad (2.11)$$

Độ ưu tiên càng cao thì sai số δ càng cao, sai số càng cao thì càng cần được huấn luyện nhiều lần, do đó xác suất được chọn của kinh nghiệm càng cao. Xác suất được chọn của 1 kinh



Hình 2.7: Minh họa kĩ thuật Prioritized experience replay.

nghiệm i được tính theo công thức sau:

$$P_i = \frac{p_i^\tau}{\sum_k p_k^\tau}. \quad (2.12)$$

$\tau \in [0, 1]$ biểu thị cho mức độ ảnh hưởng của độ ưu tiên tới xác suất được chọn của 1 kinh nghiệm. τ càng lớn thì mức độ ảnh hưởng càng lớn.

- $\tau = 0$: P_i là xác suất đồng đều (ngẫu nhiên).
- $\tau = 1$: P_i là xác suất hoàn toàn dựa trên độ ưu tiên.

2.7 Convolution neural network

Mạng nơ-ron tích chập¹ (Convolution neural network), còn được gọi là CNNs hay ConvNets là một kỹ thuật quan trọng của lĩnh vực Học sâu (Deep Learning). Một trong các ứng dụng quan trọng của mạng nơ-ron tích chập đó là cho phép các máy tính có khả năng “nhìn” và “phân tích”, nói 1 cách dễ hiểu, Convnets được sử dụng để nhận dạng hình ảnh bằng cách đưa nó qua nhiều layer với một bộ lọc tích chập để sau cùng có được một điểm số nhận dạng đối tượng. CNN được lấy cảm hứng từ vỏ não thị giác.

CNN có 02 phần chính: **Lớp trích lọc đặc trưng của ảnh** (Conv, Relu và Pool) và **Lớp phân loại** (Fully-connected layers). Chúng tôi sẽ giới thiệu chi tiết các thành phần này ngay dưới đây.

2.7.1 Lớp tích chập (Convolutional layer)

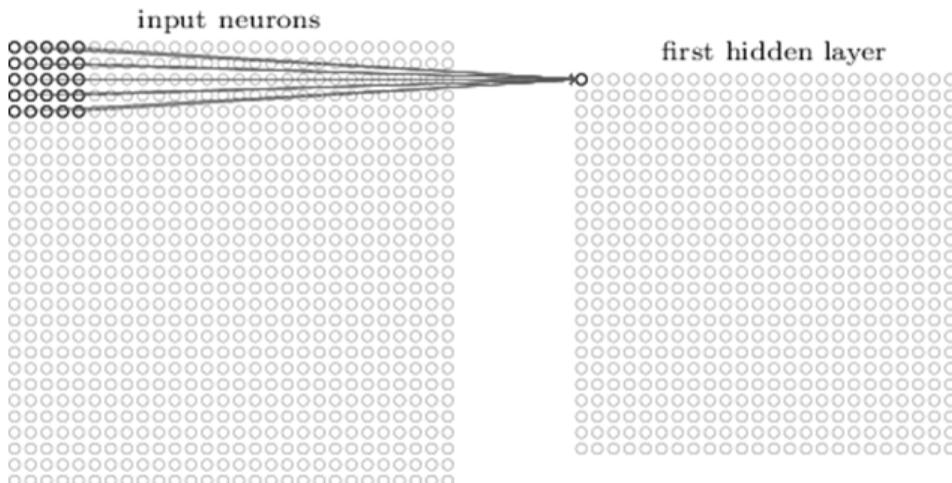
Dữ liệu đầu vào: là một bức ảnh ở dạng ma trận với kích thước [width x height x depth] tương ứng với chiều rộng, chiều cao, độ sâu. Ví dụ: 1 bức ảnh có kích thước [32x32x3] nghĩa là có chiều rộng là 32, chiều cao là 32, độ sâu là 3 (thường biểu thị cho 3 lớp màu Red, Green, Blue). **Lớp tích chập (CONV layer)** là lớp có nhiệm vụ tìm ra các **đặc trưng** của ảnh đầu vào.

Một số khái niệm cần lưu ý:

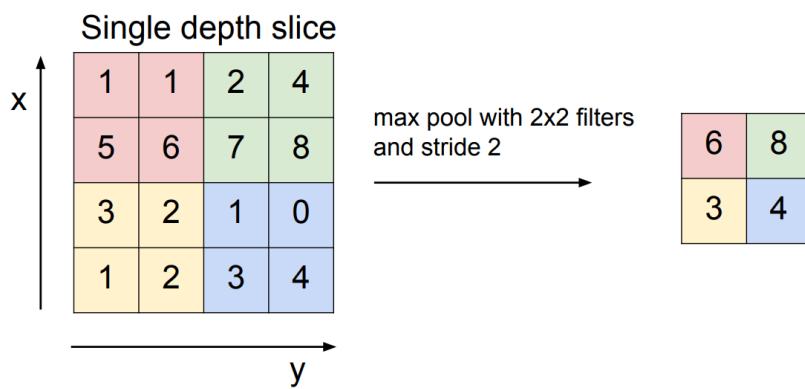
- **Bộ lọc (filter):** là một ma trận nhỏ những tham số (weights) có kích thước [w,h,d] tương ứng với chiều rộng, chiều cao, và độ sâu của filter. Lưu ý: độ sâu của filter đúng bằng với độ sâu của kích thước ảnh đầu vào.
- **Đặc trưng (feature):** của một bức ảnh là một hình ảnh mini (một mảng 2 chiều nhỏ) thể hiện đặc trưng của bức ảnh đó. Tập hợp những đặc trưng của bức ảnh là toàn bộ những đặc điểm quan trọng của nó.
- **Trường tiếp nhận (receptive field):** là một vùng nhỏ của bức ảnh được chọn để tính tích chập, có kích thước đúng bằng kích thước bộ lọc.
- **Khoảng dịch chuyển (stride) S:** là độ dịch chuyển của bộ lọc sau mỗi lần nhân tích vô hướng với trường tiếp nhận.

Bộ lọc sẽ quét qua từng mảnh của bức ảnh (trường tiếp nhận) bằng khoảng dịch chuyển S , và thực hiện phép tích vô hướng giữa ma trận **bộ lọc** và từng **trường tiếp nhận**. Đầu ra của mỗi phép tích vô hướng là một giá trị duy nhất. Do đó, đầu ra của một **phép tích chập** giữa: **ảnh đầu vào** (gồm nhiều trường tiếp nhận) và một **bộ lọc** là tập các giá trị được gọi là **mạng đặc trưng (feature map)** và cũng là đầu vào của lớp tiếp theo.

¹Tham khảo tại: <https://ereka.vn/post/chia-se-ve-mang-noron-tich-chap-convolutional-neural-networks-or-convnets-52790224348847566>



Hình 2.8: Minh họa đầu vào và đầu ra của Conv Layer [2]



Hình 2.9: Minh họa đầu vào và đầu ra của lớp pooling sử dụng max pooling

Ví dụ: Hình ảnh đầu vào có kích thước $32 \times 32 \times 3$ và bộ lọc có kích thước $5 \times 5 \times 3$. Thực hiện phép tích chập giữa hình ảnh và bộ lọc ta được ma trận kết quả: $28 \times 28 \times 1$ (ứng với 784 giá trị của 784 trường tiếp nhận của bức ảnh).

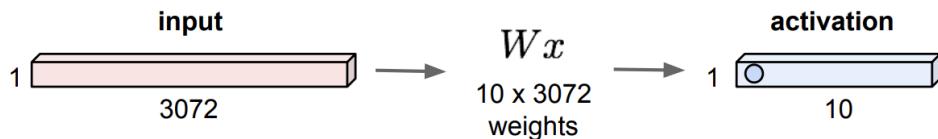
2.7.2 Lớp pooling

Một trong những lớp quan trọng khác trong CNNs là lớp pooling [3]. Đây là lớp có tác dụng thu giảm kích thước của hình ảnh và độ phức tạp tính toán của mô hình nhưng vẫn giữ được những thông tin quan trọng trong hình ảnh đó, bằng cách dịch chuyển 1 bộ lọc vuông nhỏ qua toàn bộ bức ảnh, và chỉ giữ lại giá trị lớn nhất của mỗi bộ lọc (max pooling). Lớp pooling có 2 siêu tham số: **bộ lọc (F)** và **khoảng dịch chuyển S**. Giả sử, ảnh đầu vào lớp pooling có kích thước $[W_1 \times H_1 \times D_1]$ thì ảnh đầu ra sẽ có kích thước $[W_2 \times H_2 \times D_2]$ với:

- $W_2 = \frac{W_1 - F}{S} + 1$
- $H_2 = \frac{H_1 - F}{S} + 1$
- $D_2 = D_1$

Thông thường, bộ lọc có kích thước 2×2 và khoảng dịch chuyển S là 2 sẽ đem lại hiệu quả tốt nhất.

32x32x3 image -> stretch to 3072 x 1



Hình 2.10: Minh họa cách hoạt động của lớp Fully-connected²

2.7.3 Lớp Rectified Linear Units (ReLU layer)

Lớp ReLU sử dụng hàm kích hoạt (activation function) $\max(0, x)$ lên mỗi giá trị của ma trận đầu vào, mục đích là chuyển những giá trị âm về 0. Lớp này không làm thay đổi kích thước đầu vào và không có tham số.

2.7.4 Fully-connected layers (các lớp được kết nối đầy đủ)

Fully-connected layer là lớp vô cùng quan trọng trong mạng nơ-ron tích chập. Thuật ngữ "**Fully-connected**" có nghĩa là toàn bộ nơ-ron của trước sẽ được kết nối với mỗi nơ-ron của lớp tiếp theo.

Sau khi hình ảnh đầu vào đi qua nhiều lớp trích lọc đặc trưng (Conv, ReLU, và Pool), ma trận đầu ra là một **feature map** của các đặc trưng bậc cao. Mục đích của lớp Fully-connected là sử dụng những đặc trưng đó cho việc phân loại hình ảnh đầu vào thuộc class nào.

Trước khi đưa **feature map** vào lớp này, chúng ta cần duỗi thẳng (**flatten**) ra thành một mảng 1 chiều các giá trị. Lớp đầu ra (**output layer**) sẽ có C nodes (ứng với C classes cần phân loại). Mỗi node c_i sẽ được kết nối với toàn bộ các giá trị ở trước đó và sử dụng hàm kích hoạt **softmax** lên lớp đầu ra để xác định xác suất hình ảnh ban đầu ứng với mỗi class cần phân loại, do đó tổng giá trị của lớp đầu ra luôn bằng 1.

2.7.5 Lan truyền ngược (Backpropagation)

Cho đến bây giờ, chúng ta đã có cái nhìn khá tổng quan về mạng nơ-ron tích chập. Vấn đề đặt ra tiếp theo là làm thế nào để tìm được các tham số cho bộ lọc? tham số cho mạng **Fully-connected**? Để học được các tham số này, người ta dùng giải thuật lan truyền ngược (**back propagation**) trong quá trình huấn luyện.

Quá trình huấn luyện mạng nơ-ron tích chập sẽ đi qua những bước sau:

- **Bước 1:** Khởi tạo tất cả bộ lọc và tham số những giá trị ngẫu nhiên.
- **Bước 2:** Đưa tập hình ảnh huấn luyện vào mạng qua các lớp (Conv, ReLU, Pooling, Fully-connected layer) và đầu ra là xác suất của mỗi class.
- **Bước 3:** Tính tổng sai số của kết quả đầu ra và giá trị nhãn.

$$\text{Error} = \sum (\text{target_probability} - \text{output_probability})^2$$

¹Tham khảo từ: http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf

- **Bước 4:** Sử dụng **backpropagation** để tính **gradients** ứng với tham số của mạng và sử dụng **gradient descent** để cập nhật giá trị của bộ lọc, tham số của lớp Fully-connected để cực tiểu tổng sai số.

- **Bước 5:** Lặp lại từ bước 2-4 với toàn bộ hình ảnh trong tập huấn luyện.

Do đó, nếu tập huấn luyện của chúng ta đủ lớn thì kết quả phân loại của mạng CNNs sẽ chính xác hơn.

2.7.6 Siêu tham số (Hyperparameters)

Bằng giải thuật **backpropagation**, CNNs có thể tự học được tham số cho bộ lọc và lớp Fully-connected. Tuy nhiên, còn có 1 vài **siêu tham số** khác mà người thiết kế mạng cần phải thiết lập như:

- Đối với mỗi **lớp tích chập** có bao nhiêu **đặc trưng**? Kích thước của mỗi đặc trưng là bao nhiêu?
- Đối với mỗi **lớp pooling**, kích thước cửa sổ là bao nhiêu? Bước dịch chuyển **S** là bao nhiêu?

Ngoài ra còn phải xác định xem cần bao nhiêu lớp tích chập, pooling, ReLU,...? Theo thứ tự nào?. Với rất nhiều tổ hợp và hoán vị như vậy, người thiết kế thường không thể thử nghiệm và chọn ra kiến trúc tốt nhất được. Do đó, họ thường có xu hướng sử dụng những thiết kế có hiệu suất tốt được tích lũy và chia sẻ bởi cộng đồng.

2.8 Policy Gradient

Trong thời gian gần đây, giải thuật Deep-Q-Learning (dựa trên giá trị) đã có nhiều cải tiến nhằm cải thiện hiệu năng của giải thuật như: Dueling DQN, Double DQN, Target Network,... Tuy nhiên, không phải trường hợp nào nó cũng mang lại kết quả tốt nhất. Đó là lí do tại sao chúng tôi tiếp cận **policy gradient** [20], một giải thuật dựa trên chính sách. Vậy policy gradient có những thuận lợi gì cho với Deep-Q-learning?

Thuận lợi:

1. Policy gradient có tính hội tụ tốt hơn

Vì với policy gradient chúng ta chỉ cần cập nhật tham số một cách từ từ cho đến khi hội tụ ở điểm local maximum (trường hợp không tốt) hay global maximum (trường hợp tốt nhất).

2. Policy gradient hiệu quả hơn trong không gian hành động quá lớn hay khi thực hiện những hành động liên tục.

Đối với Deep-Q-learning: dự đoán của Agent dựa trên Q value cho mỗi hành động có thể (possible action) tại mỗi bước khi biết trạng thái hiện tại. Nhưng nếu không gian hành động quá lớn sẽ như thế nào? Deep-Q-learning sẽ không thể nào tính được Q value ứng với mỗi hành động, sau đó chọn ra hành động có Q value lớn nhất được.

Ví dụ: đối với Agent là xe tự lái, thì mỗi trạng thái sẽ có gần như là vô số những hành động có thể (bẻ lái $10^\circ, 20^\circ, 30^\circ$, bóp còi, đi thẳng,). Chúng ta không thể tính được Q value ứng với mỗi hành động được.

Ngược lại, đối với policy gradient chúng ta chỉ cần cập nhật tham số cho hàm chính sách bằng gradient descent cho đến khi hội tụ.

3. Policy gradient có thể học những chính sách ngẫu nhiên.

Policy gradient có thể học chính sách ngẫu nhiên, trong khi hàm giá trị thì không thể. Có 2 thuận lợi khi policy gradient có thể học chính sách ngẫu nhiên:

- Không cần phải hiện thực cân bằng exploration/exploitation.
- Chính sách ngẫu nhiên cho phép Agent có thể khám phá không gian trạng thái mà không cần phải thực hiện cùng 1 hành động, bởi vì đầu ra của chính sách ngẫu nhiên là phân bố xác suất của các hành động. Do đó, nó có thể xử lý vấn đề cân bằng exploration/exploitation mà không cần hiện thực.

Bất lợi:

1. Thường hội tụ ở điểm local maximum thay vì global maximum.
2. Policy gradients lại hội tụ chậm, thời gian huấn luyện lâu hơn.

Trong policy gradient, chính sách được mô hình như là một hàm xấp xỉ với tham số θ , $\pi_\theta(a|s)$ với đầu ra là phân bố xác suất của các hành động, hay xác suất các hành động được chọn khi ở trạng thái s . Để đánh giá được 1 chính sách là tốt hay xấu. Chúng ta thường có 3 loại hàm mục tiêu:

1. Sử dụng giá trị của trạng thái bắt đầu.

$$J_1(\theta) = V^{\pi_\theta}(s_1) = E_{\pi_\theta}[v_1] \quad (2.13)$$

2. Trong môi trường liên tục, chúng ta có thể tính giá trị trung bình, bởi vì chúng ta không thể dựa trên 1 trạng thái cụ thể nào mà phải dựa vào chuỗi các trạng thái liên tục.

$$J_{avV}(\theta) = \sum_s d^{\pi_\theta}(s) V^{\pi_\theta}(s) \quad (2.14)$$

3. Hoặc phần thưởng trung bình ở mỗi timestep.

$$J_{avR}(\theta) = \sum_{s \in S} d^{\pi_\theta}(s) \sum_{a \in A} \pi_\theta(a|s) R_s^a \quad (2.15)$$

Trong đó, $d^\pi(s)$ là stationary distribution của chuỗi Markov cho chính sách π , biểu hiện cho xác suất để Agent đến được trạng thái s tính từ trạng thái ban đầu s_0 .

Định lý Policy Gradient: Đối với mỗi chính sách $\pi_\theta(s, a)$ với bất kì hàm mục tiêu J_1 , J_{avV} , J_{avR} nào. Thì gradient của hàm mục tiêu luôn luôn là:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a)] \quad (2.16)$$

Trong đó, $Q^{\pi_\theta}(s, a)$ là tổng phần thưởng về lâu dài dài thể hiện giá trị của hành động a .

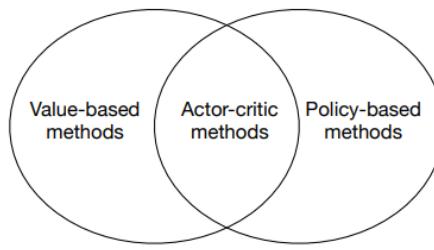
2.9 Actor-Critic [6]

Actor-critic³ là một phương pháp lai giữa **value-based** và **policy-based** bằng cách sử dụng 2 mạng nơ-ron:

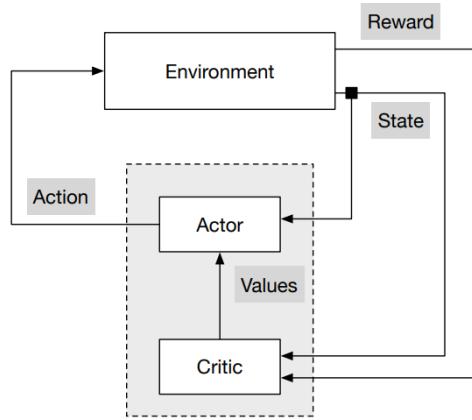
- Critic: là hàm giá trị để đo giá trị của một hành động (value-based).
- Actor: là hàm chính sách $\pi_\theta(a|s)$ điều khiển hành vi của Agent (policy-based).

Hãy tưởng tượng bạn chơi một trò chơi với một người bạn, và người bạn cung cấp cho bạn một số phản hồi. Bạn là một actor và người bạn là critic. Lúc đầu, bạn không biết chơi, nên bạn thử một số hành động ngẫu nhiên. Critic quan sát hành động của bạn và cung cấp thông tin phản hồi. Bằng cách học hỏi từ phản hồi này, bạn sẽ cập nhật chính sách của mình và chơi trò chơi đó

³Tham khảo tại: <http://mi.eng.cam.ac.uk/~mg436/LectureSlides/MLSALT7/L5.pdf>



Hình 2.11: Actor-critic kết hợp giữa value-based và policy-based



Hình 2.12: Kiến trúc tổng quan của Actor-critic

tốt hơn. Mặt khác, người bạn (critic) cũng sẽ học hỏi và cập nhật để đưa ra phản hồi cho bạn tốt hơn.

Bởi vì chúng ta có hai mô hình (Actor và Critic) phải được huấn luyện, điều đó có nghĩa là chúng ta có hai bộ trọng số (θ cho Actor và w cho Critic) phải được tối ưu hóa:

Cập nhật tham số cho Actor:

$$\Delta\theta = \alpha \nabla_{\theta} (\log(\pi_{\theta}(a|s))) Q_w(s, a) \quad (2.17)$$

Cập nhật tham số cho Critic:

$$\Delta w = \beta (R(s, a) + \gamma Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t)) \nabla_w Q_w(s_t, a_t). \quad (2.18)$$

Chương 3

Các mô hình điều khiển

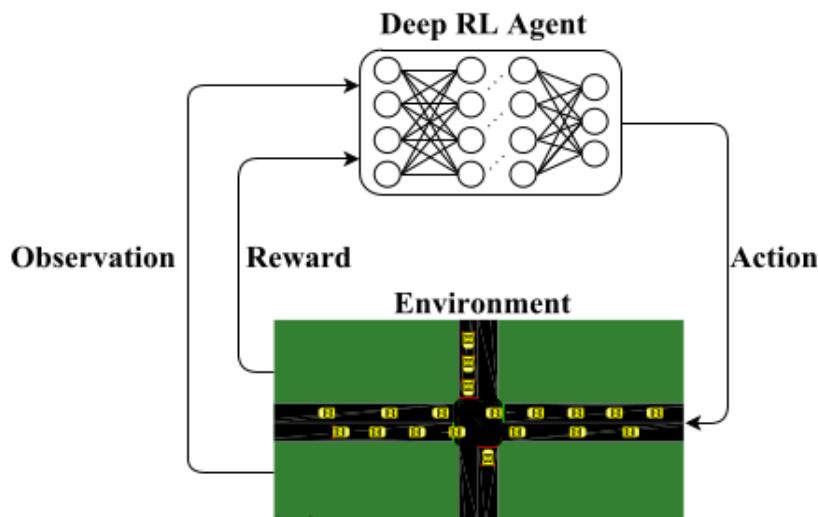
Như đã đề cập đến, chúng tôi tham khảo tới các bài báo nghiên cứu đều áp dụng kĩ thuật Reinforcement Learning để giải quyết bài toán. Vì thế, những mô hình mà chúng tôi giới thiệu sau đây gồm cách định nghĩa trạng thái, hành động và phần thưởng của từng mô hình, kèm theo đó là chi tiết về cách áp dụng những kĩ thuật khác trong Deep Learning để huấn luyện và cải thiện hiệu suất của mô hình.

3.1 Traffic Light Control Using Deep Policy-Gradient and Value-Function Based Reinforcement Learning [9]

3.1.1 Trạng thái

Để có được các trạng thái, camera sẽ được lắp đặt ở giao lộ và thực hiện chụp lại toàn bộ từ trên xuống như hình 3.1, một snapshot từ mô phỏng SUMO.

Một vector có giá trị kiểu Boolean sẽ biểu diễn cho bức ảnh được chụp lại ở thời điểm hiện tại. Ứng với mỗi giá trị trong vector thì 1 là có xe và 0 là không có xe ở vị trí đó. Và một vector khác cũng biểu diễn cho thời điểm hiện tại với kích thước tương ứng nhưng có giá trị là tốc độ của xe nếu có xe ở vị trí đó, ngược lại có giá trị 0. Trong thực tế, bằng cách giả định hằng số c là



Hình 3.1: Sơ lược kiến trúc mô hình

chiều dài của xe thường thấy nhất để biểu diễn kích thước của 1 grid trong vector, từ đó có thể dựng lên 2 vector vị trí và tốc độ như đã nói.

Mặt khác, thay vì làm như vậy, ở mô hình này, họ chọn cách đưa trực tiếp trạng thái dưới dạng hình ảnh vào CNN, hệ thống sẽ tự phát hiện vị trí của tất cả các xe với chiều dài khác nhau và cho kết quả là hàng đợi của xe ở mỗi đường.Thêm nữa, bằng cách lưu lại lịch sử của các hình ảnh được lấy về liên tiếp, deep network có thể ước lượng được vận tốc và hướng đi của xe. Và đó là cách để hệ thống hiểu được dữ liệu đầu vào từ camera.

Cụ thể khi có được ảnh chụp từ camera, ta sẽ biến đổi nó sang hình ảnh trắng đen với kích thước frame là 128 x 128. Khi lưu lịch sử các hình ảnh có thứ tự trước sau, họ chọn 4 frame cuối cùng trong lịch sử để đưa làm dữ liệu đầu vào.

3.1.2 Hành động

Hệ thống điều khiển đèn giao thông bằng cách xác định thời gian của đèn giao thông. Trước tiên, ta xác định số tuyến đường ở giao lộ để xác định số hành động có thể. Lấy ví dụ từ một ngã tư, ta sẽ có một tập hợp hành động A gồm Bắc-Nam (BN) và Đông-Tây (ĐT). Trong đó, BN là hiển thị đèn xanh ở hai đèn Bắc và Nam, cho phép chiều xe từ hai hướng này di chuyển, các chiều khác hiển thị đèn đỏ, tức là dừng lại. Đối với ĐT cũng tương tự, hiển thị đèn xanh ở hai đèn Đông và Tây, đèn đỏ ở hai đèn Bắc và Nam. Cứ sau mỗi giây, nghĩa là tăng dần thời gian t (với t ban đầu là 0), Agent sẽ chọn một hành động a_t trong tập A và các phương tiện sẽ được điều hướng bởi hành động a_t đã chọn.

3.1.3 Phần thưởng

Phần thưởng sẽ được trả về cho Agent sau khi thực hiện hành động lên môi trường. Ta định nghĩa phần thưởng r_t với $r_t \in \mathbb{R}$ là độ chênh lệch giữa hai tổng thời gian chờ D_{t-1}, D_t của các phương tiện ở bước trước và hiện tại.

$$r_t = D_{t-1} - D_t \quad (3.1)$$

Cụ thể, D_t sẽ là tổng thời gian chờ của các phương tiện xuất hiện từ thời điểm bắt đầu đếm ($t = 0$) cho đến thời điểm t . Khi phần thưởng mang giá trị dương, ta nói hành động vừa thực hiện làm giảm tổng thời gian chờ của xe. Ngược lại, với giá trị âm, hành động làm tăng tổng thời gian chờ. Vì thế, để đạt được phần thưởng cao, Agent có thể thay đổi chính sách chọn lựa hành động trong một trạng thái nhất định của hệ thống trong tương lai.

Chính sách của Agent: Agent chọn hành động dựa trên chính sách.

Trong giải thuật dựa trên chính sách, chính sách được định nghĩa như là hàm ánh xạ từ trạng thái đầu vào sang phân bố xác suất của tập hành động A . Chúng ta sử dụng deep neural network như là 1 hàm xấp xỉ và coi tham số θ của nó như là tham số của chính sách. Phân bố của chính sách $\pi(a_t | s_t, \theta)$ được học bằng cách sử dụng gradient descent lên tham số của chính sách.

Trong thuật toán dựa trên hàm giá trị (value-function), deep neural network được sử dụng để ước lượng hàm giá trị hành động (action-value function). Hàm giá trị hành động sẽ ánh xạ trạng thái đầu vào thành giá trị hành động (action-value), mỗi giá trị đại diện cho phần thưởng trong tương lai có thể đạt được từ trạng thái và hành động đã cho. Chính sách tối ưu sau đó có thể được trích xuất bằng cách thực hiện một cách tiếp cận tham lam để chọn hành động tốt nhất có thể.

3.1.4 Hàm mục tiêu và huấn luyện hệ thống

Tồn tại rất nhiều biện pháp đánh giá, mục tiêu để chúng ta hướng tới như tối đa lưu thông, tối thiểu và cân bằng chiều dài hàng đợi, tối thiểu thời gian chờ. Đây đều là mục tiêu được lấy từ tài liệu quản lý tín hiệu đèn giao thông. Và như đã nhắc đến ở mục phần thưởng, tối thiểu thời gian chờ là mục tiêu của Agent.

Để đạt được mục tiêu đó, Agent sẽ học cách cực đại hóa phần thưởng dưới chính sách phân phối $\pi(a_t|s_t; \theta)$.

$$J(\theta) = E_{\pi_\theta} \left[\sum_{t=0}^T \gamma^t r_t \right] = E_{\pi_\theta}[R] \quad (3.2)$$

Trong mô hình này, họ chia hệ thống huấn luyện với hai cách tiếp cận của học tăng cường là Value-function based và Policy-based:

- **Value-function based:** hàm giá trị $Q_\pi(s, a)$ trong phương pháp này:

$$Q_\pi(s, a) = E_\pi \left[r_t + \gamma \max_{a'} Q(s', a') | s, a \right] \quad (3.3)$$

Với $s, s' \in S$ và $a \in A$. Và gọi θ là tham số của mạn nơ-ron, cũng là tham số hóa hàm giá trị thành $Q(s, s'; \theta)$. Tham số θ sẽ được học với mục tiêu là tối thiểu hóa hàm loss (loss function) của sai số trong **Q values**. Ta viết lại hàm loss thành:

$$J(\theta) = E_\pi \left[(r + \gamma \max_{a'} Q(s', a'; \theta) - Q(s, a; \theta))^2 \right] \quad (3.4)$$

Trong đó $r + \gamma \max_{a'} Q(s', a'; \theta)$ là giá trị mục tiêu. Ta đang nói đến mạng mục tiêu (target Q-network) trong giải thuật DQN, được dùng để giải quyết vấn đề bất ổn định của chính sách. Mạng mục tiêu này có tham số θ^- được cập nhật sau mỗi bước huấn luyện bằng tham số θ của mạng chính. Giá trị mục tiêu của mô hình này được viết lại theo tham số 0–:

$$y_i = r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}^-) \quad (3.5)$$

Tham số θ của mạng chính được cập nhật lại như sau:

$$\theta_i \leftarrow \theta_{i-1} + \alpha(y_i - Q(s, a; \theta)) \nabla_{\theta_i} Q(s, a; \theta_i) \quad (3.6)$$

Với y_i là giá trị mục tiêu cho bước cập nhật thứ i và α là hệ số học (learning rate). Nói một cách tổng quát hơn về cách huấn luyện bằng phương pháp value-function based, ta biểu diễn dưới dạng mã giả được trình bày ở giải thuật [1]

- **Policy-based:** Trong cách tiếp cận này, cần tính đạo hàm của hàm mục tiêu được đề cập ở (3.2), cụ thể:

$$\nabla_{\theta} J = \sum_{t=0}^T E_{\pi_\theta} [\nabla_{\theta} \log(a_t|s_t; \theta) R_t] \quad (3.7)$$

Theo như bài báo, công thức (3.7) chính là quy tắc học chuẩn của giải thuật học tăng cường. Nó được dùng để cập nhật trực tiếp tham số θ nên xác suất của hành động a_t tại trạng thái s_t sẽ tăng khi nó có phần thưởng tích lũy cao và ngược lại. Cách tính đạo hàm này cho kết quả phương sai cao. Vì thế, để giảm phương sai, ta phải trừ đi hàm cơ bản (baseline function) $b_t(s_t)$ khỏi R_t , mà không thay đổi kì vọng. Hàm cơ bản được tính theo cách thông thường

Algorithm 1 Giải thuật Deep Value-Function based sử dụng Experience Replay cho hệ thống đèn giao thông.

Khởi tạo tham số θ ngẫu nhiên

Khởi tạo bộ nhớ M với kích thước L

for mỗi kích bản mô phỏng **do**

 Khởi tạo s là quan sát hiện tại của giao lộ

repeat

 # lần lượt các trạng thái trong mô phỏng

 Chọn hành động a dựa trên giải thuật tham lam ϵ

 Thực hiện hành động a , nhận phần thưởng r và trạng thái tiếp theo s'

 Lưu tuple (s, a, r, s') vào M

$s \leftarrow s'$

$b \leftarrow$ tập tuple được lấy ngẫu nhiên từ bộ nhớ M

for với mỗi tuple được lấy từ b **do**

if s'_j là trạng thái kết thúc **then**

$y_i \leftarrow r_j$

else

$y_i = r_j + \gamma \max_a' Q(s'_j, a'; \theta_{i-1}^-)$

end

 Cập nhật tham số θ bằng công thức (3.6)

end

until s là trạng thái kết thúc;

end

là bằng kì vọng của hàm giá trị trạng thái, $b_t(s_t) = V^{\pi_{\theta_v}}(s_t)$. Ta viết lại công thức đạo hàm (3.7) thành:

$$\nabla_{\theta} J = \sum_{t=0}^T E_{\pi_{\theta}} [\nabla_{\theta} \log(a_t | s_t; \theta) (R_t - b_t(s_t))] \quad (3.8)$$

Trong đó, $R_t - b_t$ được xem là hàm thuận lợi (advantage function). Bằng cách áp dụng phương pháp actor-critic, với tập mẫu gồm M mẫu và Agent sẽ chọn hành động cho mỗi mẫu đó cho đến khi trạng thái kết thúc. Agent sẽ nhận được $n \leq M$ phần thưởng tương ứng với n hành động được thực hiện và được cập nhật lần lượt cho tham số θ . Cụ thể, việc cập nhật được biểu diễn dưới công thức sau:

$$\theta = \theta + \alpha \sum_t \nabla_{\theta} \log(a_t | s_t; \theta) A(s_t, a_t; \theta, \theta_v) \quad (3.9)$$

$A(s_t, a_t; \theta, \theta_v)$ chính là kì vọng của hàm thuận lợi được tính bằng công thức:

$$A(s_t, a_t; \theta, \theta_v) = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n V(s_{t+n}; \theta) - V(s_t; \theta_v) \quad (3.10)$$

Mã giả ở mục 2 sẽ là tổng quan về giải thuật huấn luyện với hướng tiếp cận là policy-gradient based.

3.1.5 Kiến trúc tổng quan

Mô hình trong bài báo này được họ huấn luyện bằng mô phỏng SUMO. Như đã giới thiệu, dữ liệu đầu vào của mạng DQN sẽ là 4 ảnh có kích thước 128 x 128. Mạng DQN của mô hình

Algorithm 2 Giải thuật Deep Policy-Gradient based sử dụng Experience Replay cho hệ thống đèn giao thông.

```

Khởi tạo tham số  $\theta, \theta_v$  ngẫu nhiên
Khởi tạo biến đếm  $t \leftarrow 0$ 
for mỗi kích bản mô phỏng do
    Khởi tạo  $s$  là quan sát hiện tại của giao lộ
     $t_{start} = t$  repeat
        Thực hiện hành động  $a$  dựa trên chính sách  $\pi(a|s; \theta)$ , nhận phần thưởng  $r$  và trạng thái tiếp theo  $s'$ 
         $t \leftarrow t + 1$ 

        until  $s$  là trạng thái kết thúc or  $t - t_{start} == M$  (Max step);
        if  $s$  là trạng thái kết thúc then
            |  $R = 0$ 
        else
            |  $R = V(s; \theta_v)$ 
        end
        for  $i \in \{t - 1, \dots, t_{start}\}$  do
            |  $n \leq M$  lần
            |  $R \leftarrow r_i + \gamma R$ 
            |  $\theta \leftarrow \theta + \alpha \nabla_{\theta} \log(a_i|s_i; \theta) (R - V(s_i; \theta_v))$ 
            |  $\theta_v \leftarrow \theta_v + \frac{\partial(R - V(s_i; \theta_v))^2}{\partial \theta_v}$ 
        end
    end

```

này gồm 2 **convolutional layer**. Với layer đầu có 16 filter 8×8 , **stride** là 4 và layer thứ hai có 32 filter 4×4 , **stride** là 2.

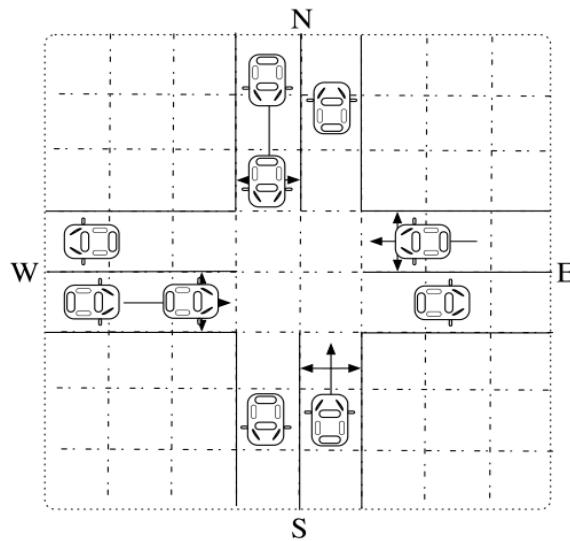
- Đối với mô hình policy-based, layer cuối cùng là lớp **Fully-connected** với 256 **hidden node**. Đầu ra của mạng DQN sẽ được tính softmax cho kết quả là xác suất phân phối trên tập hành động A và một nốt tuyển tính cho ra kết quả là kì vọng của hàm giá trị trạng thái $V(s)$.
- Đối với mô hình value-function, đầu ra của layer cuối trong mạng nơ-ron là giá trị hành động.

Tất cả trọng số của mạng nơ-ron được cập nhật bằng giải thuật tối ưu Adam. Và đó là toàn bộ hai kiến trúc mô hình mà bài báo đề cập đến với hai hướng tiếp cận khác nhau. Ở phần tiếp theo chúng tôi sẽ trình bày là một mô hình khác theo hướng tiếp cận là value-based nhưng có một mức cải tiến hơn với sự áp dụng của nhiều kĩ thuật tiên tiến khác trong Deep Learning.

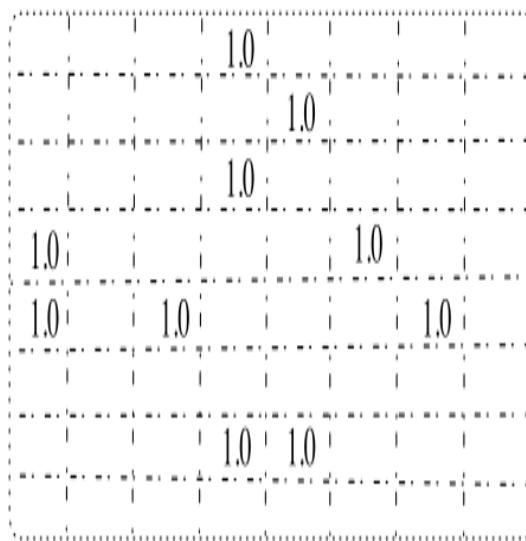
3.2 Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks [8]

3.2.1 Trạng thái

Trong mô hình này, một trạng thái (state) phải thể hiện được 2 thông tin của một giao lộ giao thông là: **vị trí** và **tốc độ** của xe. Thông tin này được lấy từ các cảm biến, camera gắn trên giao



Hình 3.2: Snapshot từ một ngã tư vào một thời điểm



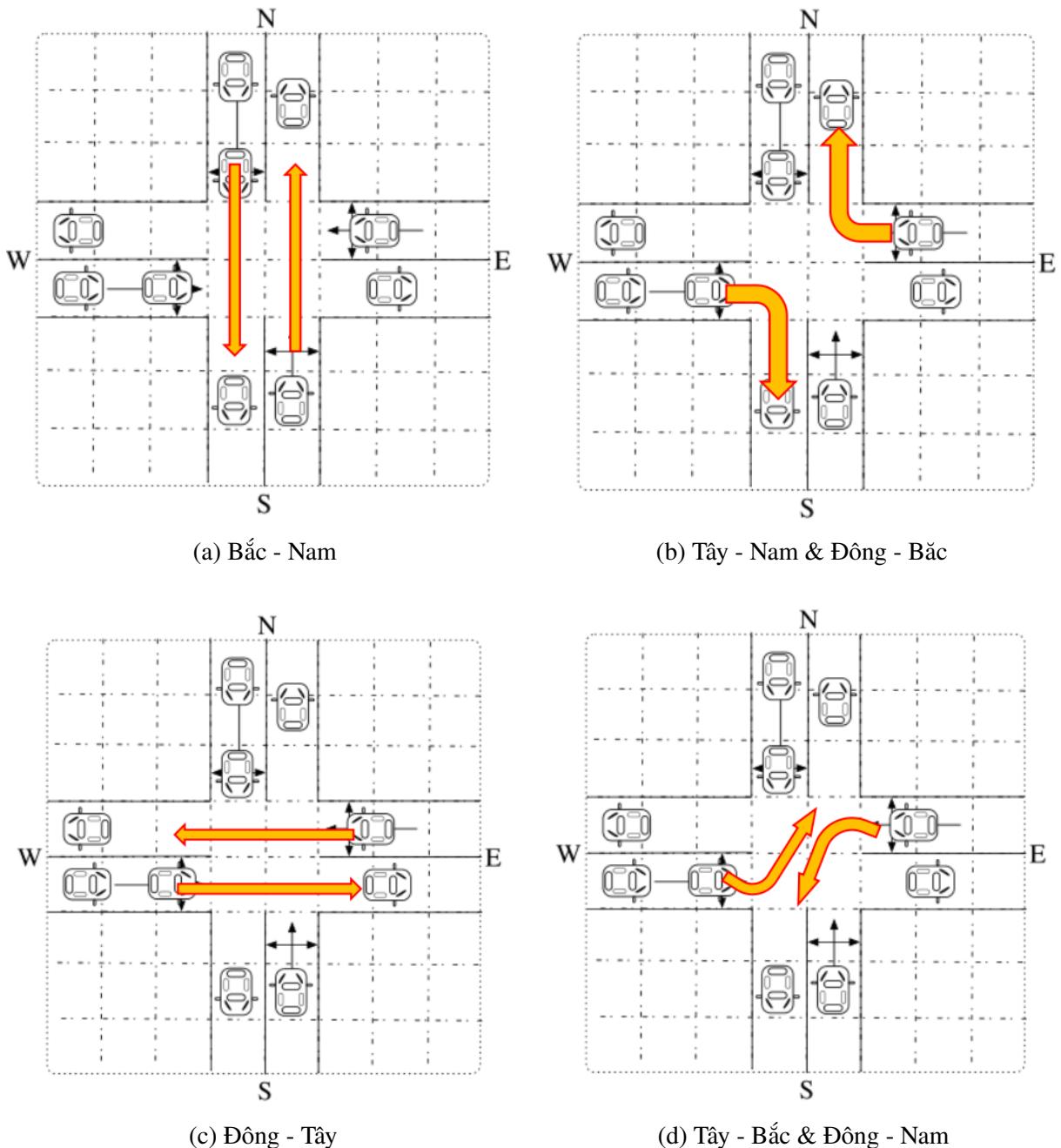
Hình 3.3: Ma trận biểu diễn vị trí xe của ngã tư

lộ và được chuyển đổi sang dạng ma trận.

Cụ thể, với một hình ảnh giao thông (Hình 3.2), ta chia thành các grid với kích thước bằng nhau có độ dài c . Mỗi grid là một vector 2 chiều $\langle position, speed \rangle$ biểu thị sự có mặt của xe và tốc độ của xe tại grid đó. Đơn vị *position* được biểu diễn dưới dạng binary. Nếu có xe tại grid đó, *position* là 1, ngược lại là 0 (Hình 3.3). Đơn vị của *speed* là m/s và được biểu diễn dưới dạng số nguyên. Nếu có xe trong grid, *speed* là tốc độ hiện tại của xe đó, ngược lại là 0.

Một điều lưu ý rằng, độ dài c được chọn sao cho một khung grid chỉ vừa với một chiếc xe, ta sẽ chọn theo một xe thường thấy nhất, điều này hiển nhiên sẽ làm giảm sự tính toán.

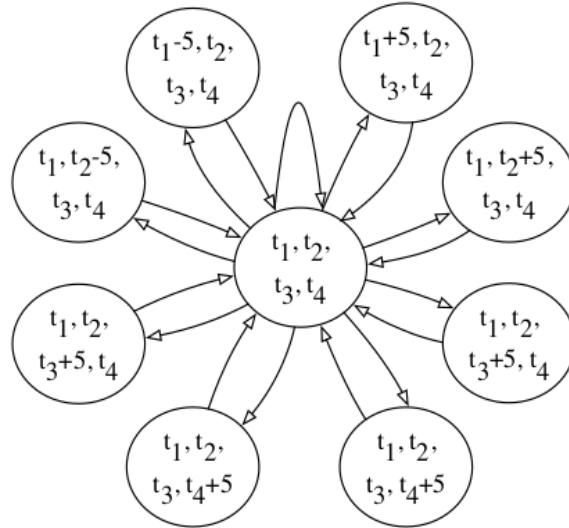
Sau khi map thành công hình ảnh giao thông của một giao lộ ta sẽ được ma trận các grids có dạng: $m \times n \times 2$ thể hiện thông tin gồm vị trí (*position*) và tốc độ (*speed*) của xe.

Hình 3.4: Các **phase** của một chu kì

3.2.2 Hành động

Nói đến hành động, ta quan sát hệ thống giao thông hiện nay, đèn giao thông điều hướng xe cộ ở các giao lộ bằng thời gian ở mỗi trạng thái của đèn (đỏ, vàng, xanh). Dựa vào đó, ta định nghĩa một hành động là đặt ra các khoảng thời gian của mỗi trạng thái giao lộ vào chu kỳ tiếp theo.

Để dễ hiểu hơn, ta tạm gọi một trạng thái đèn giao thông là một **status**. Một **chu kỳ** là vòng liên tiếp của các status. Thời gian của một **status** là một **phase**. Ở mỗi giao lộ khác nhau sẽ có số hành động khác nhau, vì thế mô hình mà chúng tôi đề cập đến chỉ áp dụng cho một giao lộ được huấn luyện theo. Lấy hình 3.2 làm ví dụ, ta định ra một ngã tư có 4 **status** tất cả (số **status** có thể khác tùy vào luật giao thông ở mỗi nước).



Hình 3.5: Một ví dụ về không gian hành động trên giao lộ Hình 3.2

Một tuple $< t_1, t_2, t_3, t_4 >$ biểu diễn chuỗi các **phase** của một chu kỳ cho 4 status trên. Trong đó, t_i là thời gian diễn ra status thứ i (thời gian đèn xanh của status, các status còn lại chưa diễn ra có đèn đỏ). Sở dĩ ta chỉ xét đèn xanh vì các status nối tiếp nhau trong một chu kỳ. Việc định ra một tuple cho chu kỳ tiếp theo chính là hành động của mô hình.

Dựa trên bài báo đang theo dõi, nếu thời gian chênh lệch lớn giữa hai chu kỳ, hệ thống sẽ trở nên không ổn định. Vì thế việc đặt ra một chính sách trong việc chọn một hành động với thời gian thay đổi một cách trơn tru là cần thiết. Mô hình MDP được áp dụng để giải quyết vấn đề này. Cụ thể trong mô hình này, tác giả chỉ cho phép một và chỉ một **phase** của chu kỳ tiếp theo là tăng hoặc giảm 5s so với chu kỳ trước.

Hệ thống có thể chọn lại cùng một hành động và thời gian tối đa của một **phase** là 60 giây, tối thiểu là 0 giây. Mô hình MDP là một mô hình linh hoạt, có thể áp dụng cho mọi giao lộ khác. Đối với những giao lộ phức tạp hơn (ngã năm, ngã sáu), mô hình MDP sẽ có nhiều hành động hơn. Có thể thấy rõ ràng, một hành động được biểu diễn bằng 1 vector n chiều, với n là số **phase** (hay số status) tại giao lộ đó.

Mỗi **phase** lần lượt thay phiên nhau trong một chu kỳ. Để đảm bảo tính an toàn, ra hiệu cho xe dừng lại trước tín hiệu đèn đỏ, hệ thống sử dụng đèn vàng. Với thời gian đèn vàng T_{yellow} được tính bằng cách lấy tốc độ tối đa v_{max} của xe trên đường chia cho gia tốc giảm a_{dec} thường thấy nhất.

$$T_{yellow} = \frac{v_{max}}{a_{dec}} \quad (3.11)$$

Công thức (3.11) mô tả rằng các phương tiện đang di chuyển có một khoảng thời gian để giảm tốc cho đến khi dừng lại.

Nói một cách ngắn gọn, hành động của mô hình này là đặt ra thời gian của các **phase**. Trình tự các **phase** trong một chu kỳ là không thay đổi để đảm bảo tính ổn định và an toàn của hệ thống.

3.2.3 Phản thưởng

Vai trò của phản thưởng trong giải thuật học tăng cường là thể hiện sự phản hồi của môi trường về hành động mà Agent vừa thực hiện, dựa vào độ lớn của phản thưởng mà Agent có thể

biết được hành động đó là tốt hay xấu. Do đó, việc định nghĩa hàm phần thưởng là cực kì quan trọng vì Agent sẽ dựa vào nó để học trong quá trình huấn luyện.

Với mục tiêu của hệ thống là cải thiện tình hình giao thông ở giao lộ, cụ thể là giảm thiểu thời gian chờ của các phương tiện đi qua giao lộ. Phần thưởng được định nghĩa là sự sai lệch giữa **tổng thời gian chờ của các xe** ở trạng thái trước và trạng thái sau khi thực hiện hành động (giữa 2 chu kỳ liên tiếp, mỗi chu kỳ được tính từ thời điểm $t = 0$). Nói một cách rõ ràng hơn, tạm gọi i_t là phương tiện thứ i trong giao lộ của chu kỳ t . Thời gian chờ của xe thứ i ở chu kỳ t đặt là $w_{i_t,t}$ ($1 \leq i_t \leq N_t$), với N_t là số lượng xe có trong giao lộ ở chu kỳ t tính từ thời điểm $t = 0$. Phần thưởng được biểu diễn dưới dạng công thức như sau:

$$r_t = W_t - W_{t+1} \quad (3.12)$$

với

$$W_t = \sum_{i_t=1}^{N_t} w_{i_t,t} \quad (3.13)$$

Việc mô hình hóa này thể hiện rõ ràng phần thưởng sau khi thực hiện hành động. Tổng thời gian chờ ở chu kỳ sau (W_{t+1}) luôn lớn hơn hoặc bằng tổng thời gian chờ ở chu kỳ trước (W_t). Xe ở chu kỳ sau chờ càng lâu, phần thưởng r_t sẽ có giá trị càng âm, lúc này ta biết hành động vừa rồi không tốt và ngược lại. Như vậy, khi phần thưởng bằng 0 ($r_t = 0$) nghĩa là chu kỳ sau không có xe nào phải đợi và là phần thưởng lớn nhất mà Agent nhận được và tổng thời gian chờ của các xe không giảm theo thời gian, phần thưởng tổng thể sẽ luôn luôn âm

3.2.4 Những kỹ thuật được áp dụng

Đối với bài toán điều khiển đèn giao thông này, số lượng trạng thái là vô cùng lớn. Do đó, không thể lập bảng **Q-table** bằng giải thuật **Q-learning**. Để có 1 hàm θ xấp xỉ giá trị **Q value**, chúng ta sẽ sử dụng 1 mạng nơ ron (Q network) với đầu vào là trạng thái hiện tại và đầu ra là mảng các Q values tương ứng với các hành động có thể. Đồng thời, để cải thiện hiệu suất của network chúng ta sẽ sử dụng thêm các kỹ thuật sau: **Convolutional Neural Network**, **Experience Replay**, **Target Network**, và những kỹ thuật mới nhất của học tăng cường như: **Dueling Network**, **Double Q-learning Network**...

1. Convolutional Neural Network trong mô hình

Để Agent có thể hiểu được dữ liệu đầu vào (trạng thái hiện tại của giao lộ), mô hình này dùng sẽ có 3 lớp tích chập. Thay vì xem xét các pixel của dữ liệu vào một cách độc lập, các lớp tích chập sẽ trích xuất ra các đặc trưng của dữ liệu vào mà vẫn duy trì được mối quan hệ trong không gian giữa các đối tượng. Sau đó, qua một số lớp **Fully-connected** để xấp xỉ được Q values của các hành động. Dưới đây là kiến trúc tổng thể của mạng tích chập:

Trong kiến trúc này, mạng tích chập bao gồm: 3 lớp tích chập và một số lớp **Fully-connected**. Dữ liệu vào (input) là ma trận các grids thể hiện 2 thông tin: **vị trí** (position) và **tốc độ** (speed) của các phương tiện tại giao lộ đó. Giả sử **dữ liệu vào** có dạng ma trận: $60 \times 60 \times 2$. Dữ liệu vào ban đầu được đưa qua 3 lớp tích chập. Mỗi lớp tích chập bao gồm 3 thành phần: **convolution**, **pooling** và **activation**

Mỗi lớp tích chập: bao gồm nhiều filters. Mỗi filter là tập các trọng số (weights) và thực hiện phép tích vô hướng với từng mảnh (local patch) của lớp trước đó.

Lớp pooling sẽ chọn ra giá trị lớn nhất của mỗi cửa sổ nhỏ (local patch) để thay thế cho toàn bộ cửa sổ đó. Qua đó, quá trình pooling sẽ xóa đi những thông tin ít quan trọng và giảm đi kích thước của dữ liệu vào.

Hàm kích hoạt là hàm để quyết định đơn vị (unit) nào sẽ được kích hoạt. Trong mô hình này, chúng ta sử dụng hàm kích hoạt không tuyến tính lên output đó là: **leaky ReLU**, có dạng như sau (giả sử x là kết quả output của 1 đơn vị (unit)):

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \beta x, & \text{if } x \leq 0 \end{cases}$$

Hàm Leaky ReLU có thể hội tụ nhanh hơn những hàm kích hoạt khác như: **tanh** hay **sigmoid**.

Cụ thể, trong kiến trúc mạng tích chập này:

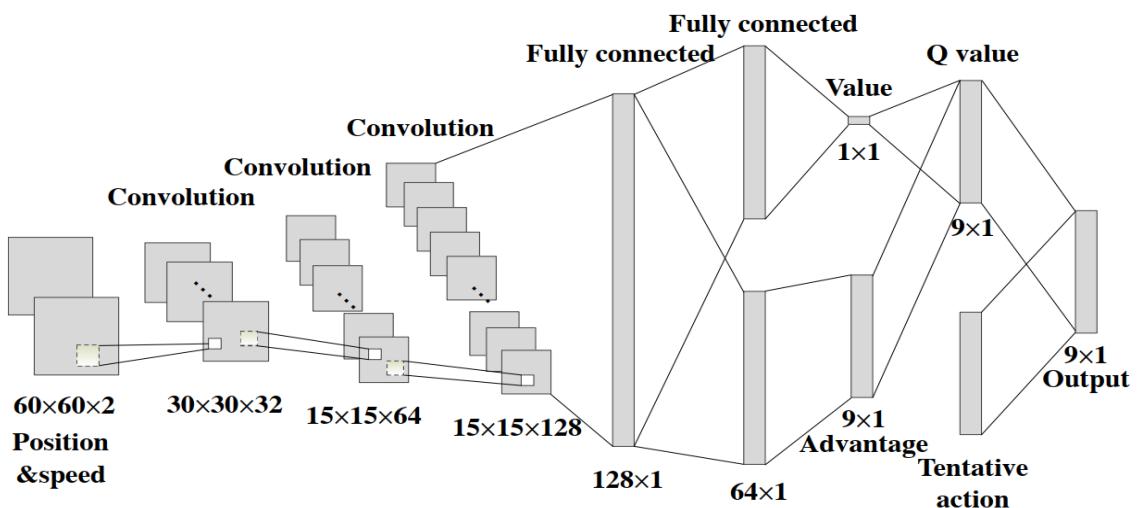
- **Lớp tích chập thứ 1:** có 32 filtets, mỗi filter có kích thước: 4×4 , khoảng dịch chuyển (stride) là 2×2 . Kích thước đầu ra là: $30 \times 30 \times 32$.
- **Lớp tích chập thứ 2:** có 64 filtets, mỗi filter có kích thước: 2×2 , khoảng dịch chuyển (stride) là 2×2 . Kích thước của đầu ra là: $15 \times 15 \times 64$.
- **Lớp tích chập thứ 3:** có 128 filtets, mỗi filter có kích thước: 2×2 , khoảng dịch chuyển (stride) là 1×1 . Kích thước của đầu ra là: $15 \times 15 \times 128$.

Sau khi qua lớp tích chập thứ 3, thực hiện flatten chuyển ma trận đầu ra thành ma trận 128×1 . Tiếp đến đi qua lớp **Fully-connected** hai lần để tạo ra 2 ma trận với kích thước bằng nhau: 64×1 . **Phần thứ 1** sẽ được sử dụng đánh tính giá trị của dữ liệu đầu vào (trạng thái). **Phần thứ 2** sẽ được sử dụng để tính độ thuận lợi của mỗi hành động, đầu ra là ma trận 9×1 . Độ thuận lợi của một hành động nghĩa là mức độ lợi ích của việc chọn hành động này so với các hành động khác. Sau đó, kết hợp kết quả đầu ra của 2 phần ta được **Q value** của mỗi hành động (ma trận 9×1).

Với các giá trị **Q value** tương ứng với các hành động, chúng ta sẽ kết hợp chúng với hành động dự kiến (tentative actions) để chắc chắn rằng hệ thống của chúng ta sẽ luôn chọn những hành động hợp lệ. Như vậy, từ dữ liệu đầu vào là trạng thái của giao lô, ta tính được các giá trị **Q value** tương ứng với các hành động hợp lệ. Gọi θ là tham số của mạng nơ ron tích chập, $Q(s,a)$ được ghi lại thành $Q(s,a;\theta)$.

2. Dueling DQN

Trong network 3.6 đã nêu ở phần trước, dueling DQN được áp dụng để tính **Q value** từ



Hình 3.6: Kiến trúc của mạng CNN để xấp xỉ Q value

hai ma trận **Value** và **Advantage**. Ta kí hiệu **V value** của trạng thái là $V(s; \theta)$, độ thuận lợi (Advantage) của hành động kí hiệu là $A(s, a; \theta)$. **Q value** được tính theo công thức sau:

$$Q(s, a; \theta) = V(s; \theta) + (A(s, a; \theta) - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta)) \quad (3.14)$$

$Q(s, a; \theta)$ bằng tổng của **V value** và giá trị thuận lợi **A value** của một hành động, với **A value** bằng hiệu số $A(s, a; \theta)$ và trung bình giá trị thuận lợi của các hành động. Nếu **A value** là mang giá trị dương, đồng nghĩa với việc hiệu quả khi thực hiện hành động đó tốt hơn hiệu quả trung bình các hành động và ngược lại. Theo bài báo, ta sử dụng **A value** thay vì công trực tiếp $A(s, a; \theta)$ giúp mô hình có tính ổn định hơn khi huấn luyện. Việc áp dụng kiến trúc dueling như trên nhằm cải thiện hiệu quả trong việc học tăng cường.

3. Target Network

Target network là 1 kĩ thuật để cải thiện hiệu suất trong quá trình huấn luyện bằng cách tạo thêm 1 network mới giống như primary network để sinh ra Q target.

Do đó, nó cũng được áp dụng để nâng cao hiệu quả cho mô hình.

4. Double DQN

Mô hình này cũng áp dụng giải thuật double Q-learning nhằm cải thiện hiệu suất bằng cách dùng primary network để chọn hành động và target network để tính ra Q target cho hành động đó.

$$Q_{target}(s, a) = r + \gamma Q(s', \arg \max_{a'} (Q(s', a'; \theta)); \theta^-) \quad (3.15)$$

Ngoài ra, giải thuật ϵ -greedy được áp dụng trong mô hình trong việc chọn ra một hành động với trạng thái cho trước. Khi mới bắt đầu huấn luyện, ta không dựa trên network để chọn ra một hành động, thay vào đó là chọn ngẫu nhiên cho đến khi mô hình đã được huấn luyện một khoảng thời gian rồi mới chọn hành động dựa trên network. Cụ thể với giải thuật ϵ -greedy, sau mỗi bước cập nhật trọng số primary network, giá trị ϵ giảm dần cho đến một giá trị giới hạn. Và đến khi ϵ đạt được giới hạn, hành động sẽ được chọn từ network.

5. Prioritized Experience Replay

Trong quá trình huấn luyện, có vài kinh nghiệm có thể sẽ quan trọng hơn những kinh nghiệm khác, nhưng chúng có thể không xảy ra thường xuyên. Khi chúng ta lấy 1 tập nhỏ các kinh nghiệm từ bộ đệm một cách ngẫu nhiên, dẫn đến tình trạng những kinh nghiệm quan trọng hiếm khi được chọn để huấn luyện. Do đó, áp dụng kĩ thuật Prioritized Experience Replay với phương pháp rank-based sẽ giải quyết được vấn đề này.

6. Tối ưu hóa

Đây là một trong những vấn đề quan trọng của bài toán, làm thế nào để cập nhật tham số θ của primary network? Theo bài báo mà chúng tôi đang theo dõi, giải thuật Adaptive Moment (Adam), một trong những giải thuật back propagation được áp dụng để giải quyết vấn đề này. Khi so sánh với các giải thuật back propagation khác, Adam đạt được hiệu suất tổng thể với sự hội tụ và tỷ lệ học tập thích ứng nhanh.

Trong phương pháp của Adam, bước đầu tiên là cập nhật hệ số học (learning rate) xem là first-order moment và second-order moment bằng phương pháp đạo hàm. Cụ thể, với θ là tham số của primary network thì $J(\theta)$ biểu diễn hàm loss (loss function ??). Ta định đạo hàm loss function theo θ ,

$$\mathbf{g} = \nabla_{\theta} J(\theta) \quad (3.16)$$

Bước tiếp theo là cập nhật first-order biased moment \mathbf{s} và second-order biased moment \mathbf{r} , cụ thể như sau:

$$\begin{aligned}\mathbf{s} &= \rho_s \mathbf{s} + (1 - \rho_s) \mathbf{g}, \\ \mathbf{r} &= \rho_r \mathbf{r} + (1 - \rho_r) \mathbf{g}.\end{aligned}\quad (3.17)$$

Trong đó, ρ_s và ρ_r là các tỷ lệ phân rã theo cấp số nhân (exponential decay rates). First-order biased moment \mathbf{s} và second-order biased moment \mathbf{r} sau đó được chỉnh lại cho chính xác dựa theo bước lặp thứ t , cụ thể:

$$\begin{aligned}\hat{\mathbf{s}} &= \frac{\mathbf{s}}{1 - \rho_s^t}, \\ \hat{\mathbf{r}} &= \frac{\mathbf{r}}{1 - \rho_r^t}.\end{aligned}\quad (3.18)$$

Cuối cùng, $\hat{\mathbf{s}}$ và $\hat{\mathbf{r}}$ được dùng để cập nhật tham số θ như sau:

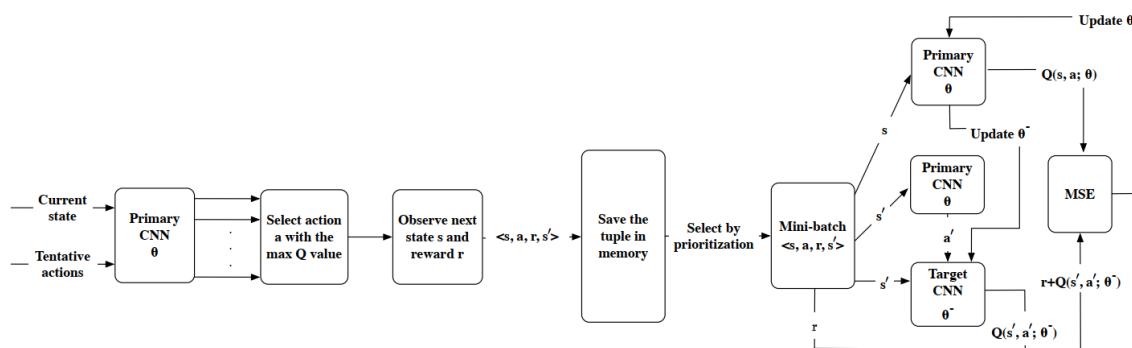
$$\begin{aligned}\theta &= \theta + \Delta\theta \\ &= \theta + (-\varepsilon_r \frac{\hat{\mathbf{s}}}{\sqrt{\hat{\mathbf{r}} + \delta}})\end{aligned}$$

Với ε_r là hệ số học khởi tạo (initial learning rate) và δ là một hằng số dương nhỏ để đạt được sự ổn định số học.

3.2.5 Kiến trúc tổng quan

Tổng kết, toàn bộ quá trình xử lý của mô hình được mô tả trong hình 3.7.

Ban đầu, trạng thái hiện tại s và các hành động dự kiến sẽ được đưa vào mạng CNN trong **primary network** để chọn ra hành động a hợp lệ có **Q value** tốt nhất. Agent thực hiện hành động a và nhận được trạng thái kế tiếp s' và phần thưởng r . Lưu kinh nghiệm dưới dạng tuple $\langle s, a, s', r \rangle$ vào bộ nhớ. Khi bộ nhớ đạt đủ số lượng kinh nghiệm cần thiết, chúng ta bắt đầu lấy từng tập nhỏ các mẫu kinh nghiệm ra theo kĩ thuật **prioritized experience replay** để huấn luyện, cụ thể là cập nhật tham số θ cho **primary network**. Đồng thời kết hợp hai kĩ thuật **double DQN** và **dueling DQN** để cải thiện hiệu suất của mô hình. Bằng cách này, **primary network** sẽ được huấn luyện cho đến khi có thể tính được giá trị xấp xỉ **Q value** ứng với mỗi



Hình 3.7: Kiến trúc tổng quan của mô hình.

Algorithm 3 Giải thuật Dueling Double Deep Q Network sử dụng Prioritized Experience Replay cho hệ thống đèn giao thông.

Khởi tạo tham số θ, θ^- ngẫu nhiên
 Khởi tạo bộ nhớ m rỗng và $i = 0$
 Khởi tạo s là trạng thái bắt đầu tại giao lộ.
while s không phải là trạng thái kết thúc **do**

- Chọn 1 hành động a thông qua giải thuật tham lam ϵ
- Thực hiện hành động a , nhận được phần thưởng r và trạng thái kế tiếp s'
- if** kích thước của bộ nhớ $m > M$ **then**
- | Xóa những kinh nghiệm cũ nhất trong bộ nhớ
- end**
- Thêm kinh nghiệm $< s, a, r, s' >$ vào bộ nhớ M
- Gán s' cho s : $s \leftarrow s'$
- $i \leftarrow i + 1$
- if** $|M| > B$ and $i > tp$ **then**
- | Chọn ra B mẫu kinh nghiệm từ m dựa trên độ ưu tiên
- | Tính độ mất mát J :
- |
$$J = \sum_s \frac{1}{B} [r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)]^2$$
- | Cập nhật tham số θ bằng ∇J trong lan truyền ngược Adam
- | Cập nhật tham số θ^- bằng phương trình: $\theta^- = \alpha \theta^- + (1 - \alpha) \theta$
- | Cập nhật độ ưu tiên của các mẫu kinh nghiệm trong bộ nhớ dựa trên δ
- | Cập nhật giá trị của ϵ
- end**

end

hành động. Cuối cùng, khi **primary network** nhận vào một trạng thái mới, nó sẽ tính toán và chọn ra hành động có **Q value** lớn nhất để thực hiện, đây chính là policy cuối cùng đạt được.

Giải thuật [3] trình bày dưới dạng mã giả về tổng quan việc khởi tạo tham số và huấn luyện mạng 3DQN của mô hình này. Mục tiêu đặt ra là huấn luyện cho hệ thống điều khiển đèn giao thông ở giao lộ bằng cách thay đổi **phase** dựa trên tình hình giao thông. Giải thuật này nói rằng, Agent ban đầu chọn ra hành động a một cách ngẫu nhiên. Qua các bước lặp cho đến khi thỏa hai điều kiện, thứ nhất là số bước lặp i lớn hơn ngưỡng bắt đầu huấn luyện pt (pre-train) và thứ hai là bộ nhớ có số tuple lớn hơn số lượng mẫu cần lấy ra để huấn luyện (ít nhất 1 mẫu). Trước khi huấn luyện, độ ưu tiên của các mẫu là như nhau. Và qua mỗi bước huấn luyện, độ ưu tiên sẽ thay đổi. Tham số trong mạng nơ-ron được cập nhật lại bằng kĩ thuật lan truyền ngược Adam. Sau khi cập nhật xong các tham số, kể cả ϵ và độ ưu tiên của các mẫu. Khi ϵ đạt đến một ngưỡng cho trước, theo như giải thuật ϵ , Agent bắt đầu chọn hành động có **Q value** cao nhất. Cuối cùng, kết thúc vòng lặp, chúng ta có một Agent có thể lấy được phần thưởng cao qua việc huấn luyện lại với một kịch bản giao thông khác cùng loại giao lộ.

Chương 4

Phân tích và lựa chọn mô hình

Dựa trên những tìm hiểu về mô hình ở hai bài báo **DeTLC** và **DePGVF** (tạm gọi là **mô hình 1** và **mô hình 2**), trong phần này chúng tôi sẽ nêu ra sự so sánh, đánh giá ưu và nhược điểm của mỗi mô hình, từ đó đưa ra các mô hình mà chúng tôi sẽ hiện thực. Các mô hình sẽ được hiện thực là tổ hợp của 4 phần chính gồm trạng thái, hành động, phần thưởng và mạng nơ-ron.

4.1 Trạng thái

Mô hình 1 biểu diễn trạng thái dưới dạng ma trận có kích thước $60 \times 60 \times 2$ thể hiện thông tin vị trí và tốc độ của xe, trong khi đó **mô hình 2** sử dụng hình ảnh được chụp trực tiếp từ mô phỏng SUMO.

Có thể thấy cách xây dựng trạng thái của **mô hình 2** là rất đơn giản, vì chúng tôi chỉ cần sử dụng APIs của thư viện Traci [17] cung cấp để chụp ảnh tình trạng giao thông trong mô phỏng, tuy nhiên việc huấn luyện Agent bằng hình ảnh sẽ tốn rất nhiều thời gian để mạng CNN có thể trích xuất được các đặc trưng của ảnh, đồng nghĩa với việc tốn nhiều thời gian để huấn luyện Agent.

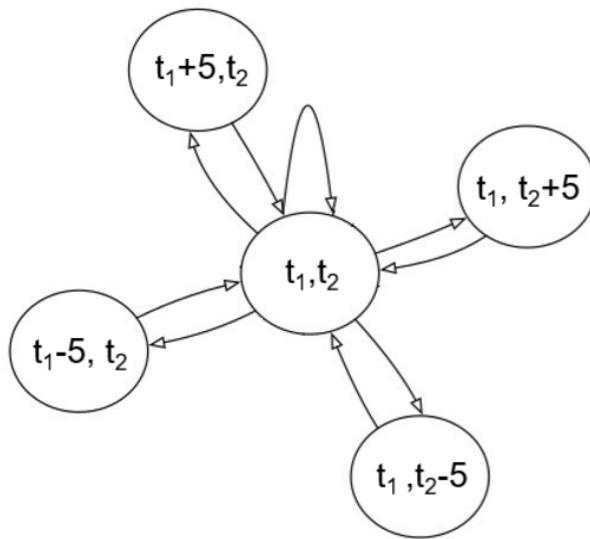
Ngược lại, với cách mô hình trạng thái của **mô hình 1**, chúng tôi sẽ sử dụng các APIs mà Traci cung cấp và lấy được thông tin cần thiết (vị trí, tốc độ,...) để xây dựng ma trận đầu vào, đồng thời cũng loại bỏ những vị trí không có xe trên bản đồ bằng giá trị 0. Cách xây dựng này tốn nhiều chi phí tính toán để xây dựng ma trận trạng thái nhưng Agent có thể nhận biết được vận tốc, hướng đi của xe một cách dễ dàng nhờ tính đơn giản của dữ liệu đầu vào. Vì thế chúng tôi chọn mô hình trạng thái theo **mô hình 1**.

4.2 Hành động

Hành động của Agent được đánh giá dựa trên sự linh hoạt giúp Agent có thể thấy rõ hiệu quả của hành động đã quyết định. Trước hết, ở cả hai bài báo cần xác định số **phase** trước sau đó mới tính bao nhiêu hành động cần có. Tại giao lộ ngã tư chúng tôi quy định có 2 **phase**:

- **Phase I:** Chiều xe Đông - Tây, Đông sang Bắc, Đông sang Tây, Tây sang Bắc, Tây sang Nam.
- **Phase II:** Chiều xe Bắc - Nam, Bắc sang Tây, Bắc sang Đông, Nam sang Tây, Nam sang Đông.

Để có nhiều hướng đánh giá và so sánh, nhóm đề xuất hiện thực cả hai cách hiện thực đã nêu ra ở hai bài báo tham khảo:



Hình 4.1: Chính sách hành động của mô hình theo cách 1

1. Cách 1: Thay đổi thời gian của mỗi phase

Số phase của mô hình là 2 tương ứng 5 hành động có thể thực hiện bởi Agent. Ở trạng thái ban đầu, Agent có thể chọn giữ nguyên thời gian các **phase** hoặc tăng hay giảm thời gian 1 **phase**. Một chính sách được áp dụng với phương án này, khi Agent thực hiện tăng hay giảm thời gian, Agent chỉ còn hai lựa chọn sau đó có thể thực hiện đó là giữ nguyên thời gian (quay lại hành động (t_1, t_2)) hoặc tiếp tục thực hiện lại hành động đó (Hình 4.1)

Ưu điểm: Biết trước thời gian chờ của một **phase**, thời gian không thay đổi nhiều tạo nên tính an toàn cho người đi đường và ổn định của hệ thống.

Nhược điểm: Chính vì không thay đổi nhiều về mặt thời gian dẫn đến sự không linh hoạt tác động của của Agent đối với môi trường mà nó tương tác.

2. Cách 2: Lựa chọn phase được bật đèn xanh

Nếu như ở cách 1 thay đổi thời gian đèn xanh của mỗi phase thì đổi với cách 2 nhóm chọn thời gian cố định 20 giây của đèn xanh cho một lần ra quyết định. Bằng cách này, thời gian đèn xanh của mỗi **phase** được tính là bội số của 20. Tập hợp 2 actions tương ứng với 2 phases mà Agent có thể chọn là:

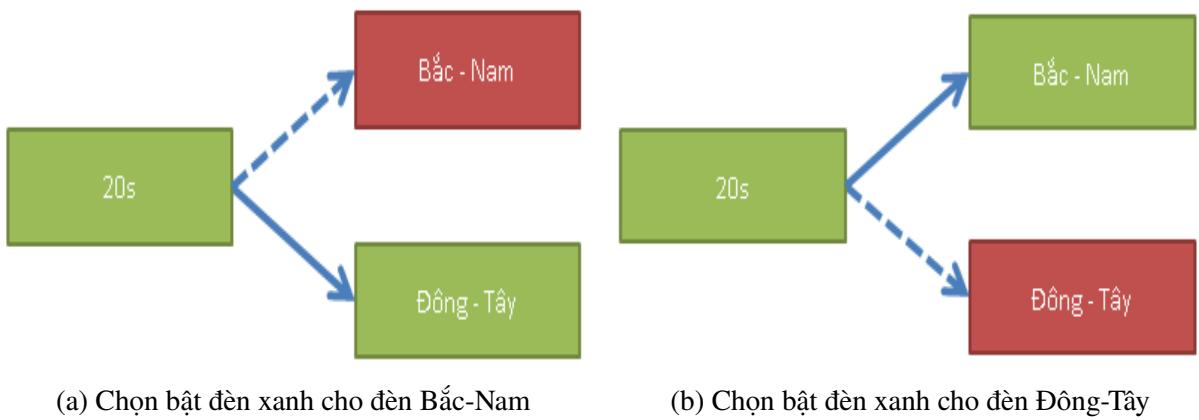
$$A = \{NS, EW\} \quad (4.1)$$

Với NS là hành động bật đèn xanh ở Bắc-Nam, EW là hành động bật đèn xanh ở Đông-Tây. Sau mỗi lần ra quyết định, Agent có thể chọn tiếp hành động cũ hoặc chọn hành động còn lại (Hình 4.2b).

Ưu điểm: Có thay đổi lớn về mặt thời gian giúp Agent thấy rõ hiệu quả của hành động.

Nhược điểm: Không biết trước thời gian của 1 **phase**, người đi đường sẽ không biết phải chờ bao lâu.

Cả hai cách đều dựa vào số **phase** của giao lộ. Mỗi cách đều có ưu và nhược điểm riêng của mình, vì thế nhóm hiện thực cả 2 cách này và sẽ đánh giá để đưa ra mô hình hành động phù hợp nhất.



Hình 4.2: Hành động bật đèn xanh theo cách 2

4.3 Phần thưởng

Phần thưởng hợp lý sẽ giúp Agent học chính xác và ra quyết định ngày càng tốt. Đối với mỗi cách mô hình phần thưởng ở hai bài báo, chúng tôi quyết định hiện thực cả hai và lần lượt đánh giá từng cách rồi dựa trên kết quả đánh giá đó để chọn ra phần thưởng phù hợp.

Mô hình 1 định nghĩa phần thưởng là độ chênh lệch giữa tổng thời gian chờ của xe ở chu kì trước và sau khi thực hiện hành động. Trong đó, tổng thời gian chờ của một chu kì được tính từ thời điểm bắt đầu ($t = 0$) của kịch bản cho đến thời điểm của chu kì t . Để đánh giá được mô hình chạy tốt hay xấu bằng phần thưởng này, nhóm tính tổng phần thưởng khi kết thúc 1 kịch bản mô phỏng. Bằng cách tính này, phần thưởng luôn âm và 0 là phần thưởng lớn nhất có thể đạt được khi không có xe nào phải đợi.

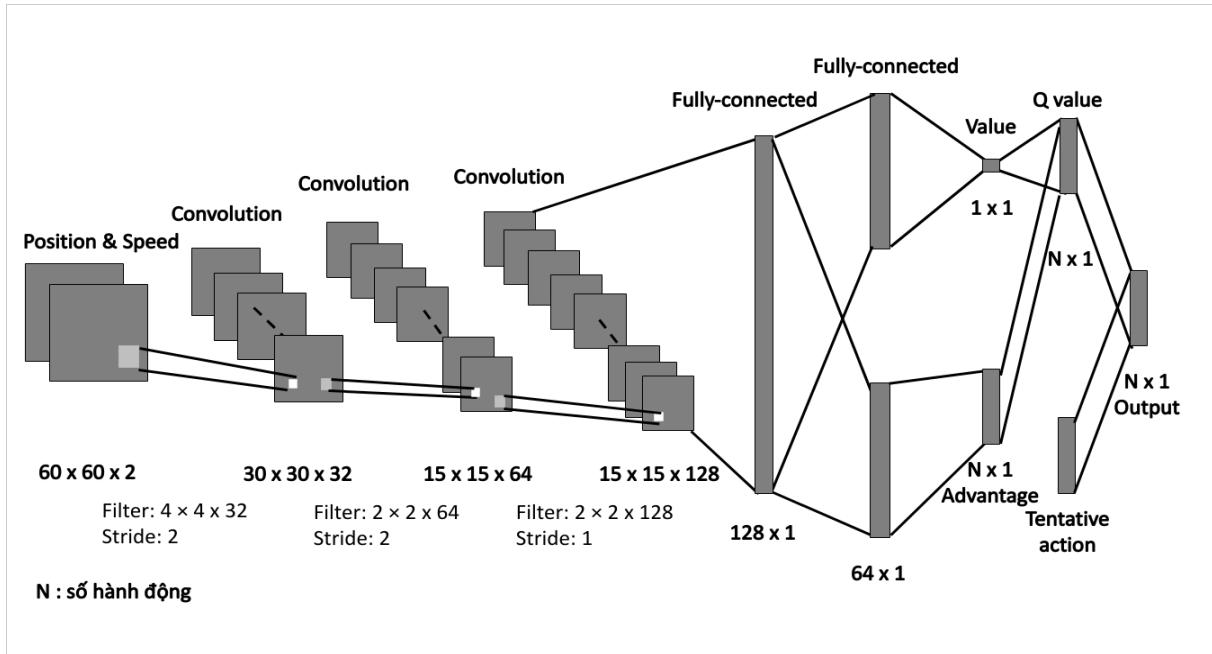
Mô hình 2 có phần thưởng được tính bằng thời gian chờ tức thời của xe ở chu kì trước và sau khi thực hiện hành động. Vì thế, phần thưởng có thể âm, có thể dương hoặc bằng 0. Phần thưởng này thể hiện mức độ tốt của hành động vừa thực hiện. Nhóm đánh giá mô hình khi dùng cách mô hình phần thưởng này bằng cách tính tổng các phần thưởng âm. Như vậy, khi tổng phần thưởng âm càng thấp, nhóm biết được Agent thực hiện hành động sai nhiều hơn hành động đúng.

4.4 Mạng nơ-ron và các kỹ thuật được áp dụng

Vì mô hình trạng thái chúng tôi chọn là **mô hình 1** nên mạng nơ-ron cũng được lấy từ mô hình này để đảm bảo sự phù hợp của dữ liệu đầu vào mà không phải thay đổi nhiều về kiến trúc.

Mạng nơ-ron mà chúng tôi chọn gồm 3 lớp Convolution và 2 lớp Fully-connected (Hình 4.3). Dữ liệu đầu vào là hai ma trận 60×60 đại diện cho hai thông tin vị trí và tốc độ được lấy từ giao lộ mô phỏng. Vì chúng tôi chọn hai cách hiện thực hành động với số hành động khác nhau nên kích thước của ma trận Advantage, Q value sẽ là 2×1 hoặc 5×1 tùy vào cách mô hình hành động. Với cách mô hình hành động 2, đầu ra của mạng nơ-ron là ma trận Q value, Agent sẽ dựa vào tập Q value này để chọn ra vị trí của phần tử có giá trị lớn nhất và đó là cách chọn ra hành động. Nhưng với cách mô hình hành động 1, sau khi tính ra ma trận Q value sẽ được kết hợp với một luật chính sách và tính ra ma trận cuối cùng để Agent lựa chọn.

Ngoài việc xây dựng mạng nơ-ron và cũng là mạng dùng trong hệ thống, nhóm xây dựng một mạng nơ-ron khác và gọi nó là mạng mục tiêu. Mạng mục tiêu có kiến trúc tương tự với mạng chính và cũng tính tập giá trị Q, gọi là Q target. Có thể nói mạng mục tiêu dùng để đánh

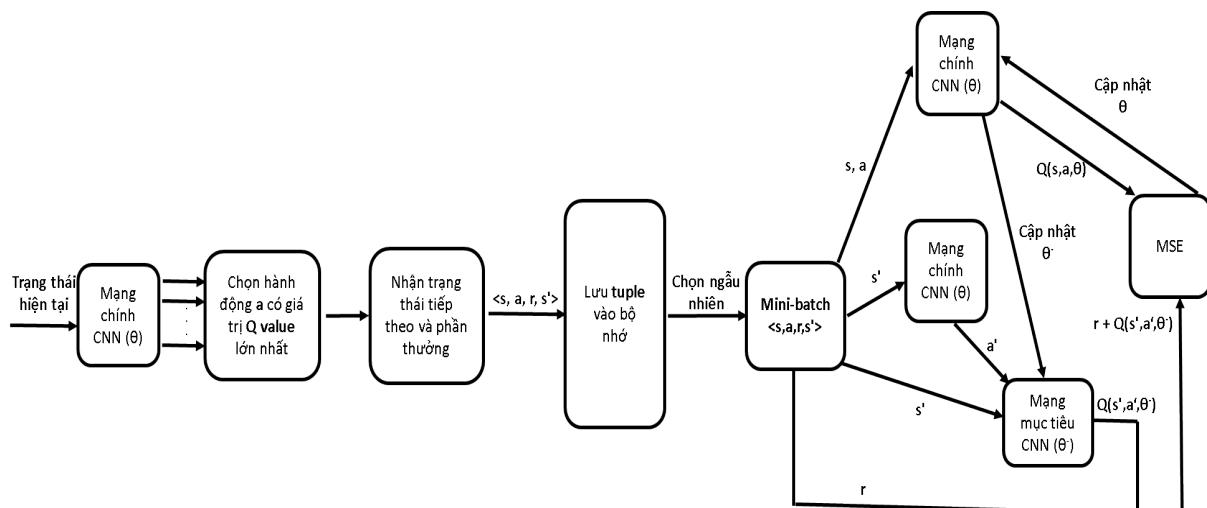


Hình 4.3: Mạng nơ-ron

nhăn như trong các giải thuật huấn luyện đánh nhăn.

Ngoài ra kiến trúc mạng nơ-ron của **mô hình 1** sử dụng rất nhiều kỹ thuật tiên tiến như: Dueling Network, Double-Q-Learning, Priority Experience Replay tạo nên sự ổn định và kết quả huấn luyện cũng sẽ tốt hơn, cụ thể:

- **Dueling Network:** tính giá trị Q value của mỗi hành động bằng cách lấy giá trị tổng thể (Value) cộng với độ thuận lợi (Advantage) của hành động đó so với độ thuận lợi trung bình của các hành động.
- **Double-Q-Learning:** tính giá trị Q target nhưng thay vì tính trực tiếp bằng mạng mục tiêu, trạng thái sẽ được đưa vào mạng chính để tìm hành động có Q value tốt nhất rồi mới đưa trạng thái và hành động vào mạng mục tiêu để tìm ra Q target.
- **Priority Experience Replay:** Sau mỗi lần Agent thực hiện hành động và nhận về cho mình



Hình 4.4: Kiến trúc tổng quan mô hình hiện thực

phần thưởng, kinh nghiệm này sẽ được lưu giữ vào một bộ nhớ. Kỹ thuật Priority Experience Replay sẽ chọn những kinh nghiệm mà tại đó sai số Agent nhận được là lớn nhất từ đó giúp Agent học nhanh hơn. Tuy nhiên sau khi tự hiện thực kỹ thuật này, kết quả mà nhóm nhận được khi áp dụng là không đổi, Agent không học nhanh hơn khi không áp dụng. Vì thế kỹ thuật Priority Experience Replay sẽ chưa được chúng tôi áp dụng cho mô hình này.

Các mô hình đều có kiến trúc tổng quan giống nhau nghĩa là giống về việc huấn luyện, kỹ thuật tiên tiến như nhau (Hình 4.4). Cụ thể:

- Trạng thái (s) sẽ là dữ liệu đầu vào của mạng nơ-ron và tính ra tập Q value, Agent dựa vào tập để đưa ra hành động (a) điều khiển đèn giao thông và nhận về cho mình trạng thái mới (s') và phần thưởng (r). Đây là phần mà mô hình dùng để thử nghiệm.
- Phần còn lại là dùng để huấn luyện, mỗi lần Agent nhận s' và r và lưu thành một tuple $\langle s, a, r, s' \rangle$ vào một bộ nhớ. Sau đó chọn ngẫu nhiên một tập nhỏ để huấn luyện.
- Để huấn luyện, trước hết là tính MSE (hàm loss), cần hai giá trị là Q value và Q target. Q value được tính bằng cách đưa s vào mạng chính và chọn ra Q value của a . Để tính Q target, s' được đưa vào mạng chính và tìm ra a' có Q value lớn nhất, tiếp đó đưa s' vào mạng mục tiêu và tìm ra Q value với a' , kết hợp giá trị này với r ta được Q target.
- Có được Q value và Q target ta tính MSE và cập nhật trọng số cho mạng chính bằng giải thuật Adam và cập nhật lại trọng số mạng mục tiêu bằng một phương trình.

Tuy nhiên, các mô hình mà chúng tôi lựa chọn là mô hình đã được hiện thực và thử nghiệm. Mô hình cho kết quả thử nghiệm tốt nhất sẽ được chọn làm mô hình chính của nhóm và được trình bày cụ thể ở chương 6.

Chương 5

Hiện thực mô hình

Với những mô hình điều khiển giao thông bằng học tăng cường đã tìm hiểu, những kiến thức liên quan cần thiết, và mô hình mà chúng tôi đã đề xuất. Trong phần này, chúng tôi sẽ trình bày công cụ sử dụng, chi tiết cách xây dựng kịch bản giao thông huấn luyện, các thành phần cơ bản của mô hình như trạng thái, hành động, phản thưởng và mạng DQN.

5.1 Đề xuất công cụ sử dụng

Sự phát triển mạnh mẽ của Deep Learning, Python cung cấp rất nhiều thư viện như Tensorflow, Theano, Torch, Caffe... giúp hiện thực các giải thuật. Mặt khác, để có thể hiện thực mô hình mạng nơ-ron dễ dàng hơn, **Keras** đã ra đời và được biết đến với tính dễ sử dụng, thân thiện với người dùng nhưng cũng rất mạnh mẽ.

Keras [7] là một thư viện được phát triển vào năm 2015 bởi François Chollet, là một kỹ sư nghiên cứu Deep Learning tại Google. Nó là một Open Source cho Neural Network được viết bởi ngôn ngữ Python. **Keras** là một API bậc cao có thể sử dụng chung với các thư viện Deep Learning nổi tiếng như Tensorflow (được phát triển bởi Google), CNTK (được phát triển bởi Microsoft), Theano (người phát triển chính Yoshua Bengio).

Keras có một số ưu điểm như :

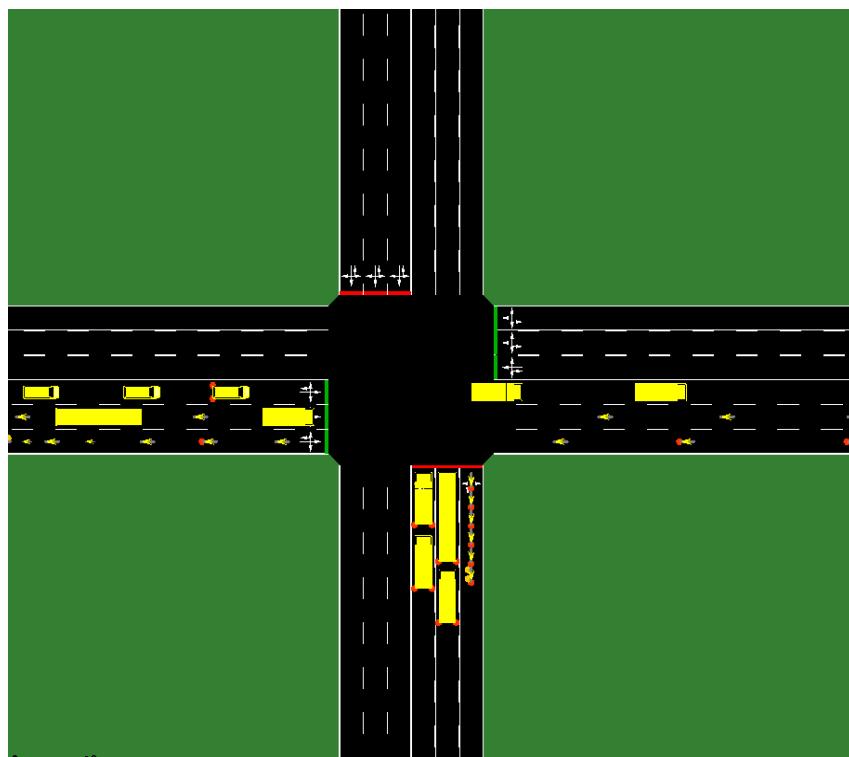
- Dễ sử dụng, xây dựng model nhanh.
- Có thể chạy trên cả CPU và GPU
- Hỗ trợ xây dựng CNN, RNN và có thể kết hợp cả 2.

Chính vì những ưu điểm như vậy, chúng tôi đã chọn **Keras** [10] để hiện thực mô hình Deep Learning Network.

5.2 Xây dựng mô phỏng giao thông bằng SUMO

Bước đầu tiên của nhóm chúng tôi thực hiện trong phần hiện thực là tìm cách sử dụng công cụ mô phỏng SUMO¹, hiểu được cách xây dựng một hệ thống giao thông, định ra kịch bản di chuyển của các phương tiện, lấy thông tin dữ liệu từ SUMO và điều khiển đèn giao thông ở giao lộ thông qua code Python.

¹<http://sumo.dlr.de/userdoc>



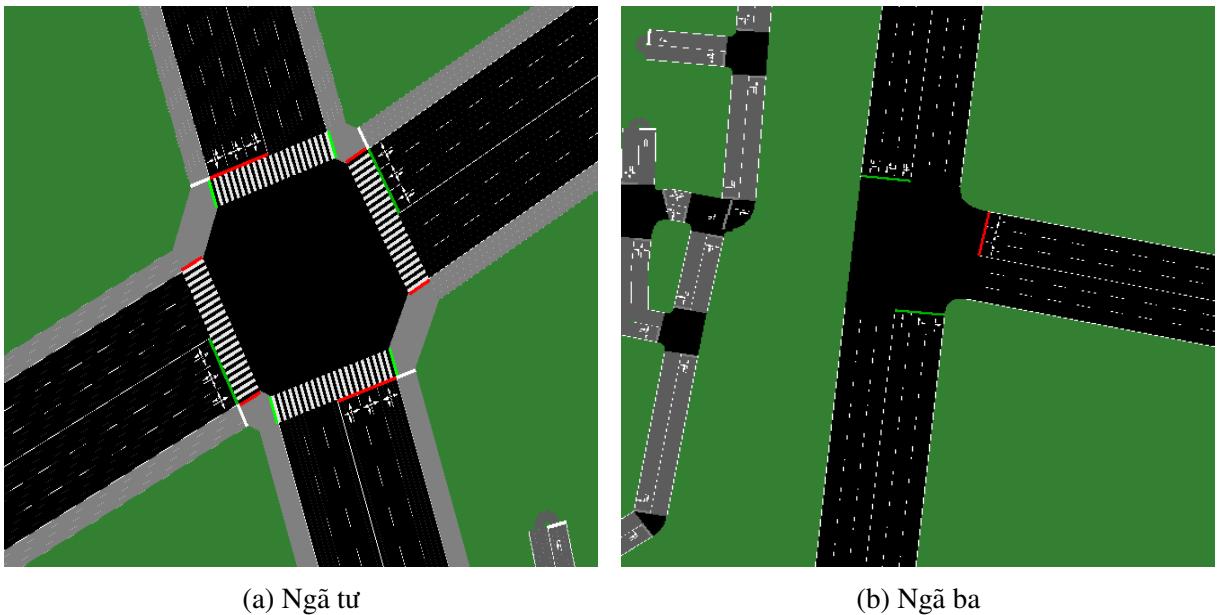
Hình 5.1: Giao lộ mô phỏng ngã tư ban đầu trong SUMO

5.2.1 Xây dựng bản đồ mô phỏng

Để có được một giao lộ chạy mô phỏng, ta có thể tạo trực tiếp một bản đồ mới trên SUMO bằng cách tạo trực quan. Hình 5.1 là kết quả tạo bằng tay của nhóm khi bắt đầu tìm hiểu về



Hình 5.2: Bản đồ mô phỏng sau khi được chuyển đổi



Hình 5.3: Giao lộ mô phỏng trong SUMO

SUMO và một số chức năng cơ bản để tùy chỉnh bản đồ. Ta tiến hành khởi tạo file dữ liệu để thiết lập mô phỏng gồm các file dạng *.xml*, cụ thể:

- **net.net.xml**: định nghĩa các con đường, vị trí, các giao điểm và đèn giao thông. Những chức năng của SUMO được chúng tôi sử dụng để chỉnh sửa bản đồ gồm tạo đường đi, thêm làn đường, thiết lập luật các phương tiện di chuyển trên đường, trên giao lộ, hệ thống đèn giao thông.
- **result.rou.xml**: định nghĩa loại xe, chiều di chuyển và số lượng của các phương tiện. Đây là file được tạo bằng tay và chỉnh sửa trực tiếp.
- **myroutes.rou.xml**: sinh ra từ result.rou.xml sử dụng lệnh **duarouter**. File chứa đầy đủ thông tin chi tiết về chiều di chuyển, loại, kích thước của từng phương tiện.
- **sumoconfig.sumoconfig**: dùng để chạy mô phỏng, chứa các đường dẫn đến file net.net.xml và myroutes.rou.xml. Để chạy mô phỏng có UI ta sử dụng lệnh **sumo-gui** và khi huấn luyện ta không cần đến UI thì sử dụng **sumo**.

Nhưng với mong muốn có một bản đồ phức tạp hệ thống giao thông thì việc xây dựng bằng tay sẽ mất rất nhiều thời gian. Vì thế, **OpenStreetMap** cho phép ta tạo một bản đồ bằng cách xuất một bản đồ thật sang dạng file *.osm*. Sau đó sử dụng **netconvert** chuyển sang dạng *.net* để SUMO có thể đọc được. Gọi lệnh **netedit {tên file .net}.net** để mở giao diện SUMO của bản đồ đó và chỉnh sửa [16]. Ví dụ như hình 5.2 được lấy từ hình ảnh một vùng ở Quận 10.

Để có được kết quả đa dạng, phù hợp với thực tế nhất có thể. Chúng tôi đã tạo ra 2 loại bản đồ, đồng thời chỉnh sửa sao cho mỗi con đường có 3 làn xe như giao lộ trên thực tế:

1. Bản đồ ngã tư tại giao lộ đường Lý Thường Kiệt và đường 3/2 (Hình 5.3a)
2. Bản đồ ngã 3 tại giao lộ đường Sư Vạn Hạnh và đường 3/2 (Hình 5.3b)

Qua đó, mục tiêu của nhóm đặt ra là mô phỏng đúng thực tế hơn, chúng tôi thiết lập luật di chuyển của các phương tiện (Hình 5.4), cụ thể:

- Sử dụng các loại xe thường thấy trên đường gồm xe buýt, xe máy, taxi.
- Làn trong cùng dành cho các loại xe bốn bánh, làn ngoài cùng dành cho xe máy, làn giữa dành cho mọi loại xe.



Hình 5.4: Ngã tư mô phỏng với xe cộ

- Các phương tiện di chuyển tùy ý trên làn xe của mình, nghĩa là chúng không thẳng hàng và có vị trí tùy ý trên làn đường.

Sau khi có được bản đồ, chúng tôi tiến hành quy định loại xe, kích thước của các phương tiện, chiều dài và chiều rộng của làn xe được sử dụng trong mô phỏng.

5.2.2 Các loại phương tiện

Quy ước chiều rộng của một làn xe là 3m, tức chiều rộng của con đường là 9m và kích thước của các loại xe được đặt lại sao cho trùng với kích thước chiếc xe thường thấy trên giao lộ như sau:

1. Xe buýt

Có chiều rộng 2.45m, chiều dài 9.44m và khoảng cách nhỏ nhất với xe phía trước là 2m.

2. Xe taxi

Có chiều rộng và chiều dài mặc định và khoảng cách nhỏ nhất với xe phía trước là 2m.

3. Xe máy

Có chiều rộng 0.74m và chiều dài 2.034m và khoảng cách nhỏ nhất với xe phía trước là 0.5m.

5.2.3 Xây dựng kịch bản giao thông

Dữ liệu huấn luyện là thành phần không thể thiếu trong bất kỳ thuật toán học máy nào. Ở phần này, chúng tôi sẽ trình bày cách tạo ra các kịch bản giao thông, chúng được sử dụng trong quá trình huấn luyện Agent cũng như đánh giá hiệu suất của Agent.

Sau khi tham khảo 2 bài báo, chúng tôi thấy được họ chỉ huấn luyện trên 1 kịch bản dẫn đến mô hình có thể sẽ không thể đạt hiệu suất cao trong các kịch bản ngẫu nhiên. Do đó, để đa dạng các loại kịch bản trong quá trình huấn luyện, nhóm sẽ tạo ra 4 loại kịch bản khác nhau (tham khảo [18]) trên mỗi loại bản đồ đã tạo:

1. **LOW:** mật độ giao thông thấp cả 4 phía, kịch bản này để huấn luyện agent nhận biết được cách ra quyết định trong tình trạng ít phương tiện giao thông.



Hình 5.5: Ngã tư kẹt xe

2. **HIGH:** mật độ giao thông cao từ cả 4 phía, kịch bản này để huấn luyện agent nhận biết được cách ra quyết định trong tình trạng giao thông kẹt xe.
3. **NORTH-SOUTH:** mật độ giao thông cao ở hướng Bắc-Nam và thấp ở hướng Đông-Tây, giúp Agent nhận biết được tình huống chỉ kẹt xe 1 phía.
4. **EAST-WEST:** mật độ giao thông cao ở hướng Đông-Tây và thấp ở hướng Bắc-Nam, cũng giúp Agent nhận biết được tình huống chỉ kẹt xe 1 phía.

Mỗi loại kịch bản được định nghĩa với các thông tin cơ bản sau đó dùng lệnh tạo kịch bản ngẫu nhiên với các thông tin đó với thời gian chạy cho đến khi không còn xe nào là 5000 giây. Cụ thể như loại xe, hướng đi của mỗi làn, thời gian xuất hiện của các phương tiện giao thông.

Tuy nhiên, một số tình huống xảy ra khi thực hiện sử dụng hai **phase** đó là khả năng bị kẹt xe (Deadlock) giữa hai chiều xe từ Nam sang Tây và từ Bắc sang Nam, tương tự với từ Tây sang Bắc và Đông sang Nam. Đây là hạn chế của SUMO khi mà hai xe đi trái chiều nhau trên một đường nó sẽ đứng yên một lúc lâu (Hình 5.5). Khi trường hợp này xảy ra, Agent chọn bất kì hành động nào cũng không cho ra kết quả tốt dẫn đến sai lệch phần thưởng và học sai. Vì thế, nhóm loại bỏ trong kịch bản hai chiều xe này để kết quả huấn luyện được hiệu quả.

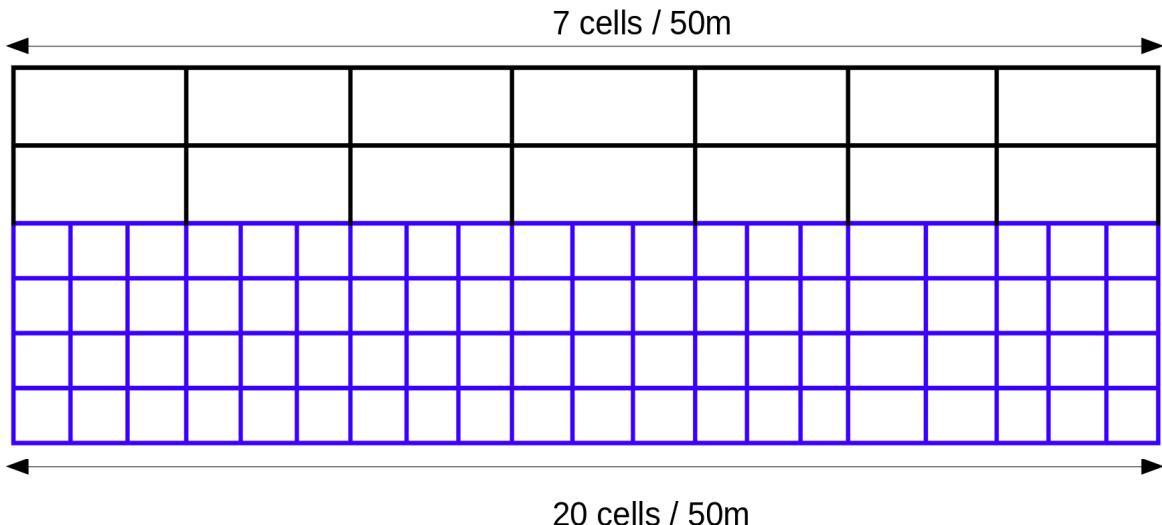
5.3 Mô hình bài toán

Ở mục này, chúng tôi đề xuất mô hình đã hiện thực gồm 4 phần chính cần hiện thực là trạng thái, hành động, phần thưởng và mạng nơ-ron dùng trong mô hình.

5.3.1 Trạng thái

Như đã giới thiệu trong phần 3.2.1, mỗi trạng thái phải được biểu diễn dưới dạng ma trận có kích thước $60 \times 60 \times 2$ thể hiện được thông tin vị trí và tốc độ. Thư viện **traci** cung cấp những Python APIs cho phép lấy hầu hết thông tin của kịch bản giao thông đã xây dựng như: số lượng xe hiện tại, vị trí và tốc độ của 1 xe,...

Ma trận trạng thái được chúng tôi hiện thực từ những thông tin đó, cụ thể là xây dựng mô hình trạng thái. Với mong muốn đánh giá mô hình trạng thái, tìm ra cách mô hình tốt nhất để



Hình 5.6: Chia làn đường thành những ô nhỏ để ánh xạ sang ma trận

phụ vụ trong quá trình huấn luyện, chúng tôi đề xuất hai phương án mô hình trạng thái của bài báo **DeTLC** như đã xác định ở phần 4.1 với ưu và nhược điểm khác nhau, cụ thể:

1. Map 1-1:

Mỗi phương tiện giao thông được ánh xạ sang 1 ô trong ma trận, đồng nghĩa với việc các phương tiện có vai trò như nhau trong cách mô hình này. Đối với bản đồ ngã tư đường Tô Hiến Thành đã tạo, mỗi đường sẽ có 6 làn, 3 làn đi vào và 3 làn đi ra. Theo như cách mô hình của bài báo **DeTLC**, thì họ ánh xạ cả 4 đường đi vào, và 4 đường đi ra. Tuy nhiên, nhóm nhận thấy sự ánh xạ những đường đi ra là không cần thiết trong mô hình này. Do đó, nhóm chỉ tập trung vào việc ánh xạ 4 đường đi vào giao lộ, đây là thông tin chính giúp Agent ra quyết định hành động.

Cụ thể, ma trận trạng thái sẽ được lấy ra từ hàm `getState()` mà nhóm đã hiện thực. Tại mỗi thời điểm t , bằng cách sử dụng các APIs mà Traci cung cấp, chúng tôi sẽ lấy được danh sách `vehicle_ids` của các xe đang xuất hiện trên bản đồ bằng hàm `traci.vehicle.getIDList()`. Mỗi `vehicle_id` tương ứng với 1 xe, do đó có thể dễ dàng lấy được **vị trí** (chính là khoảng cách từ xe đó tới giao lộ) bằng hàm `traci.vehicle.getLanePosition(vehicle_id)` và **tốc độ** bằng hàm `traci.vehicle.getSpeed(vehicle_id)`.

Sau khi xác định được **vị trí** và **tốc độ** của mỗi xe. Thì việc tiếp theo là tìm cách đưa nó vào vị trí tương ứng trong ma trận. Bằng cách chia mỗi đường thành những ô nhỏ như hình 5.6, các ô trên mỗi làn có chiều dài bằng nhau và tương ứng với chiều dài mỗi xe trung bình thường thấy trên làn đường đó. **Ví dụ:** mỗi xe máy có kích thước trung bình 2.3m và khoảng cách giữa các xe thường là 0.2m thì làn đường xe máy sẽ chia thành 20 ô, mỗi ô có chiều dài 2.5m.

Bằng cách sử dụng vị trí của xe so với giao lộ, và `lane_id = traci.vehicle.getLaneID(vehicle_id)` (làn đường xe thuộc về), chúng tôi có thể dễ dàng suy ra được xe đang thuộc ô nào và ánh xạ trực tiếp vào ma trận. Khi đã xác định được vị trí trong ma trận thì sẽ cập nhật giá trị = 1 cho ma trận vị trí và vận tốc tức thời cho ma trận vận tốc.

Sau khi tính toán ma trận vị trí và tốc độ biểu diễn trạng thái phù hợp với bản đồ, ma trận vị trí có dạng như hình 5.7. Trong đó, đường màu cam biểu thị cho làn đường sẽ được ánh xạ, còn lại sẽ có giá trị 0. Hình 5.8 là một ví dụ sự chuyển đổi từ tình trạng giao thông tại giao lộ Tô Hiến Thành sang dạng ma trận vị trí. Một ma trận khác cũng tương tự ma trận

vị trí nhưng thay giá trị 1 thành tốc độ của xe đó.

Ưu điểm: hiện thực đơn giản, tốn ít chi phí. Số xe trên một làn được đưa vào ma trận nhiều hơn cho thông tin nhiều về chiều dài hàng đợi.

Nhược điểm: không quan tâm sự khác biệt giữa các loại xe, kích thước.

2. Map 1-N:

Trong kịch bản có nhiều loại xe khác nhau với kích thước khác nhau trên giao lộ, nhóm chọn xe máy làm kích thước chuẩn của ô. Mỗi một xe máy trên đường tương đương với một ô có giá trị 1, loại xe khác tính theo kích thước của xe máy và có thể tương đương với một hoặc nhiều ô giá trị 1. Không những thế, tất cả các xe đi trên mọi làn xe của giao lộ đều được đưa vào ma trận.

Với cách map này, một xe có thể tương ứng với nhiều giá trị 1 trong ma trận nên ma trận tốc độ các giá trị 1 đó đều được thay bằng tốc độ của xe đó (Hình 5.9).

Ưu điểm: Phân biệt các loại xe, kích thước xe. Thể hiện hầu hết thông tin ở mức chính xác nhất.

Nhược điểm: Hiện thực khó, tốn chi phí tính toán. Số lượng xe đưa vào ma trận ít hơn nhiều so với phương án 1.

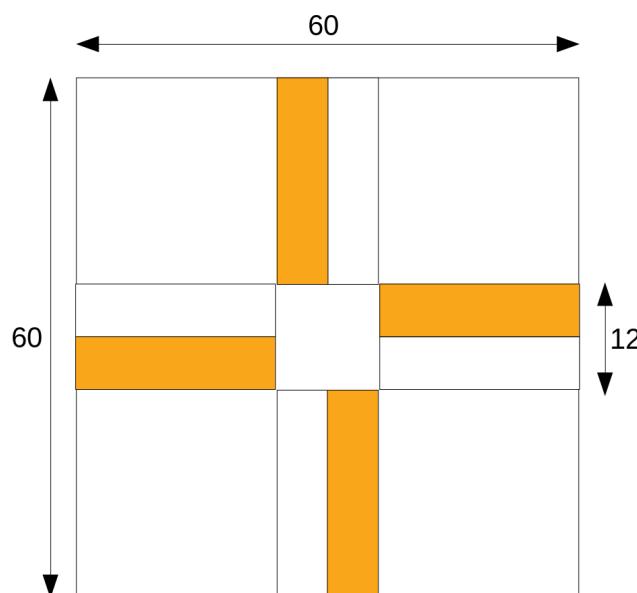
Mỗi cách hiện thực trạng thái đều có ưu điểm và nhược điểm riêng của mình. Trong phần hiện thực, chúng tôi tiến hành thử nghiệm cả hai phương án và đánh giá qua các tiêu chí khác nhau để chọn ra phương án tốt nhất.

5.3.2 Hành động

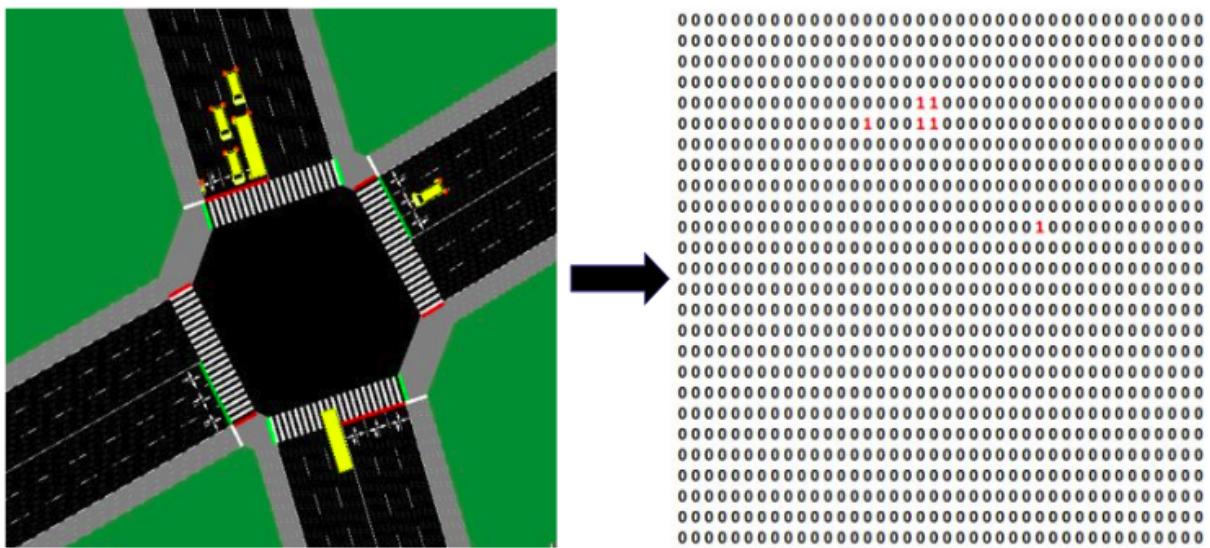
Với mỗi mô hình hành động ở hai bài báo tham khảo, nhóm đưa ra hai cách hiện thực, cụ thể:

1. Cách 1: Thay đổi thời gian của mỗi phase

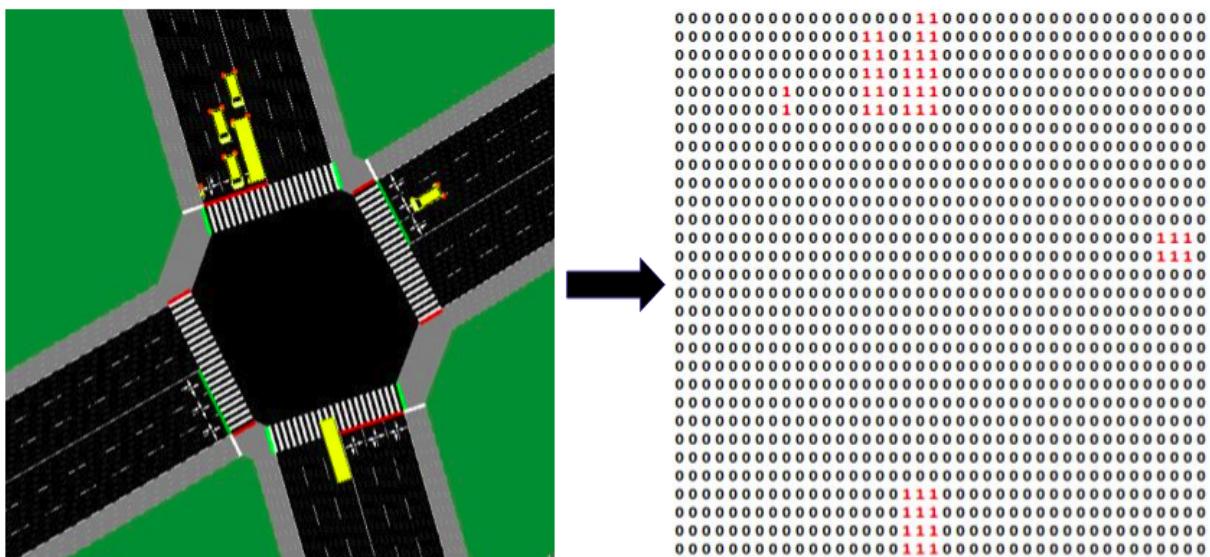
Ta định nghĩa một mảng 5 phần tử tương ứng 5 hành động có thể thực hiện bởi Agent. Vì chỉ có 2 **phase**, mỗi hành động là một mảng 2 phần tử có giá trị 0 (giữ nguyên), 5 (tăng 5 giây), -5 (giảm 5 giây) cho một phase.



Hình 5.7: Biểu diễn state dưới dạng ma trận



Hình 5.8: Chuyển đổi sang ma trận trạng thái map 1-1 bằng thư viện traci



Hình 5.9: Chuyển đổi sang ma trận trạng thái map 1-N bằng thư viện traci

Để hiện thực chính sách buộc Agent đưa ra hành động hợp pháp, ta cần định nghĩa **tentative action** tương ứng với mỗi hành động. **Tentative action** là một mảng có 5 phần tử, mỗi phần tử thể hiện sự hợp pháp của một hành động. Giá trị $-\inf$ là hành động đó không được phép, 1 là hành động đó được phép. Chúng tôi lưu lại hành động trước đó của Agent để có thể đưa ra mảng **tentative action** hợp lý đối với hành động tiếp theo. Cụ thể:

- **Action:** $[[0, 0], [5, 0], [-5, 0], [0, 5], [0, -5]]$.
- **Tentative action:** $[[1, 1, 1, 1, 1], [1, 1, -\inf, -\inf, -\inf], [1, -\inf, 1, -\inf, -\inf], [1, -\inf, -\inf, 1, -\inf], [1, -\inf, -\inf, -\inf, 1]]$.

Khi hành động $[0, 0]$ được thực hiện, tức là giữ nguyên thời gian 2 phase (t_1, t_2) đồng thời **tentative action** cho hành động sau là $[1, 1, 1, 1, 1]$ nghĩa là cả 5 hành động đều được phép chọn để thực hiện. Khi hành động $[5, 0]$ được thực hiện ($t_1 + 5, t_2$), **phase I** tăng 5 giây, **phase II** giữ nguyên đồng thời **tentative action** cho hành động sau là $[1, 1, -\inf, -\inf, -\inf]$ nghĩa là Agent chỉ được thực hiện **action** $[0, 0]$ hoặc $[5, 0]$. Tương tự với các hành động

khác. Sau mỗi **phase** luôn có một **phase** đèn vàng để bảo đảm tính đúng đắn của hệ thống đèn giao thông.

Giải thuật tham lam ϵ được dùng để ra quyết định của Agent. Trong trường hợp chọn hành động ngẫu nhiên, Agent chọn ngẫu nhiên các hành động có giá trị 1 trong tentative action để đảm bảo tính hợp pháp. Trường hợp Agent tự ra quyết định, ta tìm hành động hợp lệ và có giá trị **Q value** là lớn nhất. Thời gian ban đầu của mỗi phase là 33 giây được biểu diễn bằng mảng hành động [33,33], sau mỗi lần Agent ra quyết định, ta cập nhật mảng đó theo hành động tương ứng.

Với 2000 bước đầu tiên, Agent thực hiện hành động ngẫu nhiên và lưu kết quả vào bộ nhớ. 10000 bước tiếp theo, ϵ giảm từ 1 về 0 sau mỗi hành động và cũng lưu kết quả vào bộ nhớ, trong giai đoạn này Agent bắt đầu được huấn luyện sau mỗi hành động. Từ bước 12000 trở đi, $\epsilon = 0$ và cũng là lúc Agent thực hiện hành động của riêng mình và cũng được huấn luyện sau mỗi hành động. Lúc này dựa vào kết quả của tổng phần thưởng nhận được, thể hiện rõ ràng sự hiệu quả của mô hình.

2. Cách 2: Lựa chọn phase được bật đèn xanh

Đối với cách hiện thực này, nhóm chọn thời gian đèn xanh cố định để cấp cho 1 **phase** là 20 giây. Ngoài ra, để đảm bảo tính thực tế của cách hiện thực này, khi agent lựa chọn 1 action (tạm gọi là **new_action**), hệ thống sẽ so sánh **new_action** với action trước đó (tạm gọi là **old_action**). Nếu như **new_action** khác **old_action** thì hệ thống sẽ tự động bật đèn vàng cho phase cũ trong vòng **yellow_duration** giây. Ngược lại nếu như **new_action** giống như **old_action**, tức là vẫn giữ trạng thái cũ thì sẽ không cần bật đèn vàng.

Mỗi quyết định của Agent chỉ qua 20 giây, vì thế nếu thực hiện giảm ϵ sau mỗi hành động là rất nhanh chóng, Agent sẽ không kịp thực hiện hết các trường hợp ngẫu nhiên. Nhóm quyết định ϵ giảm theo số bước mô phỏng **eposide** và giảm từ **eposide** 1 đến 60. Từ **eposide** thứ 61 trở đi nghĩa là $\epsilon = 0$, Agent tự ra quyết định.

5.3.3 Phần thưởng

Ý tưởng của hiện thực phần thưởng đều dựa trên hai cách mô hình phần thưởng đã chọn được nêu ra ở mục 4.3. Với mỗi ý tưởng, chúng tôi đề xuất ra cách hiện thực một cách chi tiết sử dụng thư viện **traci** và trích xuất thông tin cần dùng.

1. Mô hình DeTLC:

Để định nghĩa phần thưởng mô hình này, trước hết cần phải tính được tổng thời gian chờ của các xe từ thời điểm $t = 0$ cho đến tại thời điểm t đang xét. Ở đây, thư viện **traci** cung cấp một thông tin là số lượng xe đang chờ tại một thời điểm. Bằng cách cộng dồn số lượng xe này mỗi giây trôi qua trong mỗi **eposide**, nhóm nhận được tổng thời gian chờ của các xe tại thời điểm t tính từ $t = 0$.

Lúc này, ta chỉ cần giữ lại tổng thời gian chờ của xe tại chu kỳ trước khi thực hiện hành động và tính tổng thời gian chờ của xe tại chu kỳ sau khi thực hiện hành động. Lấy hiệu 2 giá trị này ta được phần thưởng luôn âm như bài báo tham khảo.

2. Mô hình DePGVF:

Ở mỗi phương tiện di chuyển trong mô phỏng SUMO đều có một biến đếm thời gian chờ của chính phương tiện đó từ lúc bắt đầu tiến vào một con đường và đến khi ra khỏi con đường đó. Nhóm sử dụng hàm có sẵn trong thư viện **traci** để lấy được tổng thời gian chờ tức thời (*waiting_time_t*) của xe đang chờ tại giao lộ. Sau khi Agent thực hiện một hành động, tổng thời gian chờ tức thời được tính lại (*waiting_time_t1*). Hiệu của *waiting_time_t* – *waiting_time_t1* cũng chính là phần thưởng mà Agent sẽ nhận được. Phần thưởng này có thể âm hoặc dương hoặc bằng 0.

5.4 Mạng chính Deep-Q-Learning

Mạng nơ-ron chúng tôi xây dựng dựa trên bài báo **DeTLC** vừa phù hợp với mô hình trạng thái đã chọn vừa có cấu trúc mạng phức tạp hơn so với mô hình mạng của bài báo **DePGVF**. Mạng này được mô tả với 3 lớp Convolution, và một số lớp Fully-connected. Mỗi lớp Convolution bao gồm ba phần: convolution, pooling, activation. Sự phức tạp của mô hình này được đơn giản hóa với các hàm có sẵn của thư viện **Keras**. Vì mạng Q-Learning này cần biết trước số hành động và hành động mà chúng tôi chọn hiện thực có số hành động khác nhau nên tạm gọi $number_{act}$ là số hành động được mô phỏng ($number_{act}$ có thể là 2 hoặc 5), các lớp này được hiện thực cụ thể như sau:

- **Lớp Convolution:**

$Conv2D(filters, kernelSize, strides, padding = 'Same', activation = LeakyReLU(alpha))$

Mỗi lớp Convolution có giá trị filters, kernel, strides khác nhau. Giá trị 'Same' của padding cho **Keras** biết phải chọn giá trị padding phù hợp để giữ nguyên kích thước ma trận đầu vào. Hàm activation sử dụng LeakyReLU giống như bài báo tham khảo. Ma trận đầu vào có kích thước 60x60x2, ba lớp Convolution được định nghĩa như sau:

1. **Convolution I:** filters = 32, kernel size = (4,4), strides = (2,2).
Input: Ma trận 60x60x2.
Output: Ma trận 30x30x32.
2. **Convolution II:** filters = 64, kernel size = (2,2), strides = (2,2).
Input: Ma trận 30x30x32.
Output: Ma trận 15x15x64.
3. **Convolution III:** filters = 128, kernel size = (2,2), strides = (1,1).
Input: Ma trận 15x15x64.
Output: Ma trận 15x15x128.

- **Lớp Fully-connected:**

$Dense(units, activation = LeakyReLU(alpha))$

Trước khi thực hiện lớp Fully-connected, cần thực hiện flatten ma trận đầu ra của lớp Convolution. Hàm *Flatten()* của **Keras** hỗ trợ ta làm được điều này. Ma trận nhận được sau đó có kích thước 128x1 sẽ được dùng làm ma trận đầu vào lớp Fully-connected. Việc xây dựng hai lớp Fully-connected chỉ là thay thế units tương ứng với số chiều mong muốn của ma trận, cụ thể:

1. **Fully-connected I:** units = 64.
Input: Ma trận 128x1.
Output: Ma trận 64x1.
Ma trận 64x1 được dùng làm dữ liệu đầu vào cho cả hai lớp Fully-connected II và III.
2. **Fully-connected II:** units = 1.
Input: Ma trận 64x1.
Output: Ma trận 1x1 (Value).
3. **Fully-connected III:** units = $number_{act}$ (số chiều của lớp này cũng là số hành động mà chúng tôi mô phỏng).
Input: Ma trận 64x1.
Output: Ma trận $number_{act} \times 1$ (Advantage).

Công việc tiếp theo là tính trung bình của Advantage, nhóm thêm một dữ liệu đầu vào là một ma trận $number_{act} \times number_{act}$, mỗi phần tử có giá trị 0.5 rồi nhân ma trận với Advantage để được một ma trận 1x1 cũng chính là Advantage trung bình ($AvgAdvantage$).

Lúc này, nhóm áp dụng công thức tính **Q value** trong dueling DQN với các ma trận đã có gồm Value, Avantage và $AvgAdvantage$. **Q value** này là một ma trận có kích thước $number_{act} \times 1$. Mỗi phần tử trong **Q value** tương ứng với giá trị Q value của một hành động. Như vậy, khi có dữ liệu đầu vào là trạng thái của mô hình (ma trận 60x60x2), Agent tính được **Q value** và dựa vào nó để chọn ra hành động tác động lên đèn giao thông. Ngoài ra, mạng chính Q-Learning sử dụng giải thuật Adam làm hàm optimizer của mình và đổi với **Keras**, chúng tôi chỉ cần khai báo optimizer trong hàm compile.

5.5 Mạng mục tiêu Deep-Q-Learning

Kĩ thuật Double Q-Learning được áp dụng cho mô hình, đó là lý do chúng tôi cần một mạng mục tiêu. Mạng này có kiến trúc giống với mạng chính. Nhiệm vụ của mạng mục tiêu là để tính **Q target**.

Tuy nhiên, trọng số của mạng mục tiêu không phải cập nhật bằng giải thuật Adam mà thông qua trọng số của mạng chính. Sau khi trọng số của mạng chính được cập nhật, nhóm lấy được trọng số đó và thực hiện cập nhật lại trọng số của mạng mục tiêu theo công thức đã đề ra. Cụ thể, hàm `get_weights()` của model tạo ra bằng Keras cho phép lấy ra tập trọng số của model. Nhóm lấy được tập trọng số của cả hai model mạng chính và mạng mục tiêu sau đó tính ra trọng số mạng mục tiêu mới và cập nhật lại bằng hàm `set_weight()`.

5.6 Thông số liên quan

Hầu hết các thông số mà chúng tôi trình bày trong phần này đều giống trong bài báo mà chúng tôi tham khảo, cụ thể là bài **DeTLC**. Mặt khác, vì phần hiện thực của nhóm có thay đổi nhằm mục tiêu cải thiện tình hình của giao lộ mà gần đúng với thực tế (khác với giao lộ trong bài báo), có một số giá trị được chúng tôi tùy chỉnh nhằm đạt được kết quả tốt hơn. Trong đó, với mô hình huấn luyện có hành động là cách 1 ta có các thông số được biểu diễn ở bảng 5.1. Còn mô hình huấn luyện có hành động là cách 2 cũng tương tự nhưng có một số thông số được thay đổi như kích thước minibatch, số bước giảm của ϵ , số bước pre-training t_p và γ . Cụ thể giá trị được biểu diễn tại bảng 5.2.

Các thông số đều được tham khảo ở bài báo **DeTLC**. Tuy nhiên, khi xây dựng mô hình hành động theo cách 2, nhóm quyết định thay đổi số bước giảm ϵ nhằm tăng thêm nhiều hành động ngẫu nhiên. Với cả hai bảng thông số trên, nhóm tiến tới bước tiếp theo đó là lựa chọn cách huấn luyện.

5.7 Chiến lược huấn luyện

Chiến lược huấn luyện cũng là yếu tố ảnh hưởng đến kết quả của mô hình. Để kiểm chứng điều này, chúng tôi đề xuất hai chiến lược huấn luyện khác nhau nhằm có nhiều lựa chọn mô hình có kết quả tốt nhất. Sự giống nhau của hai chiến lược này là đều có đủ 4 loại kịch bản giao thông như đã trình bày ở mục 5.2.3 nhưng khác nhau ở lưu lượng xe, cụ thể:

- Chiến lược huấn luyện kịch bản cố định:** Ở chiến lược này, 4 loại kịch bản sẽ được cố định lưu lượng xe và luân phiên nhau nạp vào SUMO để tiến hành huấn luyện.

Thông số	Giá trị
Kích thước Memory	20000
Kích thước minibatch	64
Starting ϵ	1
Ending ϵ	0
Số bước giảm từ Starting ϵ đến Ending ϵ	10000 (bước huấn luyện)
Số bước pre-training t_p	2000
Update rate α	0.001
Discount factor γ	0.99
Learning rate ϵ_r	0.0001
Leaky ReLU β	0.01

Bảng 5.1: Thông số mô hình hành động cách 1

Thông số	Giá trị
Kích thước Memory	20000
Kích thước minibatch	100
Starting ϵ	1
Ending ϵ	0
Số bước giảm từ Starting ϵ đến Ending ϵ	100 (eposide)
Số bước pre-training t_p	0
Update rate α	0.001
Discount factor γ	0.75
Learning rate ϵ_r	0.0001
Leaky ReLU β	0.01

Bảng 5.2: Thông số mô hình hành động cách 2

2. **Chiến lược huấn luyện kịch bản ngẫu nhiên:** Chiến lược này được hiện thực nhằm với kì vọng sẽ làm tăng tính tổng quát cho mô hình. Trong đó, chúng tôi quyết định thực hiện tương tự là luân phiên 4 loại kịch bản nhưng lưu lượng xe của kịch bản được nạp vào sẽ được thay đổi ngẫu nhiên.

Đó là toàn bộ phần hiện thực mà nhóm đã lựa chọn và xây dựng. Nhằm tìm ra được mô hình nào là tốt nhất, chúng tôi tiến hành thử nghiệm, đo đạc kết quả, so sánh với hệ thống giao thông cố định.

Chương 6

Đánh giá và kết quả đạt được

Trong chương này, chúng tôi sẽ trình bày về những thử nghiệm và kết quả. Trong đó, để đánh giá được hiệu suất của Agent, chúng tôi sẽ giới thiệu các phương pháp đánh giá khác nhau, sau đó so sánh kết quả giữa Agent đã được huấn luyện với hệ thống giao thông cố định.

Bảng 6.1 là những ký hiệu và ý nghĩa của nó mà chúng tôi sẽ sử dụng trong chương này.

Kí hiệu	Ý nghĩa
STL	Hệ thống giao thông tĩnh (static traffic light).
T	Tổng số timesteps của mỗi kịch bản (episode).
NV_t	Tổng số lượng phương tiện (numb of vehicles) đang xuất hiện trên bản đồ (nằm trong vùng quan sát của Agent) tại thời điểm t và đang có vận tốc nhỏ hơn 0.5m/s.
TV	Tổng số lượng phương tiện (total vehicles) đã xuất hiện trong một kịch bản.
MS	Thời gian tối đa (max steps) cho một kịch bản.
TWT	Tổng thời gian chờ (total waiting times) của tất cả các phương tiện đã đi qua giao lộ trong mỗi kịch bản.
wt_{vt}	Thời gian chờ tức thời của phương tiện v tại thời điểm t
AWT	Thời gian chờ trung bình của mỗi phương tiện trong một kịch bản (s).

Bảng 6.1: Kí hiệu cho chương 6

6.1 Phương pháp đánh giá

Để đánh giá hiệu suất của Agent, chúng tôi sẽ sử dụng bản đồ mà chúng tôi đã huấn luyện.

Mỗi loại bản đồ sẽ được đánh giá trên **9 kịch bản** bao gồm: **4 kịch bản cố định** (HIGH, LOW, NS, EW) đã được dùng để huấn luyện và **5 kịch bản** được tạo ngẫu nhiên để đánh giá được kết quả chính xác nhất.

Các độ đo được sử dụng để đánh giá là:

1. Thời gian chờ trung bình của mỗi phương tiện trong toàn bộ kịch bản.

$$AWT = \frac{\sum_{t=0}^{MS} NV_t}{TV} = \frac{TWT}{TV} \quad (6.1)$$

Độ đo này dùng để xác định trong mỗi kịch bản thời gian chờ của mỗi phương tiện trung bình là bao nhiêu, thời gian chờ càng lớn thì hiệu suất hệ thống càng thấp và ngược lại thời gian chờ càng nhỏ thì hiệu suất càng cao.

2. Thời gian chờ trung bình của mỗi phương tiện tại 1 thời điểm

$$AWT_t = \frac{\sum_{v=0}^{NV_t} wt_{vt}}{NV_t} \quad (6.2)$$

Độ đo AWT cho biết thời gian chờ trung bình tổng thể. Ngược lại, độ đo AWT_t cho thấy được tại mỗi thời điểm, trung bình mỗi xe đã phải đợi bao lâu.

6.2 Mô phỏng hệ thống giao thông tĩnh

Để dễ dàng so sánh, đánh giá được hiệu suất của Agent, chúng tôi sẽ mô phỏng 2 hệ thống giao thông tĩnh (SLT) để áp dụng lên **9 kịch bản** nêu trên với bản đồ ngã tư (Lý Thường Kiệt - 3/2). Cả 2 hệ thống SLT có các thông số về mô phỏng giống như phần huấn luyện, tuy nhiên tập hành động của hệ thống sẽ là 1 chu kỳ của 2 phase (NS và EW) và giữa 2 phase sẽ luôn luôn có 1 phase đèn vàng. Thời gian đèn xanh, đèn vàng của mỗi phase của 2 hệ thống này sẽ cố định như bảng 6.2 và 6.3

Phase	Thời gian (s)
EW	33
NS	33
Yellow	4

Bảng 6.2: Thời gian cố định cho hệ thống STL (33,33)

Phase	Thời gian (s)
EW	40
NS	40
Yellow	4

Bảng 6.3: Thời gian cố định cho hệ thống STL (40,40)

Khi đã cài đặt xong 2 hệ thống STL, chúng tôi tiến hành đánh giá chúng bằng các độ đo trên phần 6.1. Kết quả đánh giá **4 kịch bản cố định** trên ngã tư (Lý Thường Kiệt và 3/2) được thể hiện trong bảng 6.4 và 6.5. Đây là kết quả cơ sở để chúng tôi so sánh với Agent sau khi được huấn luyện.

Dĩ nhiên, không phải lúc nào hệ thống STL cũng có hiệu suất thấp. Tùy thuộc vào loại kịch bản (hay tình trạng giao thông) mà kết quả đánh giá cũng sẽ khác nhau.

Trong kịch bản giao thông LOW, sử dụng hệ thống STL không phải là chiến lược tốt bởi vì sẽ có rất nhiều xe phải đợi 1 lượng thời gian, bởi vì đèn xanh được bật cho đường không có xe nào. Trong kịch bản giao thông HIGH, có rất nhiều xe đến từ 4 hướng. Do đó, hệ thống STL có vẻ như là chiến lược hiệu quả trong tình huống này (vì nó chia đều thời gian để giải phóng xe đều cho các phase). Còn trong kịch bản NS (hoặc EW), hệ thống STL hoạt động không hiệu

	Low	High	NS	EW
TWT	19400	85995	49142	50508
TV	1701	4221	2904	3009
AWT	11.41	20.37	16.92	16.79

Bảng 6.4: Kết quả đánh giá của hệ thống STL (33,33) trên ngã tư Lý Thường Kiệt - 3/2

	Low	High	NS	EW
TWT	22815	90648	52970	56905
TV	1701	4221	2904	3009
AWT	13.41	21.48	18.24	18.91

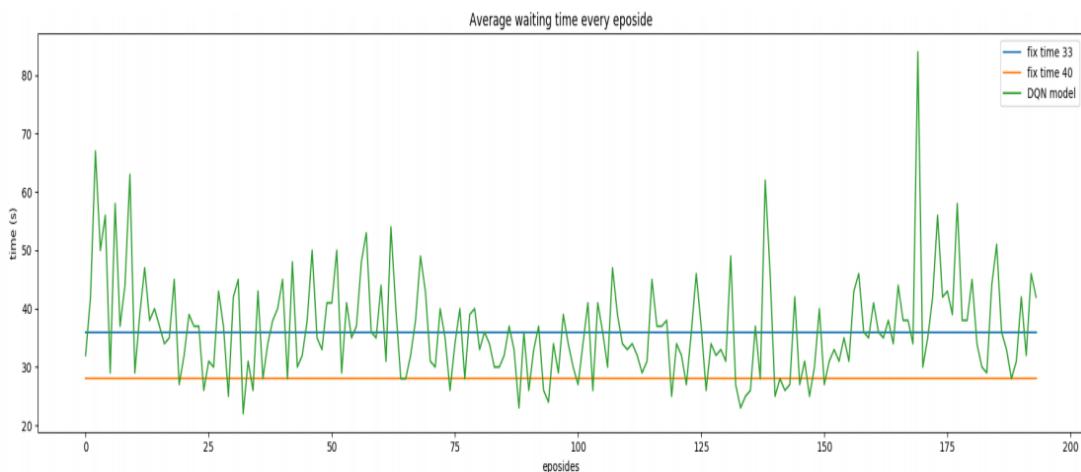
Bảng 6.5: Kết quả đánh giá của hệ thống STL (40,40) trên ngã tư Lý Thường Kiệt - 3/2

quả vì độ ưu tiên của 2 phases là như nhau gây ra thời gian chờ của mỗi xe trên đường NS (hoặc EW) sẽ rất lớn.

6.3 Quá trình thử nghiệm và mô hình đề xuất

Sau khi hiện thực các mô hình đã chọn, chúng tôi lần lượt quan sát thời gian chờ trung bình của xe trong quá trình huấn luyện. Theo quan sát nhóm nhận được nhiều kết quả như thời gian chờ trung bình ngày càng tăng, thời gian chờ trung bình giảm nhưng vẫn không thấp hơn so với hệ thống giao thông tĩnh,... Hình 6.1 biểu diễn kết quả huấn luyện đối với mô hình được hiện thực theo trạng thái ánh xạ 1-N, hành động cách 1 và phần thưởng là hiệu tổng thời gian chờ trung bình. Có thể thấy mô hình này không ổn định trong quá trình huấn luyện khi mà thời gian chờ trung bình liên tục thay đổi theo chiều hướng thấp và cao hơn hệ thống giao thông tĩnh.

Mặc dù vậy, chúng tôi cũng nhận được một kết quả khả quan khi cải thiện được thời gian chờ của phương tiện so với mô hình hệ thống giao thông tĩnh trong quá trình huấn luyện. Mô hình này gồm trạng thái được ánh xạ 1-1, hành động theo cách 2, phần thưởng là tổng thời gian chờ trung bình tức thời. Đây chính là mô hình mà nhóm đề xuất của chúng tôi. Ở phần tiếp theo,



Hình 6.1: Thời gian chờ trung bình khi thử nghiệm một mô hình đã hiện thực

nhóm sẽ trình bày kết quả cụ thể đã thử nghiệm được.

6.4 Kết quả

Sau khi huấn luyện Agent với hai chiến lược trên 2 loại bản đồ (ngã ba, ngã tư), chúng tôi dựa vào kết quả để đưa ra một mô hình có kết quả tốt nhất.

Vì kết quả nhận được và sự so sánh là tương đương nhau nên trong phần này, chúng tôi sẽ đưa ra kết quả đạt được để đánh giá và so sánh độ hiệu quả của hai chiến lược chi tiết trên mô phỏng ngã tư. Đồng thời, chúng tôi cũng đưa ra kết quả huấn luyện và đánh giá ngã 3 ở phần cuối.

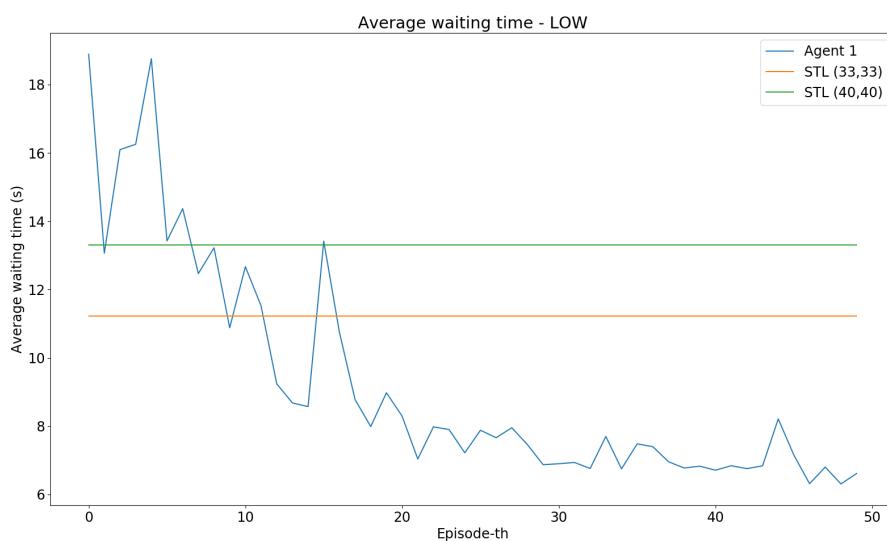
6.4.1 Chiến lược 1

Thời gian chờ trung bình trong toàn bộ kịch bản (AWT)

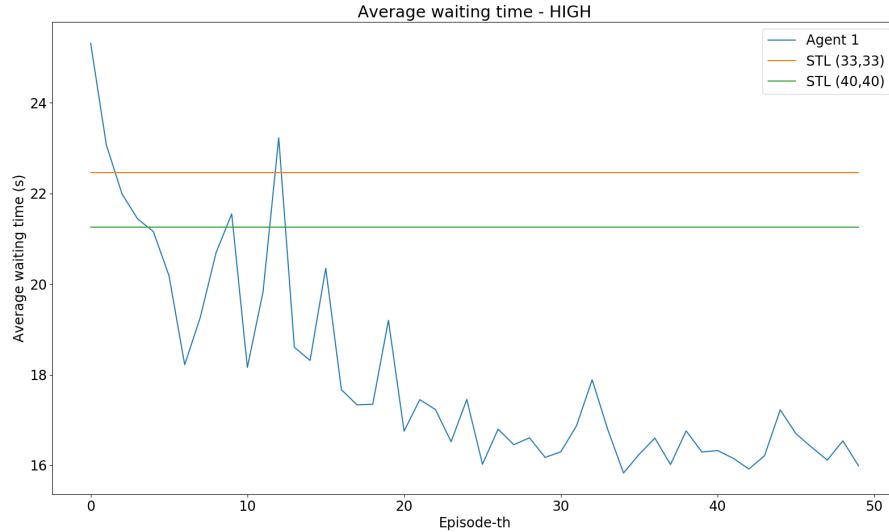
Với chiến lược thứ 1, chúng tôi đã ghi lại sự so sánh thời gian chờ trung bình của mỗi phương tiện (AWT) trong quá trình huấn luyện với thời gian chờ trung bình của hệ thống STL để thấy được quá trình Agent học đã giảm được thời gian chờ như thế nào. Hình 6.2, 6.3, 6.4, 6.5 là kết quả ghi lại quá trình huấn luyện Agent so với AWT của 2 hệ thống STL ứng với 4 loại kịch bản. Trong đó, đường màu xanh là AWT của Agent, đường màu cam là AWT của hệ thống STL (33,33), đường màu xanh lá là AWT của hệ thống STL (40,40).

Ta có thể thấy trong những episode đầu quá trình huấn luyện, Agent có thời gian chờ trung bình khá cao so với 2 hệ thống STL do chúng chỉ bắt đầu đưa ra những hành động ngẫu nhiên (theo giải thuật Greedy- ϵ).

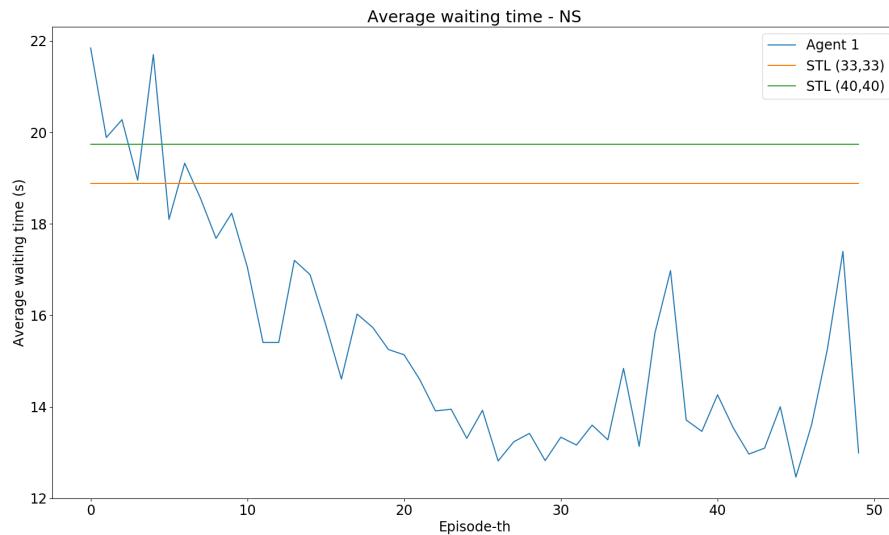
Nhưng sau đó, Agent dần dần học được cách đưa ra hành động để mang lại phần thưởng tích lũy lớn nhất và càng được huấn luyện Agent càng giảm được thời gian chờ trung bình. Khi huấn luyện Agent lên đến khoảng 30 episodes, thời gian chờ trung bình bắt đầu ổn định. Ngược lại, hệ thống STL được đánh giá trên cùng 1 kịch bản với hành động cố định theo chu kỳ nên



Hình 6.2: AWT trong quá trình huấn luyện theo chiến lược 1 - LOW



Hình 6.3: AWT trong quá trình huấn luyện theo chiến lược 1 - HIGH



Hình 6.4: AWT trong quá trình huấn luyện theo chiến lược 1 - NS

dĩ nhiên tổng thời gian chờ sẽ không thay đổi theo thời gian nên sẽ có dạng đường thẳng nằm ngang.

Ví dụ trong kịch bản LOW hình 6.2, thời gian chờ trung bình ban đầu lên đến 18 giây. Trong khi đó AWT của hệ thống STL chỉ khoảng 11 đến 13 giây. Nhưng sau khoảng 30 episode, thời gian trung bình giảm chỉ còn khoảng 7 giây.

Agent sau khi huấn luyện 200 simulations theo chiến lược 1 (tạm gọi là **Agent 1**), sẽ được đánh giá lại độ đo AWT lên 4 kịch bản cố định mà nó đã được huấn luyện. Bảng 6.6 là kết quả so sánh AWT giữa 2 hệ thống STL và Agent 1 trên 4 kịch bản cố định đã được dùng để huấn luyện và độ cải thiện của Agent 1.

Nhận xét: dựa vào bảng 6.6, ta thấy được hệ thống STL (40,40) có hiệu suất thấp hơn hệ thống STL (33,33) vì thời gian chờ trung bình AWT lớn hơn. Đối với Agent 1, nó đã điều khiển



Hình 6.5: AWT trong quá trình huấn luyện theo chiến lược 1 - EW

	STL (33,33)	STL (40,40)	Agent 1	Độ cải thiện
LOW	11.41	13.41	8.81	23% - 34%
HIGH	20.37	21.48	18.54	10% - 14%
NS	16.92	18.24	14.84	12% - 19%
EW	16.79	18.91	13.14	22% - 31%

Bảng 6.6: So sánh AWT của 2 hệ thống STL với Agent 1

giao thông tốt hơn cả 2 hệ thống STL. Trong đó, Agent 1 cải thiện được thời gian chờ trung bình của mỗi phương tiện trên kịch bản LOW, NS, EW trung bình khoảng 29%, 16%, 27%, kịch bản HIGH trung bình khoảng 12%. Kịch bản HIGH, không cải thiện được nhiều bởi vì với tình huống giao thông đông từ 4 hướng thì rất khó để cải thiện thời gian chờ. Như vậy, khi đánh giá lên 4 kịch bản cố định, Agent 1 đã cải thiện trung bình khoảng 21% so với 2 hệ thống STL.

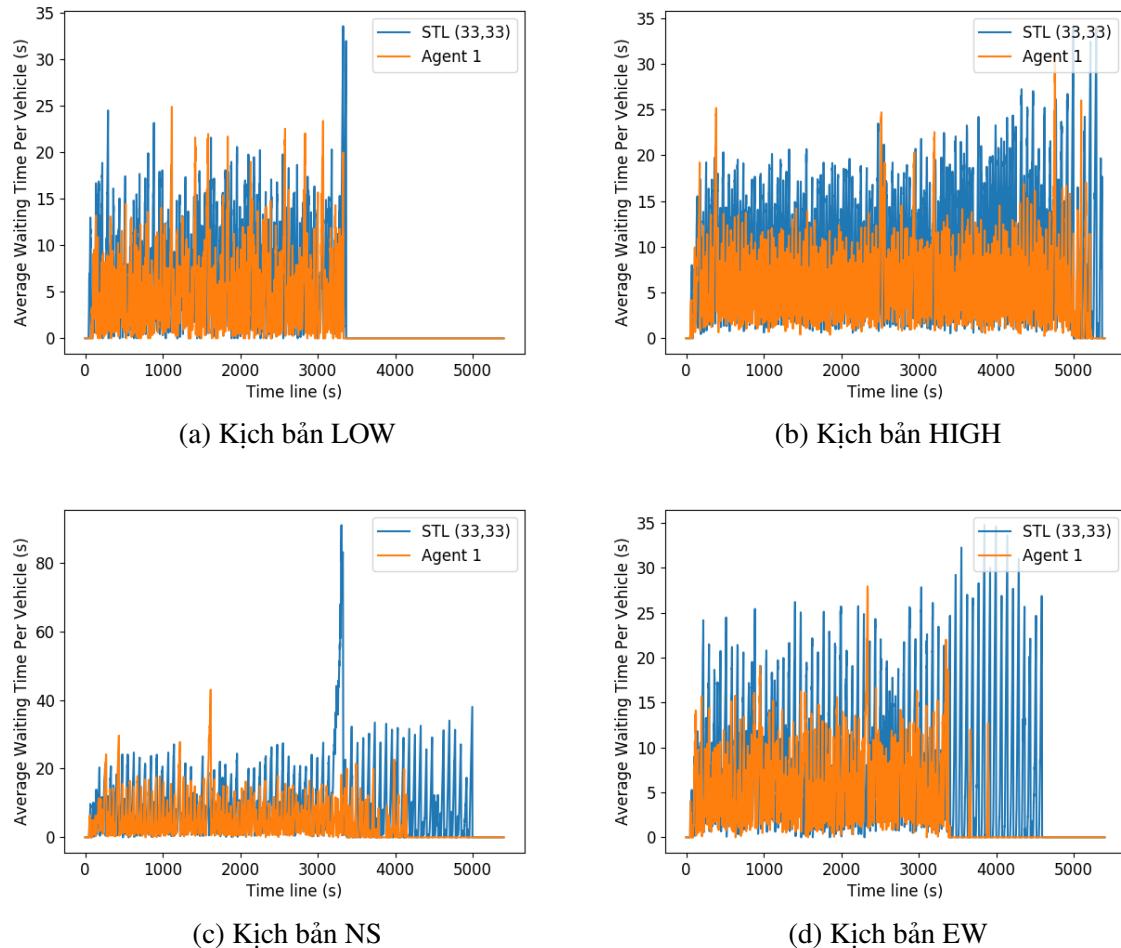
Thời gian chờ trung bình ở mỗi giây (AWT_t)

Khi đánh giá Agent 1 lên một kịch bản, AWT_t cho ta thấy được sự so sánh trực quan hơn mức độ cải thiện của Agent 1 với hệ thống STL. Hình 6.6 là kết quả so sánh AWT_t trong toàn bộ thời gian của 4 loại kịch bản giữa Agent 1 và hệ thống STL (33,33).

Quan sát hình 6.6, ta có thể thấy được đa số thời gian trong 4 kịch bản, Agent đều đã cải thiện được thời gian chờ trung bình của các phương tiện. Ví dụ trong hình 6.6c, Agent 1 không những giảm thời gian chờ trung bình đáng kể so với hệ thống STL (33,33) mà còn giải phóng hết xe sớm tại thời điểm khoảng 4200 giây, trong khi đó hệ thống STL phải đến thời điểm khoảng 5000 giây mới giải phóng hết xe của kịch bản (chậm hơn khoảng 13 phút).

Đánh giá AWT của Agent 1 trên 5 kịch bản ngẫu nhiên

Ngoài việc đánh giá Agent 1 trên kịch bản mà nó được huấn luyện, chúng tôi còn tạo ra 5 kịch bản ngẫu nhiên và đánh giá Agent 1 trên nó. Qua đó, chúng tôi có thể biết được đối với



Hình 6.6: So sánh Agent 1 và hệ thống STL (33,33)

những kịch bản ngẫu nhiên, Agent có thể cải thiện được bao nhiêu so với 2 hệ thống STL. Bảng 6.7 là kết quả so sánh giữa Agent 1 và 2 hệ thống STL trên 5 kịch bản ngẫu nhiên này.

Dựa vào bảng 6.7, ta có thể thấy Agent 1 đã cải thiện hiệu suất khá tốt so với 2 hệ thống STL. Độ cải thiện cao nhất là 27% và thấp nhất là 10%. Độ cải thiện trung bình của cả 5 kịch bản ngẫu nhiên là **20%**, thấp hơn 4 kịch bản cố định (**21%**).

Nhận xét: do được huấn luyện với kịch bản cố định nên Agent 1 có được độ cải thiện trung bình khoảng **21%** khi đánh giá trên chúng. Tuy nhiên, khi đánh giá với những kịch bản hoàn toàn ngẫu nhiên, Agent 1 vẫn có kết quả khá tốt, cải thiện trung bình khoảng **20%**. Qua đó cho

Kịch bản	STL (33,33)	STL (40,40)	Agent 1	Độ cải thiện
Ngẫu nhiên 1	16.03	18.59	13.47	16% - 28%
Ngẫu nhiên 2	20.12	21.74	17.19	15% - 21%
Ngẫu nhiên 3	16.71	17.87	13.04	22% - 27%
Ngẫu nhiên 4	14.09	15.76	11.95	15% - 24%
Ngẫu nhiên 5	18.17	19.44	16.32	10% - 16%

Bảng 6.7: So sánh AWT của 2 hệ thống STL với Agent 1 trên 5 kịch bản ngẫu nhiên

thấy, Agent 1 vẫn có tính tổng quát khi kiểm tra trên các kịch bản ngẫu nhiên.

6.4.2 Chiến lược 2

Thời gian chờ trung bình trong toàn bộ kịch bản (AWT)

Tương tự như chiến lược 1, chúng tôi cũng ghi lại AWT (kịch bản huấn luyện ngẫu nhiên) của Agent trong quá trình huấn luyện để thấy được AWT được cải thiện như thế nào. Tuy nhiên, vì toàn bộ kịch bản là ngẫu nhiên nên chúng tôi sẽ không so sánh với AWT của Agent với AWT của kịch bản cố định. Hình 6.7, 6.8, 6.9, 6.10 là kết quả ghi lại AWT của Agent trong quá trình huấn luyện (200 episodes).

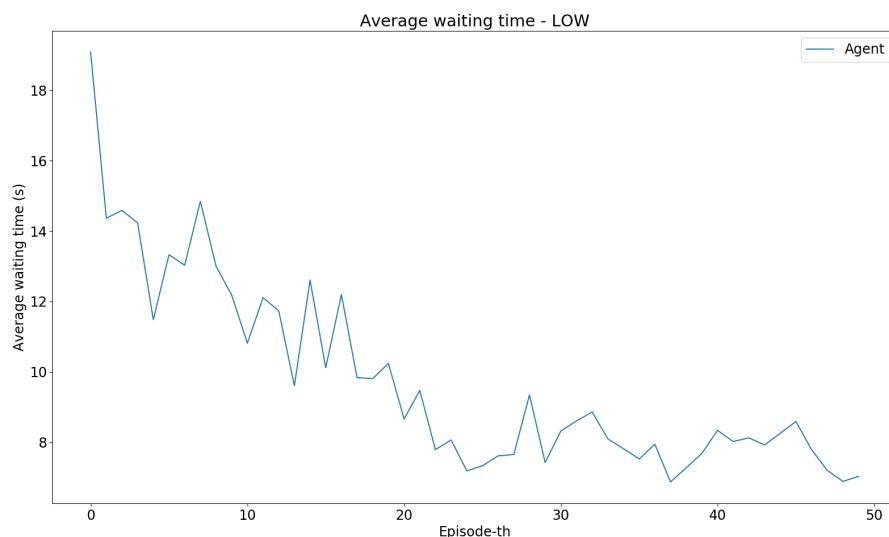
Nhận xét: mặc dù kịch bản có thể bị thay đổi ngẫu nhiên trong lúc huấn luyện, nhưng Agent vẫn có thể học và giảm được thời gian chờ trung bình ở cả 4 kịch bản.

Để so sánh 2 Agent được huấn luyện bằng **chiến lược 1** và **chiến lược 2**. Chúng tôi cũng sẽ đo kết quả AWT của Agent sau khi huấn luyện của chiến lược 2 (tạm gọi là Agent 2) lên **4 kịch bản cố định** đã dùng để huấn luyện trong chiến lược 1. Bảng 6.8 là kết quả so sánh của Agent 2 với 2 hệ thống STL trên kịch bản cố định.

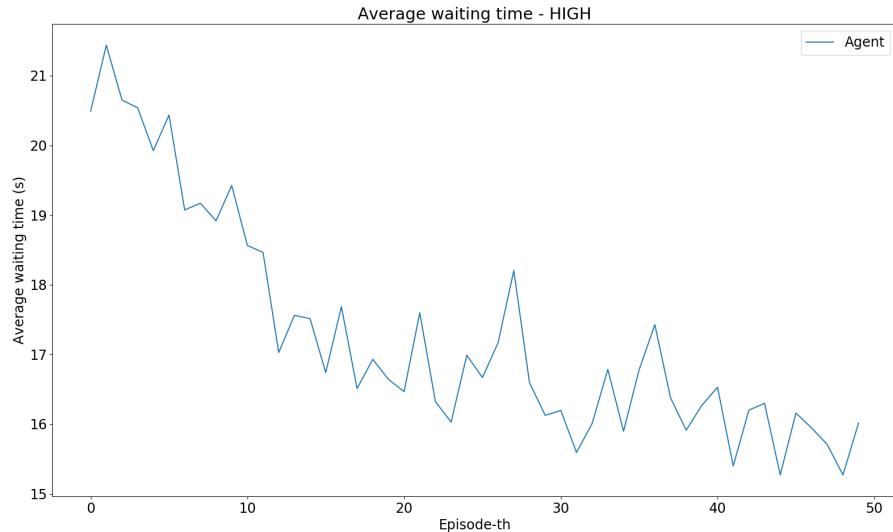
Nhận xét: dựa vào bảng kết quả 6.6 và bảng 6.8, chúng tôi thấy được Agent 2 cải thiện kịch bản LOW, NS tốt hơn nhiều so với Agent 1, và Agent 1 cải thiện kịch bản HIGH, EW tốt hơn Agent 2. Chúng tôi cũng tính được độ cải thiện trung bình của cả 4 kịch bản của Agent 2 là **21%** trong khi độ cải thiện trung bình 4 kịch bản cố định của Agent 1 là **21%**. Mặc dù Agent 2 được huấn luyện với kịch bản ngẫu nhiên, nhưng khi so sánh với các kịch bản cố định, Agent 2 vẫn cho kết quả tương đương với Agent 1.

Thời gian chờ trung bình ở mỗi giây (AWT_t)

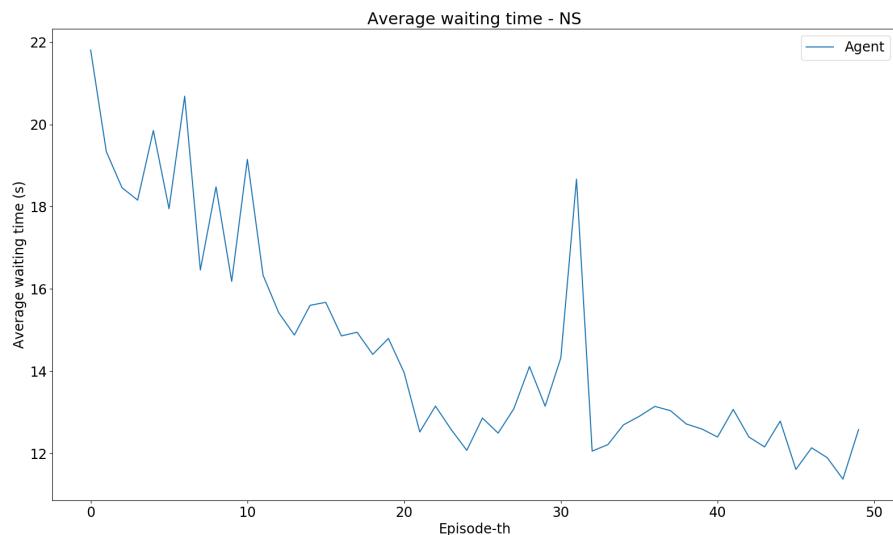
Tương tự như đánh giá chiến lược 1, Agent 2 cũng được so sánh độ đo AWT_t theo thời gian lên các kịch bản cố định. Hình 6.11 là kết quả được ghi lại khi so sánh Agent 2 với hệ thống STL (33,33) trên độ đo AWT_t.



Hình 6.7: AWT trong quá trình huấn luyện theo chiến lược 2 - LOW



Hình 6.8: AWT trong quá trình huấn luyện theo chiến lược 2 - HIGH



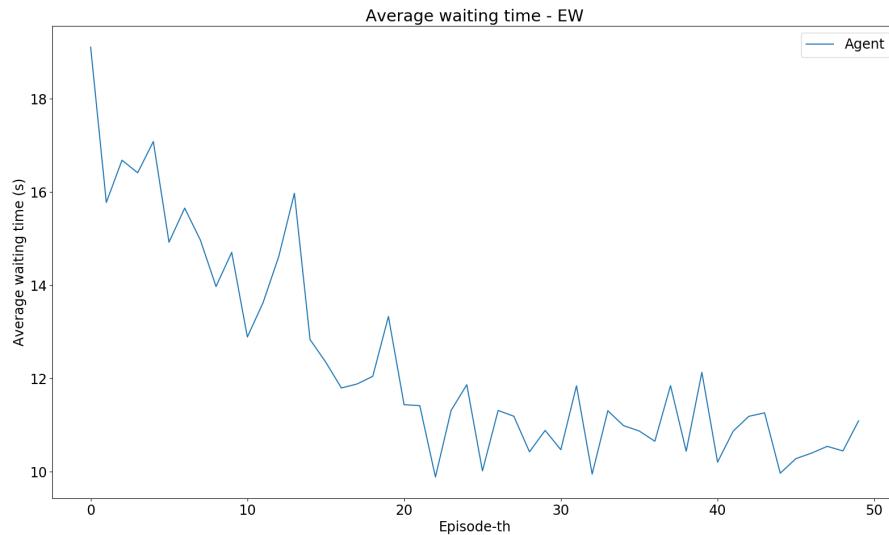
Hình 6.9: AWT trong quá trình huấn luyện theo chiến lược 2 - NS

Đánh giá AWT của Agent 2 trên 5 kịch bản ngẫu nhiên

Agent 2 cũng được đánh giá trên 5 kịch bản ngẫu nhiên. Bảng 6.9 là kết quả so sánh AWT giữa Agent 2 và 2 hệ thống STL. So sánh 2 bảng 6.7, 6.9 kết quả của Agent 1 và Agent 2 lên 5 kịch bản ngẫu nhiên. Agent 2 đã cải thiện tốt hơn Agent 1 ở 4 kịch bản ngẫu nhiên 2, 3, 4 ,5 và Agent 1 tốt hơn ở kịch bản ngẫu nhiên 1. Vậy nên, với chiến lược huấn luyện 2 chúng tôi đã có được Agent 2 có thể điều khiển giao thông tốt hơn Agent 1 ở trong các kịch bản giao thông ngẫu nhiên.

Kịch bản	STL (33,33)	STL (40,40)	Agent 2	Độ cải thiện
LOW	11.41	13.41	8.38	26% - 37%
HIGH	20.37	21.48	19.09	6% - 11%
NS	16.92	18.24	13.19	22% - 28%
EW	16.79	18.91	14.46	14% - 24%

Bảng 6.8: So sánh AWT của 2 hệ thống STL với Agent 2



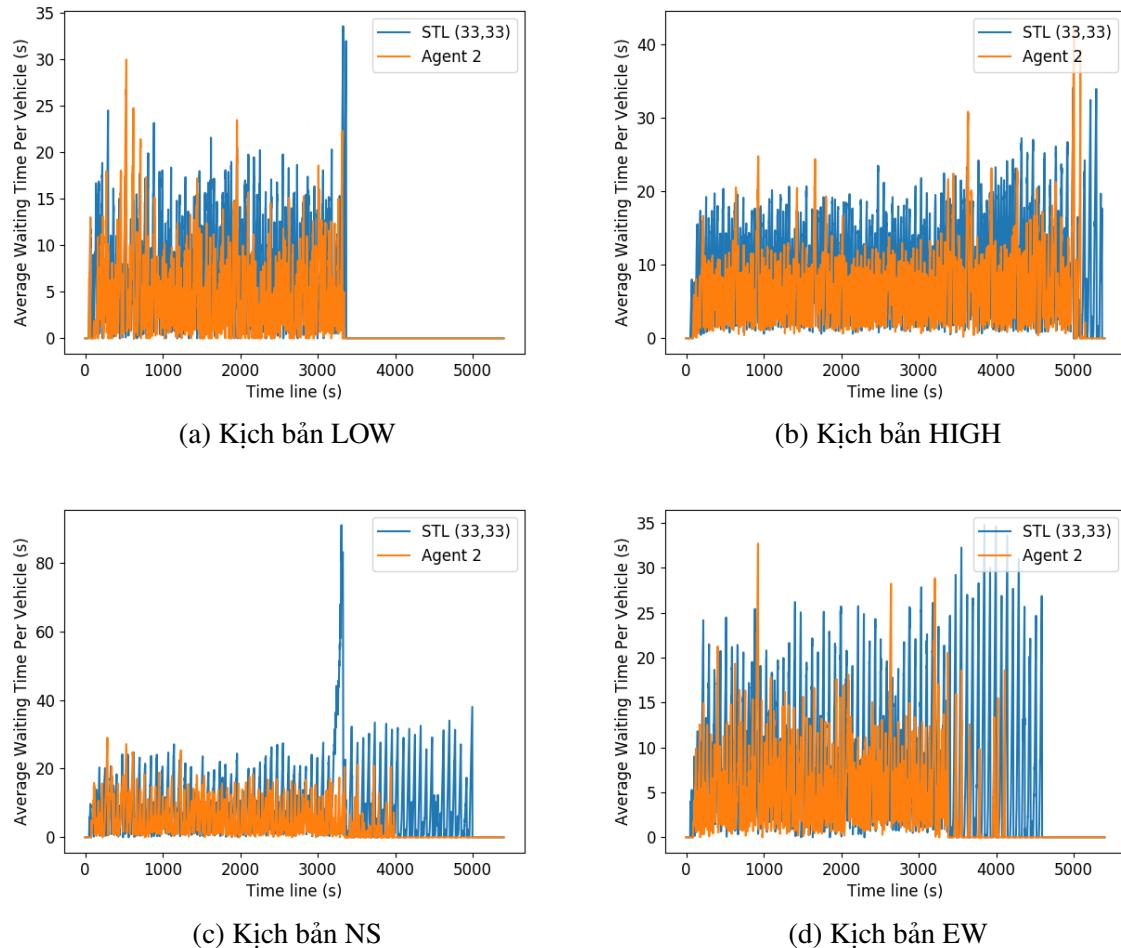
Hình 6.10: AWT trong quá trình huấn luyện theo chiến lược 2 - EW

6.4.3 Nhận xét

Sau khi đã có kết quả đo được cho các chiến lược huấn luyện với các kịch bản ngẫu nhiên, kịch bản cố định. Chúng tôi kết luận rằng, bằng chiến lược huấn luyện 2, Agent có thể học được nhiều kinh nghiệm hơn và có thể giải quyết các kịch bản giao thông ngẫu nhiên tốt hơn chiến lược 1.

Kịch bản	STL (33,33)	STL (40,40)	Agent 2	Độ cải thiện
Ngẫu nhiên 1	16.03	18.59	15.24	5% - 18%
Ngẫu nhiên 2	20.12	21.74	16.23	19% - 25%
Ngẫu nhiên 3	16.71	17.87	12.00	28% - 33%
Ngẫu nhiên 4	14.09	15.76	11.34	20% - 28%
Ngẫu nhiên 5	18.17	19.44	16.13	11% - 17%

Bảng 6.9: So sánh AWT của 2 hệ thống STL với Agent 2 trên 5 kịch bản ngẫu nhiên



Hình 6.11: So sánh Agent 2 và hệ thống STL (33,33)

6.4.4 Kết quả huấn luyện ngã ba

Khi xây dựng mô hình huấn luyện trên ngã ba, chúng tôi cũng nhận được kết quả tương tự về độ hiệu quả của Agent 1 và Agent 2. Cụ thể ở bảng 6.10 là số liệu khi so sánh trên kịch bản ngẫu nhiên và bảng 6.11 là số liệu so sánh trên kịch bản cố định.

Theo đó, Agent huấn luyện bằng chiến lược 2 thắng 3 trên tổng số 5 kịch bản ngẫu nhiên so với Agent huấn luyện bằng chiến lược 1 chứng minh được tính tổng quát được cải thiện khi dùng chiến lược 2. Một điểm đặc biệt nữa ở đây là ngã ba chúng tôi ánh xạ nhiều xe hơn vào trạng thái nên độ hiệu quả của mô hình được cải thiện đáng kể hơn so với ngã tư.

Kịch bản	STL (33,33)	STL (40,40)	Agent 1	Độ cải thiện 1	Agent 2	Độ cải thiện 2
Ngẫu nhiên 1	20.63	31.28	12.80	38% - 59%	14.16	31% - 55%
Ngẫu nhiên 2	48.05	51.65	46.66	3% - 10%	35.39	26% - 31%
Ngẫu nhiên 3	22.78	29.52	11.57	49% - 61%	11.28	51% - 61%
Ngẫu nhiên 4	44.03	48.75	13.94	68% - 71%	20.35	54% - 58%
Ngẫu nhiên 5	61.65	60.95	53.15	13% - 14%	52.08	15% - 16%

Bảng 6.10: So sánh AWT của 2 hệ thống STL kịch bản ngẫu nhiên giao lộ ngã ba

Kịch bản	STL (33,33)	STL (40,40)	Agent 1	Độ cải thiện 1	Agent 2	Độ cải thiện 2
LOW	11.97	12.81	8.61	28% - 33%	7.32	39% - 43%
HIGH	66.30	67.97	57.35	14% - 16%	54.28	18% - 20%
NS	51.15	57.06	35.54	31% - 38%	36.05	30% - 37%
EW	52.52	54.94	21.90	58% - 60%	14.88	72% - 73%

Bảng 6.11: So sánh AWT của 2 hệ thống STL kịch bản ngẫu nhiên giao lộ ngã ba

Kết quả này cũng góp phần nào trong việc khẳng định mức độ cải thiện của mô hình mà chúng tôi xây dựng và cũng là điều chứng minh tính tổng quát của mô hình có thể khác nhau nhờ vào chiến lược huấn luyện khác nhau.

Chương 7

Tổng kết

Trong chương này, chúng tôi sẽ tổng kết lại toàn bộ những gì đã nghiên cứu, hiện thực và kết quả đạt được. Bên cạnh những kết quả đó, nhóm cũng sẽ nêu ra những khó khăn và hạn chế của đề tài mà nhóm gặp phải. Cuối cùng là hướng phát triển của đề tài trong tương lai.

7.1 Kết quả đạt được

Cho đến bây giờ, chúng tôi đã trải qua hai học kì với hai giai đoạn đề cương và luận văn tốt nghiệp. Đó là khoảng thời gian không quá dài cũng không quá ngắn, nhưng chúng tôi cũng đã cố gắng nghiên cứu, hiện thực để đạt được những kết quả nhất định.

Trước hết là kiến thức về kĩ thuật học tăng cường, các giải thuật Q-Learning, Deep-Q-Learning, kiến trúc mạng CNN, mô hình Markov Decision Process và các kĩ thuật tiên tiến giúp tăng cường hiệu suất của học tăng cường như Priority Experience Replay, Double DQN, Dueling DQN.

Điều thứ hai nhóm đạt được ở phần nghiên cứu này đó là cách áp dụng những kiến thức đó vào mục tiêu có cải thiện vấn đề thực tế, cụ thể là giải quyết bài toán ùn tắc giao thông. Cùng với sự kiên trì và nỗ lực cộng thêm một chút may mắn, nhóm đã tìm ra được mô hình có thể giải quyết mục tiêu này ở các giao lộ ngã ba và ngã tư. Đồng thời, bản đồ mà chúng tôi mô phỏng cũng đã tính đến vấn đề xe cộ di chuyển không thẳng hàng nên phần nào cũng chứng minh được nó có thể đạt kết quả tương tự khi áp dụng được vào thực tế.

Để hiện thực mô hình đó, nhóm cũng đã tìm hiểu về công cụ mô phỏng SUMO giúp xây dựng mô phỏng các loại giao lộ (ngã ba, ngã tư) và các APIs giúp trích xuất thông tin từ giao lộ đó. Chúng tôi đã hiện thực mô hình, giải thuật và tiến hành rất nhiều thử nghiệm để đạt được kết quả tốt nhất. Kết quả cuối cùng nhóm đạt được là đã huấn luyện thành công hệ thống điều khiển giao thông trên hai mô phỏng giao lộ (ngã ba và ngã tư) với độ cải thiện lên khoảng 20% - 30% so với các hệ thống giao thông cố định, đó cũng là mục tiêu mà nhóm đặt ra từ đầu.

Và điều quan trọng nhất, kết quả lớn nhất mà chúng tôi nhận được đó là kinh nghiệm huấn luyện, đánh giá, xây dựng một mô hình. Qua quá trình nghiên cứu, nhóm đã biết được việc huấn luyện không phải chỉ tuân theo bài báo là đủ, có những điều mà bài báo không nhắc đến khiến nhóm phải tự tìm tòi và hiểu theo ý mình. Điều này tuy mất khá nhiều thời gian nhưng nó lại đem lại nhiều cách hiện thực khác nhau và việc đánh giá cũng sẽ có thêm nhiều lựa chọn.

Không những thế, một trong những kinh nghiệm khác mà chúng tôi đạt được xuất phát từ hạn chế của mô phỏng SUMO, khi hai phương tiện đi trái chiều nhau và đụng nhau, nó sẽ đứng một lúc lâu mới thoát khỏi tình trạng Deadlock, điều này sẽ khiến cho Agent học sai. Ngoài ra, Không phải cứ sinh ngẫu nhiên kịch bản là có thể huấn luyện được mà cần phải xét thật kĩ rằng với kịch bản đó Agent sẽ học được những gì. Và khi ta chọn được kịch bản tốt, tính tổng quát

của mô hình sẽ tăng cao và việc áp dụng mô hình đó cho các giao lộ có cùng loại nhưng khác lưu lượng xe thì đều đạt được hiệu quả tốt.

7.2 Hạn chế

Mặc dù hiện đề tài còn gói gọn trong phạm vi mô phỏng bằng công cụ SUMO, nhưng với mô hình mà chúng tôi huấn luyện trên ngã ba và ngã tư, nó sẽ là nền tảng và hi vọng để có thể đặt ra mục tiêu lớn hơn đó là ứng dụng vào thực tế. Tuy kết quả đạt được đúng với mục tiêu mà chúng tôi đặt ra nhưng trong quá trình thực hiện luận văn, chúng tôi nhận ra hệ thống này vẫn còn nhiều hạn chế không thể tránh khỏi, cụ thể:

1. Hệ thống điều khiển không hiển thị thời gian đèn do cách mô hình hành động làm giảm sự trải nghiệm của người đi đường.
2. Mô hình mới được áp dụng vào hai giao lộ ngã ba và ngã tư.
3. Mô hình chưa thể áp dụng trong thực tế. Trạng thái đầu vào của mô hình được xây dựng từ thông tin của SUMO cung cấp, nhưng để có trạng thái đầu vào từ giao thông thực tế thì nó sẽ cần một hệ thống khác (camera, cảm biến) và đó chính là thách thức đối với nhóm.
4. Dù có xây dựng được trạng thái thực tế, mô hình khó mà áp dụng được với tình trạng giao thông Việt Nam. Mặc dù đã giải quyết được vấn đề nhiều loại phương tiện và các phương tiện đều di chuyển không thẳng hàng nhưng vẫn chưa thể mô phỏng được các vấn đề như vượt đèn đỏ, ưu tiên loại xe khẩn cấp, tình huống có người đi bộ, các xe lấn làn. Các yếu tố này đều có thể ảnh hưởng đến hiệu suất của hệ thống và Agent hiện tại chưa học được những tình huống giao thông đó.

Ngoài những hạn chế đó ra là việc thất bại trong việc thử nghiệm mô hình với kĩ thuật Priority Experience Replay. Chúng tôi đã tìm hiểu và cách hiện thực tham khảo ở những nguồn khác nhau và thử nghiệm với mô hình này. Kết quả đạt được tương tự với mô hình cũ, không thấy được khả năng hội tụ nhanh của kĩ thuật này như lý thuyết đã biết. Nhóm dự đoán việc hiện thực đã thất bại và sẽ tìm cách giải quyết được vấn đề này trong tương lai.

7.3 Hướng phát triển

Kết quả đạt được là thế, song song với nó cũng có rất nhiều hạn chế mà chúng tôi gặp phải. Trong tương lai, nhóm đặt ra hướng tiếp cận với những mục tiêu cụ thể. Điều này sẽ làm rõ ràng hóa các vấn đề cần phải giải quyết cũng như hạn chế việc đi sai hướng trong quá trình nghiên cứu.

Nhiệm vụ trước mắt là tiếp tục tìm ra phương án hiện thực kĩ thuật Priority Experience Replay và tiến hành thử nghiệm. Kết quả mong muốn mô hình sẽ học nhanh hơn và tiết kiệm thời gian huấn luyện. Để làm được điều này, chúng tôi sẽ tìm hiểu một cách rõ ràng giải thuật và tham khảo một số code mẫu đồng thời nhờ sự trợ giúp của thầy hướng dẫn.

Ngoài việc áp dụng cho ngã ba với ngã tư, nhóm sẽ tiến hành hiện thực và thử nghiệm với nhiều loại giao lộ khác nhau như ngã năm, ngã sáu, vòng xoay. Không những thế, mô hình hành động cũng cần được nghiên cứu lại và tìm hiểu phương án khác để có thể dự đoán trước thời gian chờ mà vẫn đạt được hiệu quả cao, ít nhất là bằng với kết quả hiện tại.

Và để cho việc huấn luyện sát với thực tế hơn nữa, chúng tôi cũng sẽ tiếp tục nghiên cứu về mô phỏng SUMO và xác định các tính năng mà nó cung cấp có giải quyết được hay không những hạn chế thiếu thực tế đã trình bày ở mục trước. Trước mắt là tìm ra mô hình trạng thái mà Agent có thể nhận biết được ở đó có người đi bộ.

Cuối cùng, chúng tôi cũng hi vọng sau khi hoàn thành những mục tiêu trên, việc xây dựng hệ thống trích xuất hình ảnh từ camera, vận tốc từ máy đo, cảm biến, hệ thống xử lý ảnh và sinh trạng thái đầu vào cho mô hình cũng sẽ được nghiên cứu và giải quyết vấn đề lớn lao nhất và cũng là mục tiêu thực sự của bài nghiên cứu này đó là áp dụng vào thực tế để cải thiện tình hình giao thông so với hệ thống đèn giao thông cố định thời gian hiện nay.

Tài liệu tham khảo

- [1] Prateek Bajaj. What is reinforcement learning, 2018. <https://www.geeksforgeeks.org/what-is-reinforcement-learning/>.
- [2] Adit Deshpande. A Beginner's Guide To Understanding Convolutional Neural Networks, 2016. <https://adethpande3.github.io/adethpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/>.
- [3] Firdaouss Doukkali. Convolutional Neural Networks (CNN, or ConvNets), 2017. <https://medium.com/@phidaouss/convolutional-neural-networks-cnn-or-convnets-d7c688b0a207>.
- [4] Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc., 2010.
- [5] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Learning from demonstrations for real world reinforcement learning. *CoRR*, abs/1704.03732, 2017.
- [6] Sergios Karagiannakos. The idea behind actor-critics and how a2c and a3c improve them, 2018. https://sergiostkar.github.io/Actor_critics/.
- [7] Keras. Keras: The python deep learning library, 2019. <https://keras.io/>.
- [8] Xiaoyuan Liang, Xunsheng Du, Guiling Wang, and Zhu Han. Deep reinforcement learning for traffic light control in vehicular networks. *CoRR*, abs/1803.11115, 2018.
- [9] Seyed Sajad Mousavi, Michael Schukat, Peter Corcoran, and Enda Howley. Traffic light control using deep policy-gradient and value-function based reinforcement learning. *CoRR*, abs/1704.08883, 2017.
- [10] Matthias Plappert. keras-rl. <https://github.com/keras-rl/keras-rl>, 2016.
- [11] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. 2016.
- [12] Thomas Simonini. Diving deeper into Reinforcement Learning with Q-Learning, 2018. <https://medium.freecodecamp.org/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe/>.
- [13] Thomas Simonini. Improvements in deep q learning: Dueling double dqn, prioritized experience replay, and fixed q-targets, 2018. <https://medium.freecodecamp.org/improvements-in-deep-q-learning-dueling-double-dqn-prioritized-experience-replay-and-fixed-58b130cc5682>.
- [14] Thomas Simonini. An introduction to deep q-learning: let's play doom, 2018. <https://medium.freecodecamp.org/an-introduction-to-deep-q-learning-lets-play-doom-54d02d8017d8>.

-
- [15] Thomas Simonini. An introduction to reinforcement learning, 2018. <https://medium.freecodecamp.org/an-introduction-to-reinforcement-learning-4339519de419>.
 - [16] SUMO. Netedit tutorial, 2019. <https://sumo.dlr.de/wiki/NETEDIT>.
 - [17] Traci. Traci introduction, 2019. <https://sumo.dlr.de/wiki/TraCI>.
 - [18] Andrea Vidali. Simulation of traffic scenario controlled by a deep reinforcement learning agent, 2019. https://www.dropbox.com/s/aqhdp0q6qhp8q9/780747_Vidali_tesi.pdf?dl=0.
 - [19] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.
 - [20] Lilian Weng. Policy gradient algorithms, 2018. <https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html>.