Multi-Scale Deep Reinforcement Learning for Real-Time 3D-Landmark Detection in CT Scans

Florin C. Ghesu[1,2], Bogdan Georgescu[2], Yefeng Zheng[2], Sasa Grbic[2], Andreas Maier[1], Joachim Hornegger[1], and Dorin Comaniciu[2]

Presented By:
Anmol Sharma[3]

[1]Friedrich Alexander-Universitat Erlangen-Nurnberg

[2]Siemens Healthineers

[3]Medical Image Analysis Laboratory
School of Computing Science
Simon Fraser University
Burnaby, Canada

March 14, 2018

Outline

Outline

Outline

Problem Statement

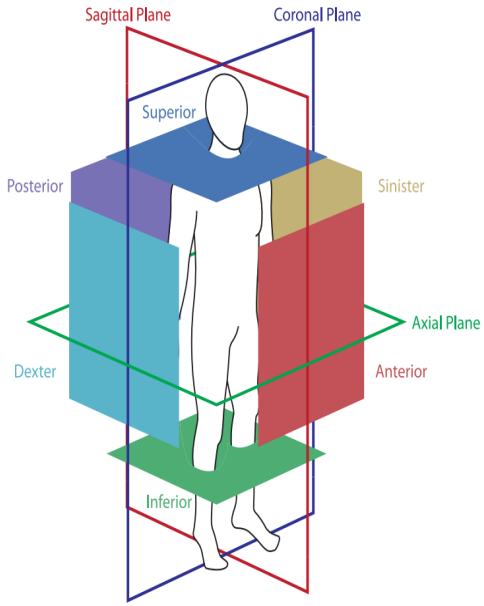
Data and Objects of Interest

- Detection of anatomical structures in CT scans.

Problem Statement

Data and Objects of Interest

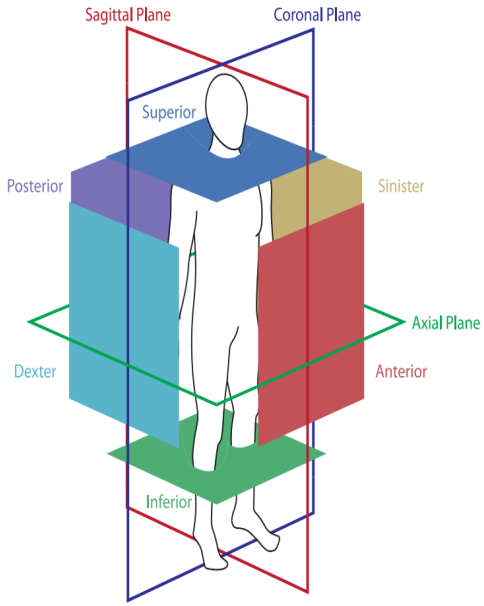
- Detection of anatomical structures in CT scans.



Problem Statement

Data and Objects of Interest

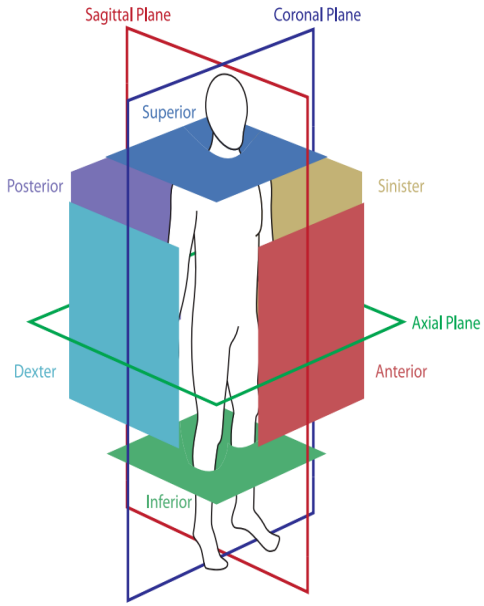
- Detection of anatomical structures in CT scans.
- Anatomical structures include:



Problem Statement

Data and Objects of Interest

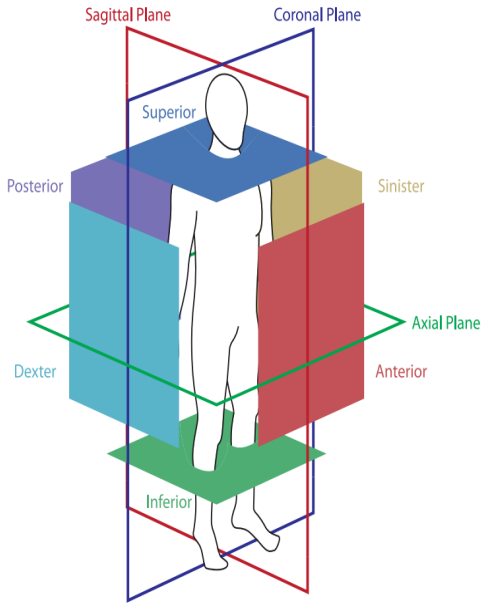
- Detection of anatomical structures in CT scans.
- Anatomical structures include:
  - Left and right kidneys.



Problem Statement

Data and Objects of Interest

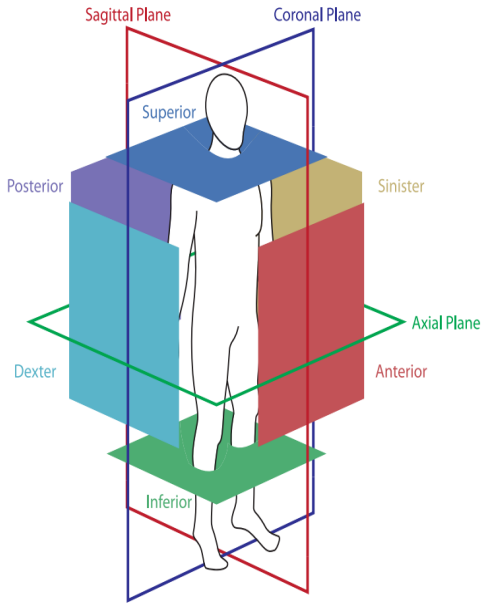
- Detection of anatomical structures in CT scans.
- Anatomical structures include:
  - Left and right kidneys.
  - Left and right hip bones.



Problem Statement

Data and Objects of Interest

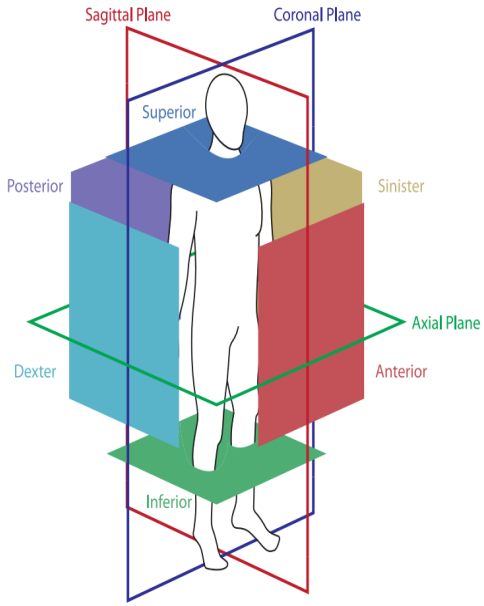
- Detection of anatomical structures in CT scans.
- Anatomical structures include:
  - Left and right kidneys.
  - Left and right hip bones.
  - Bronchial bifurcation.



Problem Statement

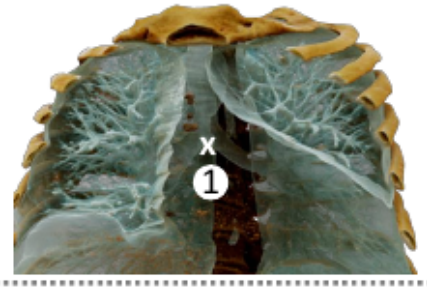
Data and Objects of Interest

- Detection of anatomical structures in CT scans.
- Anatomical structures include:
  - Left and right kidneys.
  - Left and right hip bones.
  - Bronchial bifurcation.
  - And 5 more anatomical structures...

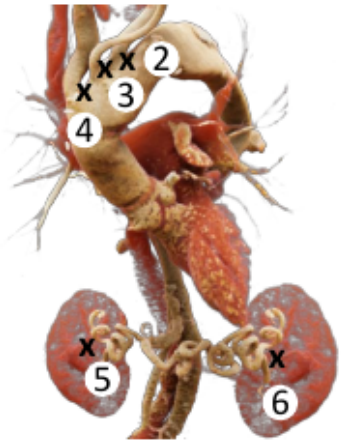


Problem Statement

Data and Objects of Interest



Bronchial Bifurcation



Aortic Arch Bifurcations

Kidneys

Hip Bones

Outline

Previous Work

Scanning Based Systems

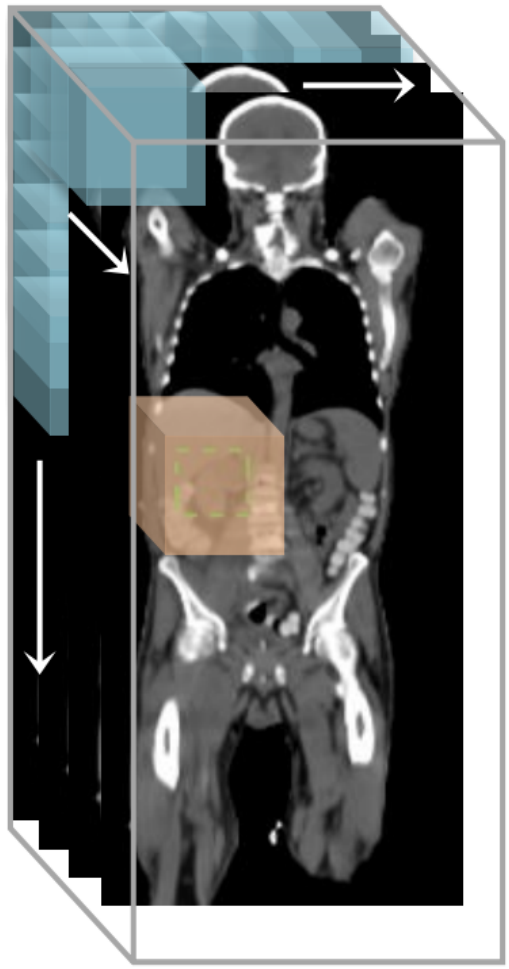


Figure: Systems that scan the image exhaustively or through a smarter region proposal based method.

Previous Work

End-to-End Systems

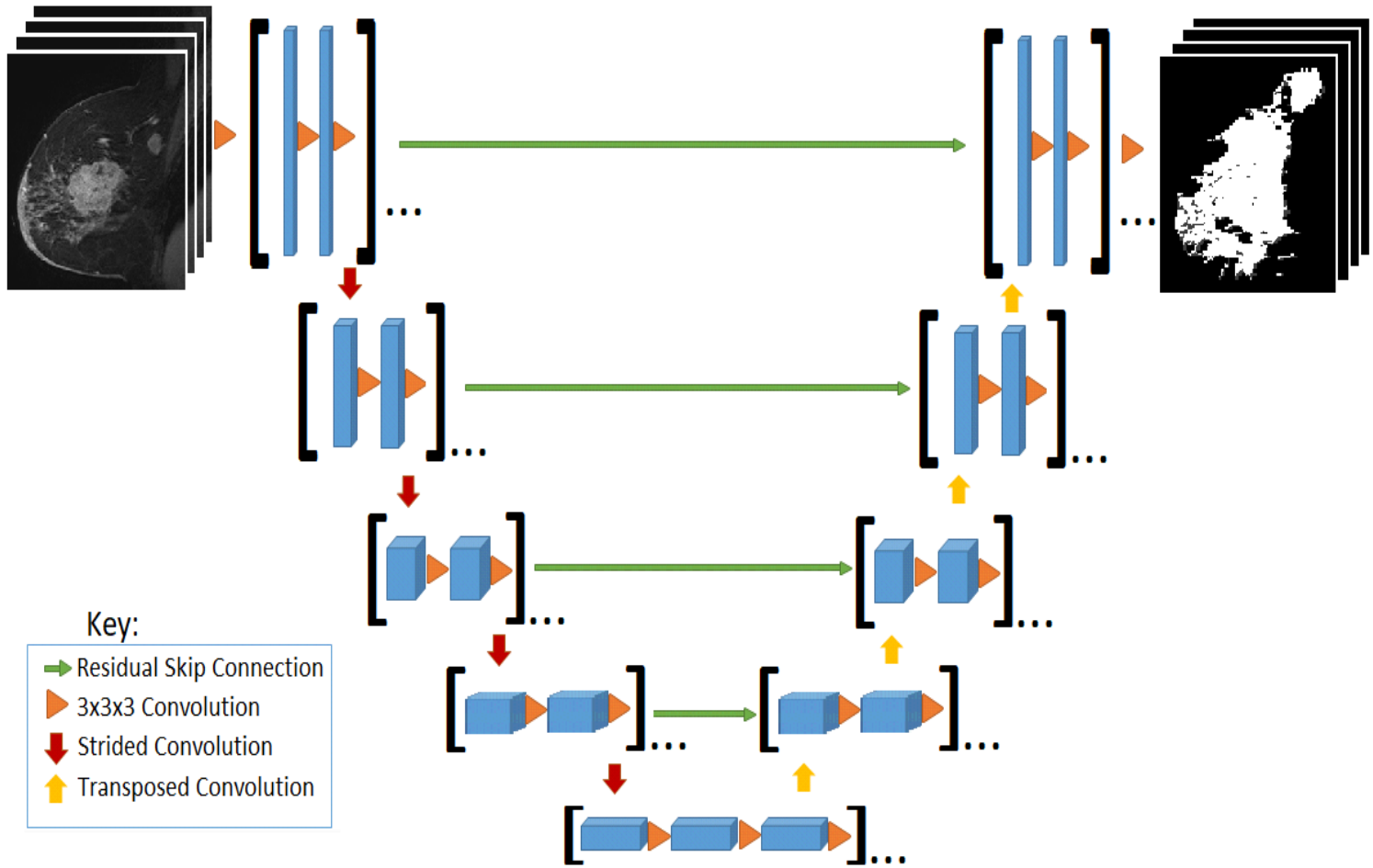


Figure: Systems which output the same sized localization or segmentation maps.

Previous Work

Regression-based Systems

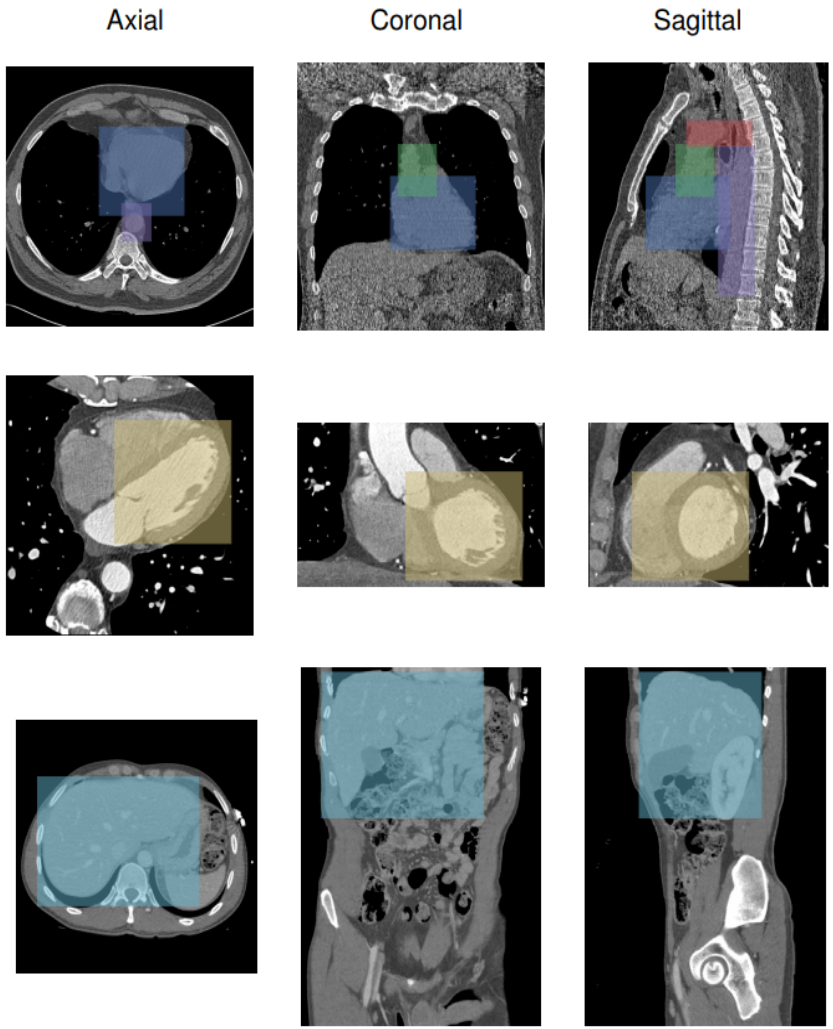


Figure: Systems directly output the position of the structure of interest.

Previous Work

Atlas-based Systems

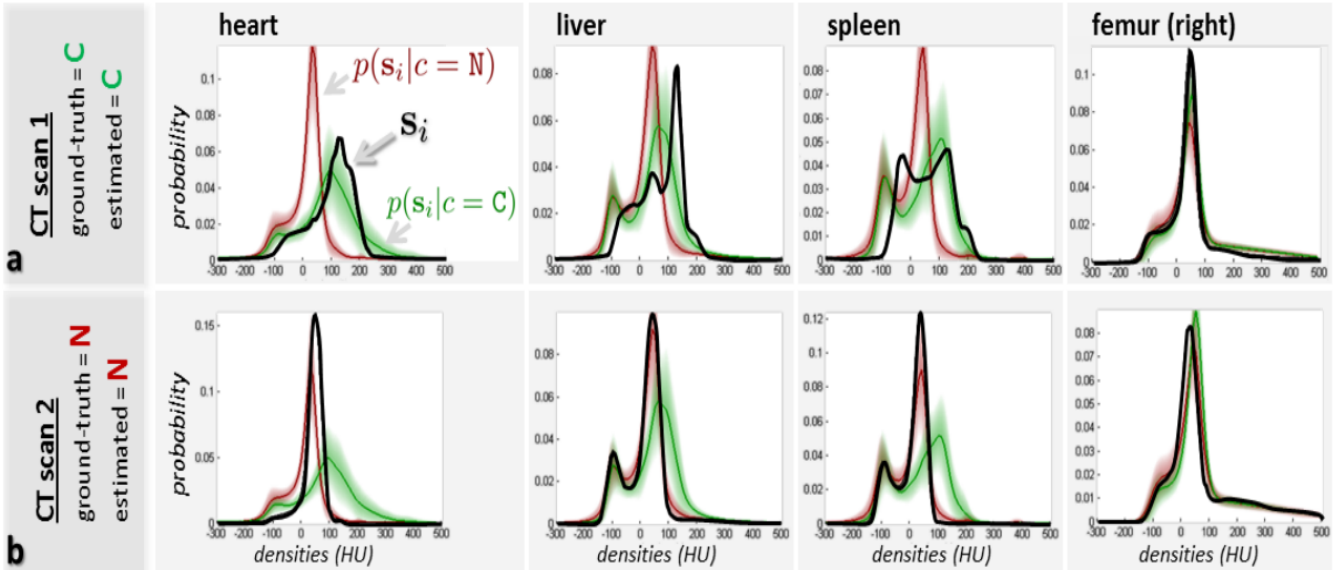


Figure: Systems that search the image using a learnt atlas. Also called Generative Models.

Outline

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a

.

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a behaviour learning task for an artificial agent.

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a behaviour learning task for an artificial agent.
- Model the

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a behaviour learning task for an artificial agent.
- Model the anatomy of appearance

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a behaviour learning task for an artificial agent.
- Model the anatomy of appearance + object search

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a behaviour learning task for an artificial agent.
- Model the anatomy of appearance + object search in a unified framework.

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a behaviour learning task for an artificial agent.
- Model the anatomy of appearance + object search in a unified framework.
  - Combine the knowledge of how the object looks

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a behaviour learning task for an artificial agent.
- Model the anatomy of appearance + object search in a unified framework.
  - Combine the knowledge of how the object looks
  - and where to find it.

Proposed Reformulation Overview

Description

- Reformulate the structure of interest detection (object detection) problem as a behaviour learning task for an artificial agent.
- Model the anatomy of appearance + object search in a unified framework.
  - Combine the knowledge of how the object looks
  - and where to find it.
- Model the problem as a Reinforcement Learning Agent.

Outline

Outline

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.
- Success is measured by a scalar reward signal.

Reinforcement Learning

In a Nutshell

- Theory and algorithms uniting multiple fields of machine learning, optimal control theory, psychology and neuroscience.
- A class of machine learning algorithms that learn by trial-and-error.
- Success is measured by a scalar reward signal.
- Goal: select actions to maximize future reward.

Reinforcement Learning

Atari Example

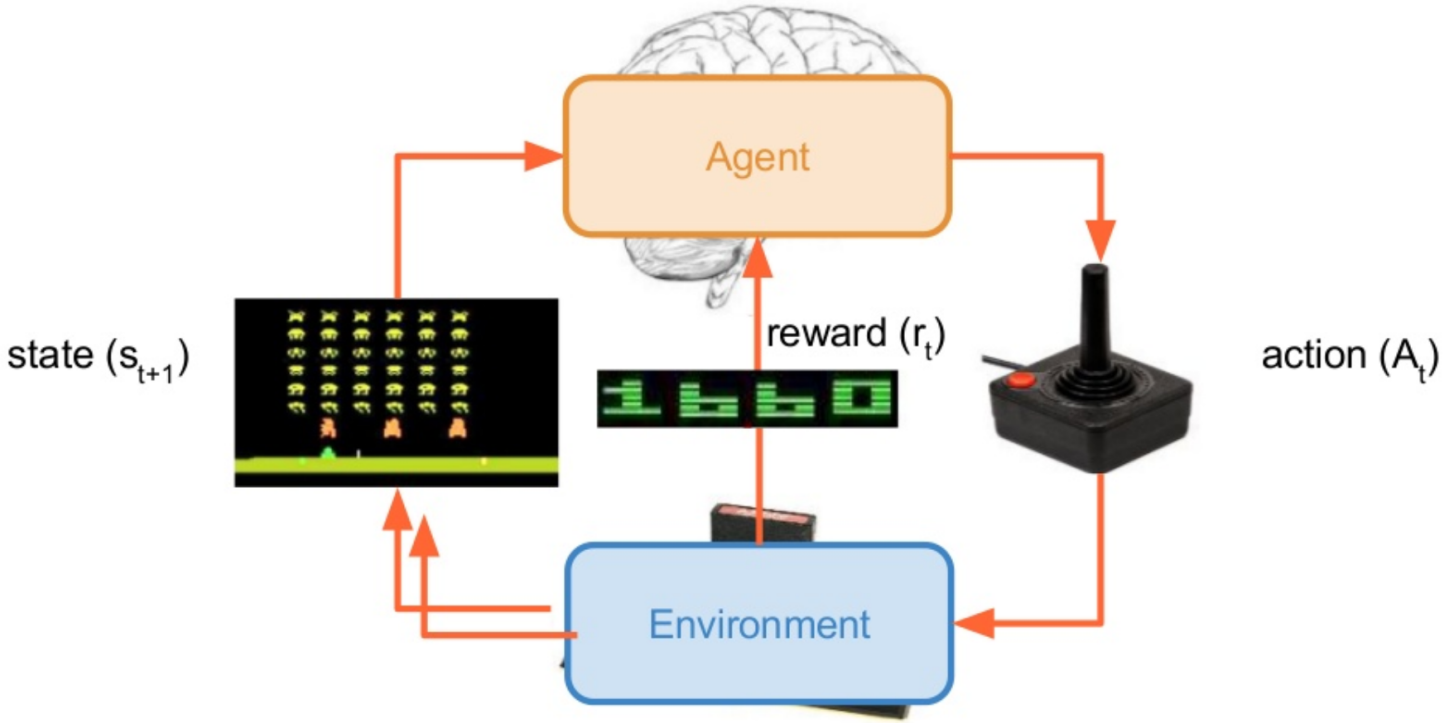


Figure: Agent-Environment interaction

Reinforcement Learning

Game Playing Agent Example

Figure: Mario trying to figure out life (like most grad students)

Reinforcement Learning

Game Playing Agent Example

Figure: Mario trying to figure out life (like most grad students)

- Think of the agent as a robot playing mario.

Reinforcement Learning

Game Playing Agent Example

Figure: Mario trying to figure out life (like most grad students)

- Think of the agent as a robot playing mario.
- Should the robot press jump button? (To take the big star?)

Reinforcement Learning

Game Playing Agent Example

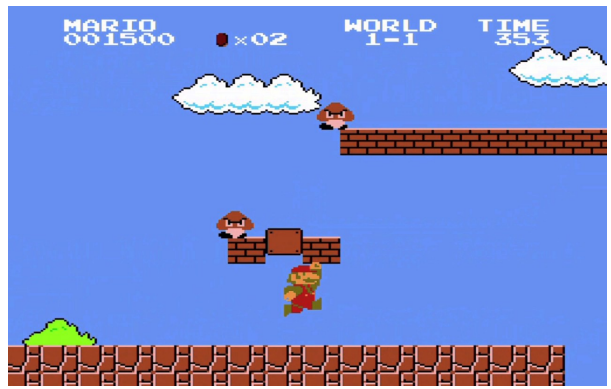
Figure: Mario trying to figure out life (like most grad students)

- Think of the agent as a robot playing mario.
- Should the robot press jump button? (To take the big star?)
- Or should it press forward button?

Outline

Policy Function

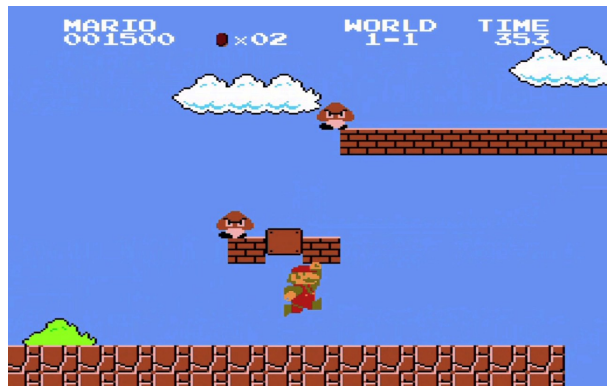
Predict What Action to Take

$$\pi\left(\text{  }\right) = \text{“Jump”}$$

Current State

Policy Function

Predict What Action to Take

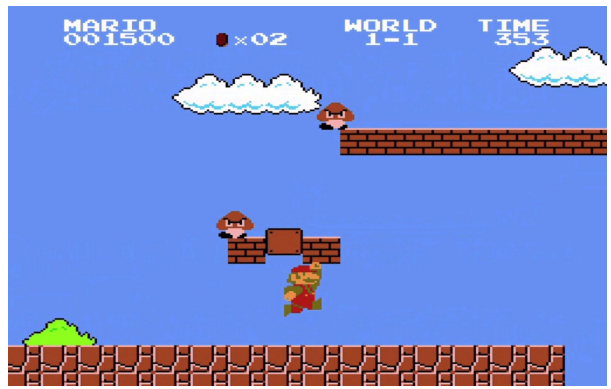
$$\pi\left(\text{  }\right) = \text{“Jump”}$$

Current State

- The state is input to the policy function as an image.

Policy Function

Predict What Action to Take

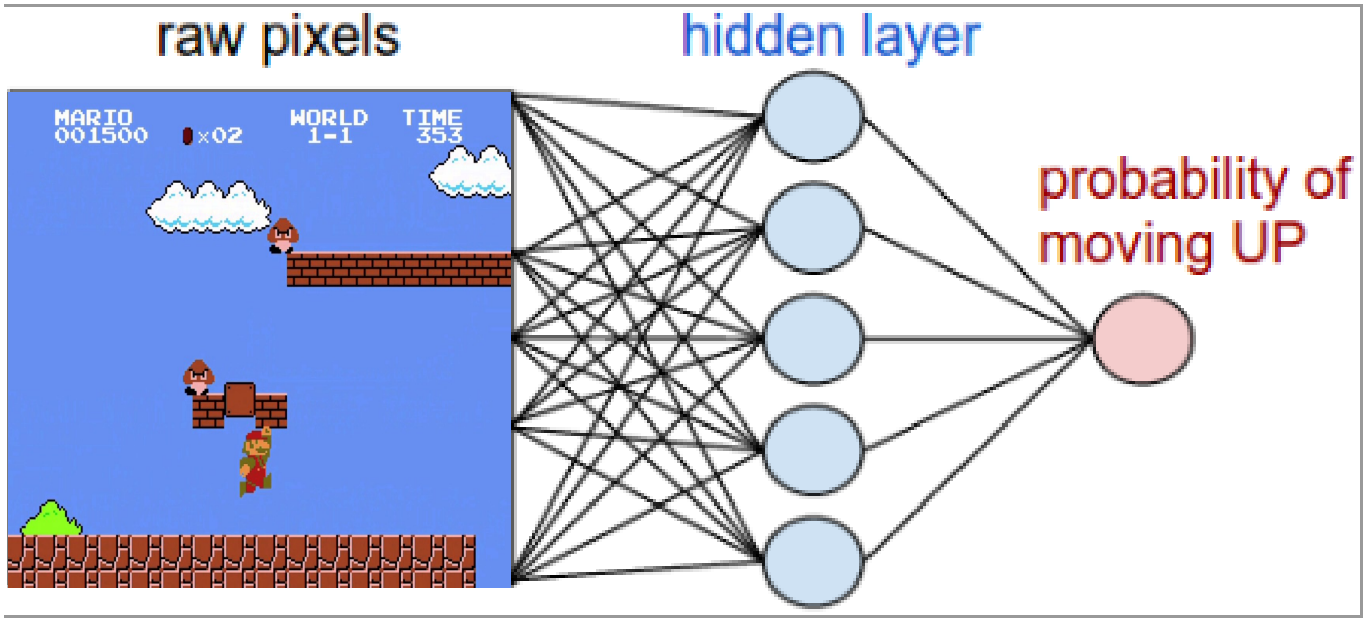
$$\pi\left(\text{  }\right) = \text{“Jump”}$$

Current State

- The state is input to the policy function as an image.
- That is, a “screenshot” of the game at a time stamp t is called the “state” at that time.

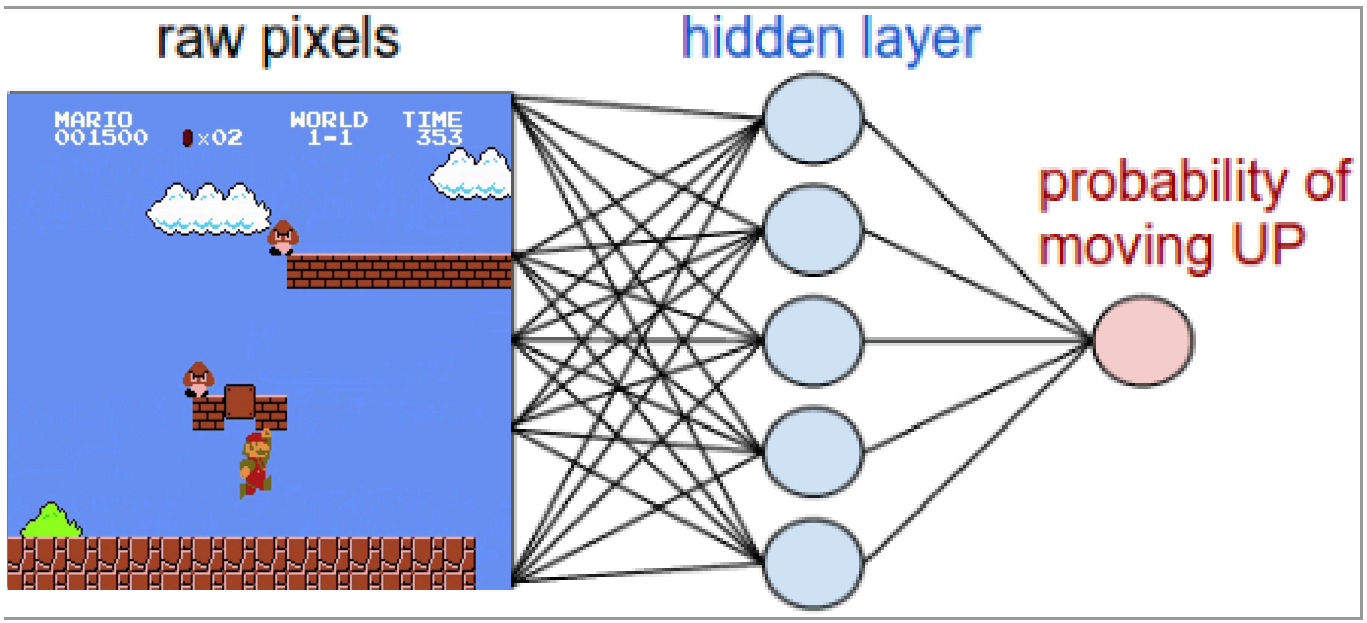
Policy Function

Representation



Policy Function

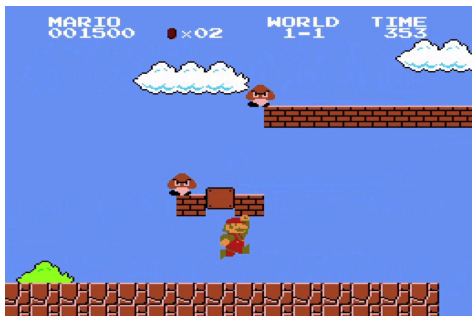
Representation



- Policy is represented as a Neural Network which takes the state (in pixels) as input and outputs probability of an action.

Reward Function

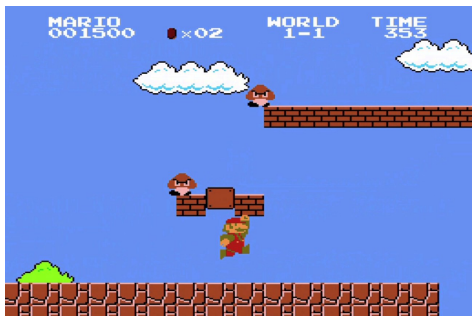
Compute Immediate Reward for an Action

$$\mathcal{R}\left(\text{  }, \text{"Jump"}\right) = 100$$

Current State Current Action

Reward Function

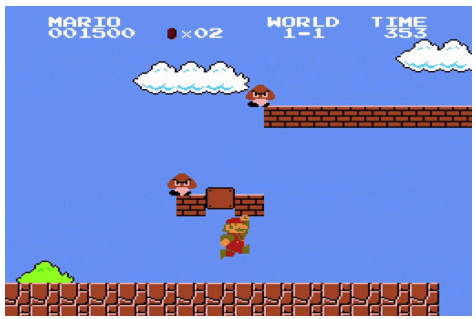
Compute Immediate Reward for an Action

$$\mathcal{R}\left(\underbrace{\text{  }}_{\text{Current State}}, \underbrace{\text{"Jump"}}_{\text{Current Action}}\right) = 100$$

- Reward function calculates immediate reward.

Reward Function

Compute Immediate Reward for an Action

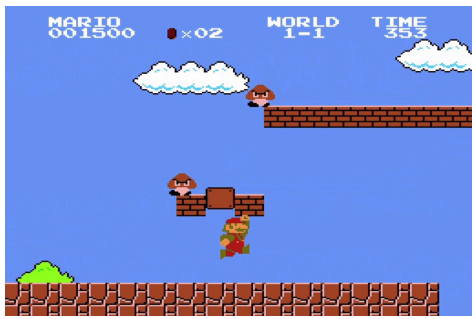
$$\mathcal{R}\left(\text{  }, \text{"Jump"}\right) = 100$$

Current State Current Action

- Reward function calculates immediate reward.
- For Atari games, it is predefined in the emulator (games decide how much reward you get).

Reward Function

Compute Immediate Reward for an Action

$$\mathcal{R}\left(\text{  }, \text{"Jump"}\right) = 100$$

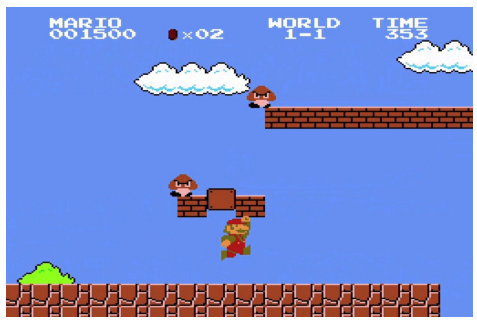
MARIO 001500 ×02 WORLD 1-1 TIME 353

Current State Current Action

- Reward function calculates immediate reward.
- For Atari games, it is predefined in the emulator (games decide how much reward you get).
- For real world custom problems, you have to define one yourself.

Value Function

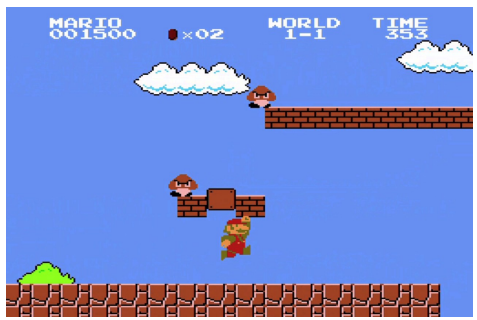
Predict Future Reward for an Action-State

$$\nu\left(\text{  }, \text{"Jump"}\right) \approx 5000$$

Current State Current Action

Value Function

Predict Future Reward for an Action-State

$$\nu\left(\text{  }, \text{"Jump"}\right) \approx 5000$$

Current State Current Action

- Value function predicts future reward.

Trajectory

Path Taken by Agent in Environment

Trajectory

Path Taken by Agent in Environment

$\tau = ($  "Jump", 10 ,  "Forward", 10 ,  , , , ,  "Jump", 10 $)$

$$\tau = (s_0, a_0, r_0, s_1, a_1, r_1 \ldots\ldots s_n, a_n, r_n)$$

Total Reward

Total Reward for a Trajectory

- We need to find the total reward that we received after this whole episode.

Total Reward

Total Reward for a Trajectory

- We need to find the total reward that we received after this whole episode.
- It is given as:

Total Reward

Total Reward for a Trajectory

- We need to find the total reward that we received after this whole episode.
- It is given as:

$$R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$$

Total Reward

Total Reward for a Trajectory

- We need to find the total reward that we received after this whole episode.
- It is given as:

$$R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$$

- Where γ is the "discount factor".

Total Reward

Total Reward for a Trajectory

- We need to find the total reward that we received after this whole episode.
- It is given as:

$$R(\tau) = \sum_{i=0}^{H-1} \gamma^i * r_i$$

- Where γ is the "discount factor".
  - Larger the value of γ, greater the influence of later rewards on the return.

Total Reward

Total Reward for a Trajectory

$$\tau = \Big(\text{  } \text{“Jump”, 10} \; , \; \text{  } \text{“Forward”, 10} \; , \; \text{  } \; , \; , \; , \; , \; \text{  } \text{“Jump”, 10} \Big)$$

$$\mathcal{R} = \gamma^0 * 10 \quad + \quad \gamma^1 * 10 \quad + \quad .... \quad \gamma^n * 10$$

Outline

Outline

Method

Deep Learning + Reinforcement Learning

Method

Deep Learning + Reinforcement Learning

- Proposed to apply Deep Reinforcement Learning for the task of detecting anatomical structures in 3D CT Scans.

Method

Deep Learning + Reinforcement Learning

- Proposed to apply Deep Reinforcement Learning for the task of detecting anatomical structures in 3D CT Scans.
- Combination of:
  - Convolutional Neural Network as the Policy Function and,

Method

Deep Learning + Reinforcement Learning

- Proposed to apply Deep Reinforcement Learning for the task of detecting anatomical structures in 3D CT Scans.
- Combination of:
  - Convolutional Neural Network as the Policy Function and,
  - Cognitive Model of Reinforcement Learning Framework.

Method

Deep Learning + Reinforcement Learning

- Proposed to apply Deep Reinforcement Learning for the task of detecting anatomical structures in 3D CT Scans.
- Combination of:
  - Convolutional Neural Network as the Policy Function and,
  - Cognitive Model of Reinforcement Learning Framework.
- A system capable of both:

Method

Deep Learning + Reinforcement Learning

- Proposed to apply Deep Reinforcement Learning for the task of detecting anatomical structures in 3D CT Scans.
- Combination of:
  - Convolutional Neural Network as the Policy Function and,
  - Cognitive Model of Reinforcement Learning Framework.
- A system capable of both:
  - Identifying an anatomical structure

Method

Deep Learning + Reinforcement Learning

- Proposed to apply Deep Reinforcement Learning for the task of detecting anatomical structures in 3D CT Scans.
- Combination of:
  - Convolutional Neural Network as the Policy Function and,
  - Cognitive Model of Reinforcement Learning Framework.
- A system capable of both:
  - Identifying an anatomical structure
  - Able to find the structure in a full 3D scan.

Method

Deep Learning + Reinforcement Learning

- Proposed to apply Deep Reinforcement Learning for the task of detecting anatomical structures in 3D CT Scans.
- Combination of:
  - Convolutional Neural Network as the Policy Function and,
  - Cognitive Model of Reinforcement Learning Framework.
- A system capable of both:
  - Identifying an anatomical structure
  - Able to find the structure in a full 3D scan.

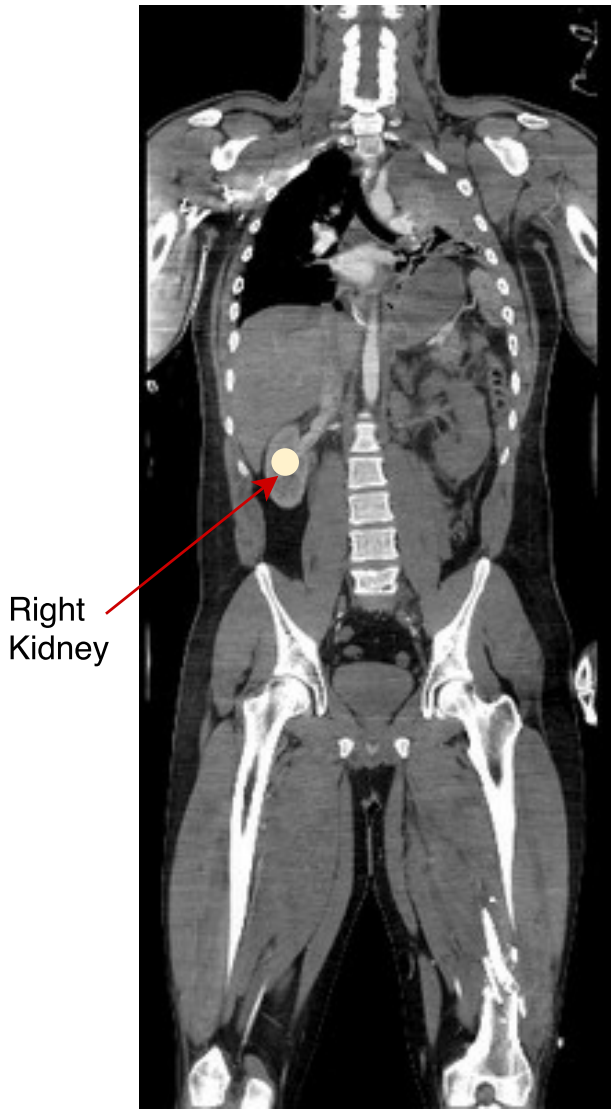
Outline

Problem Definition

Visualization

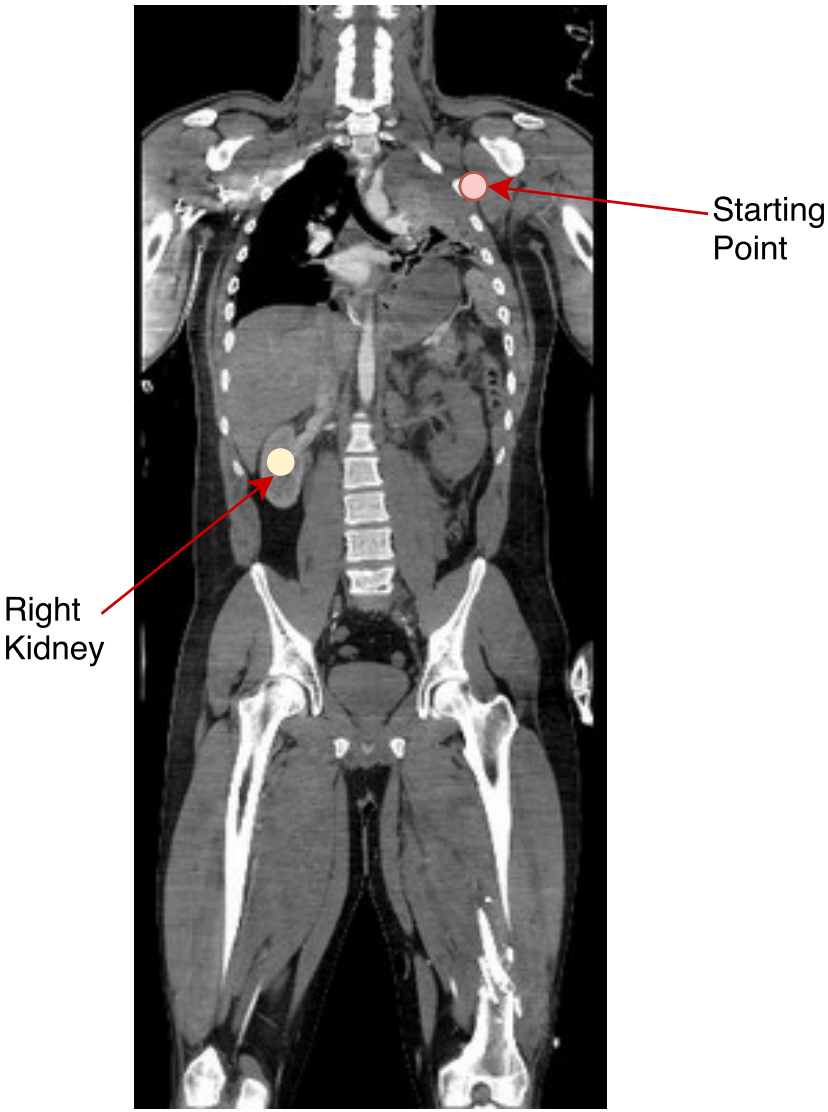
Problem Definition

Visualization



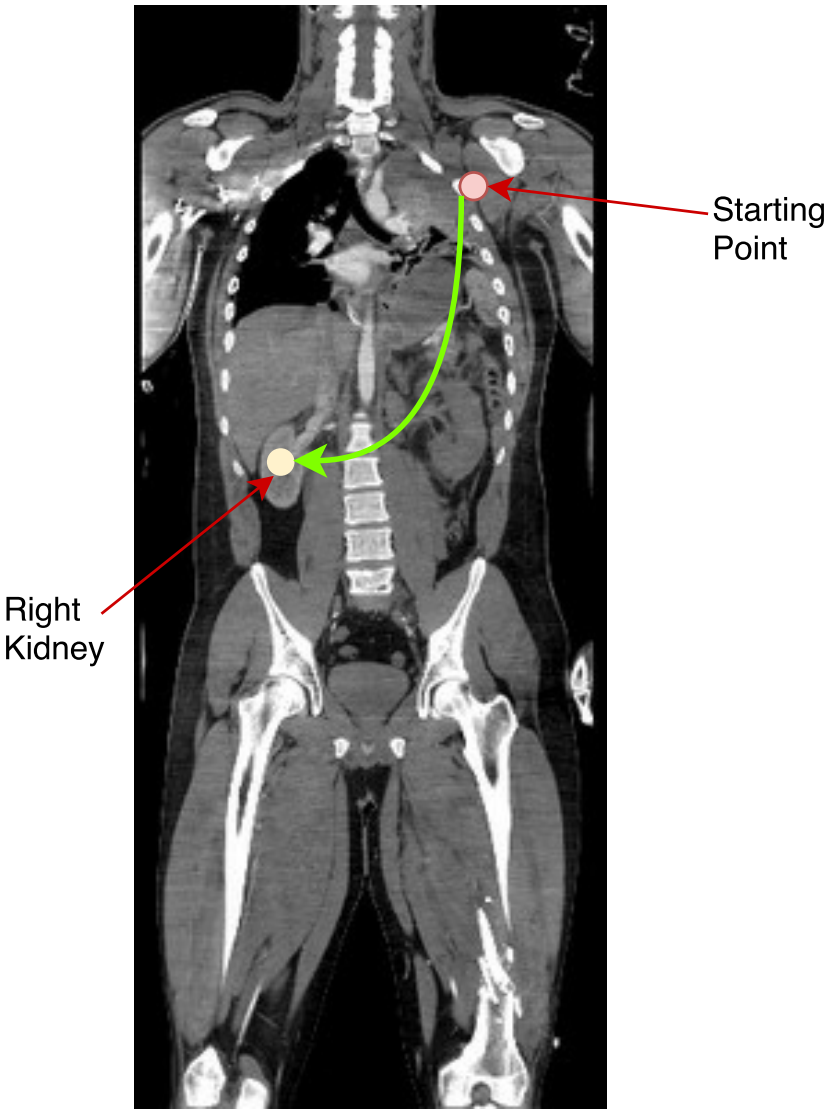
Problem Definition

Visualization



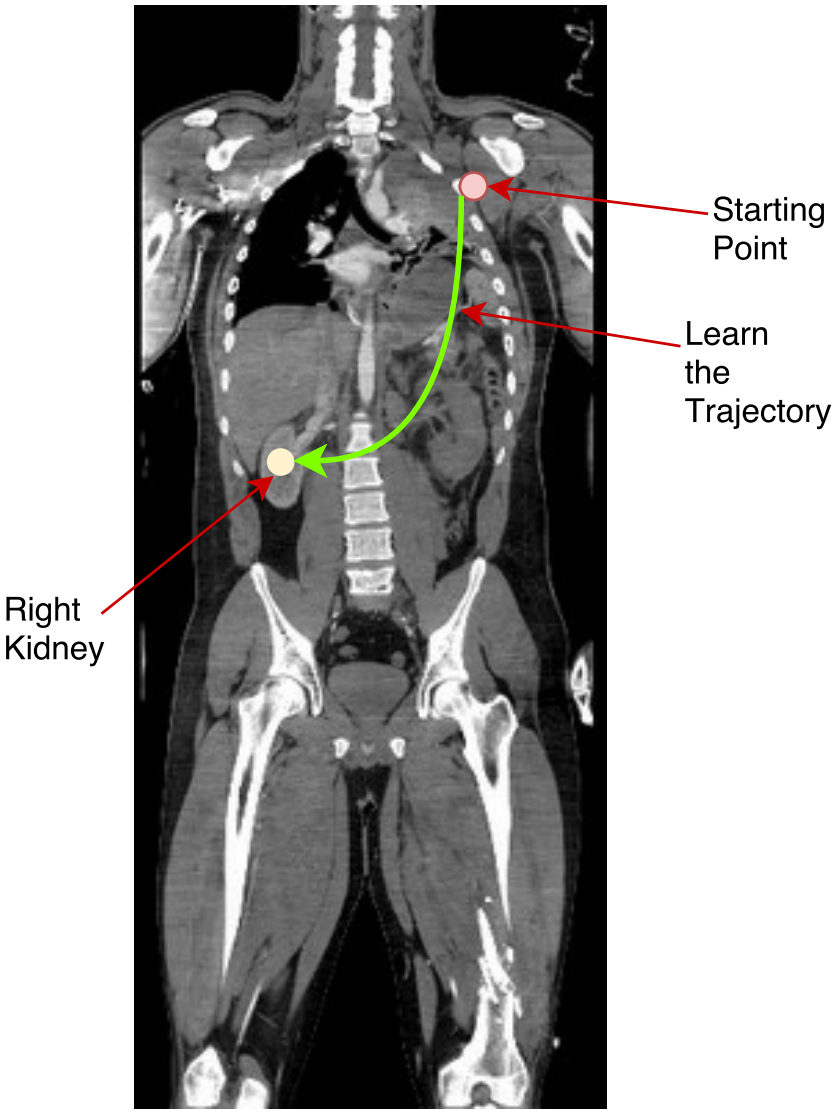
Problem Definition

Visualization



Problem Definition

Visualization



Outline

Modeling the Agent

Agent as a controller of Moving Cube

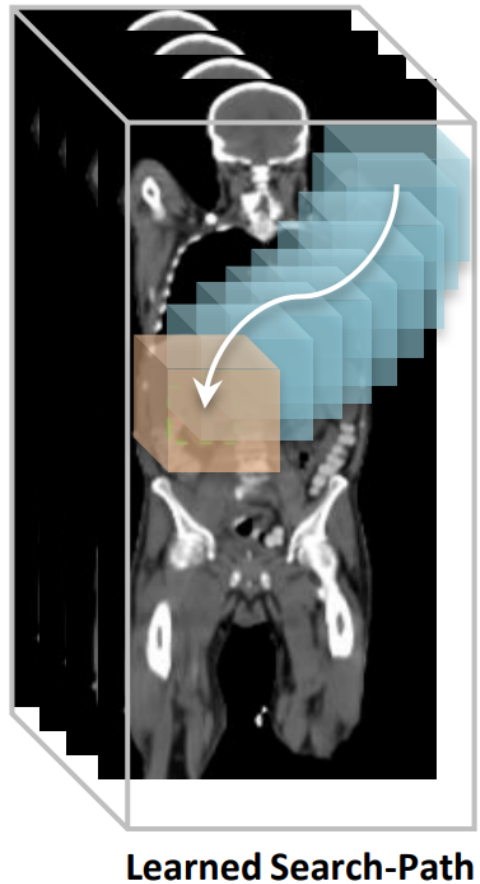


Figure: The search is performed by a moving cube with $\vec{p}_k$ as its centre.

Modeling the Agent

Agent as a controller of Moving Cube

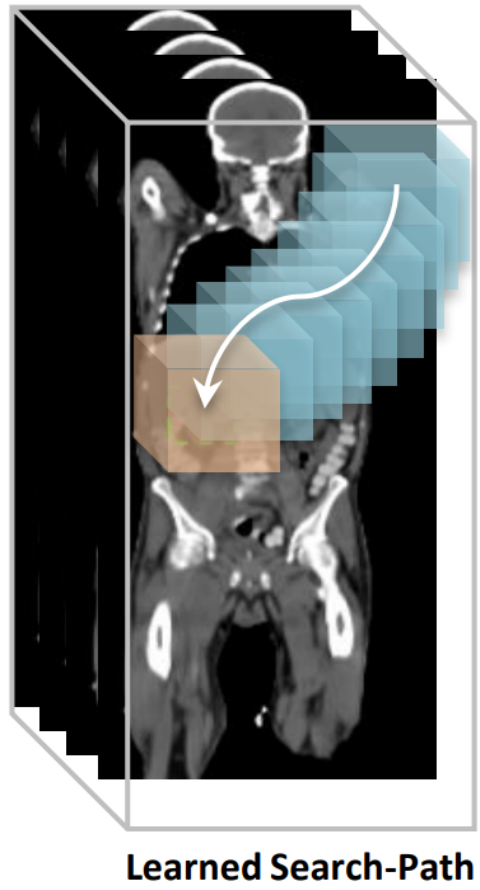


Figure: The search is performed by a moving cube with $\vec{p}_k$ as its centre.

Mario == Cube

Modeling the Agent

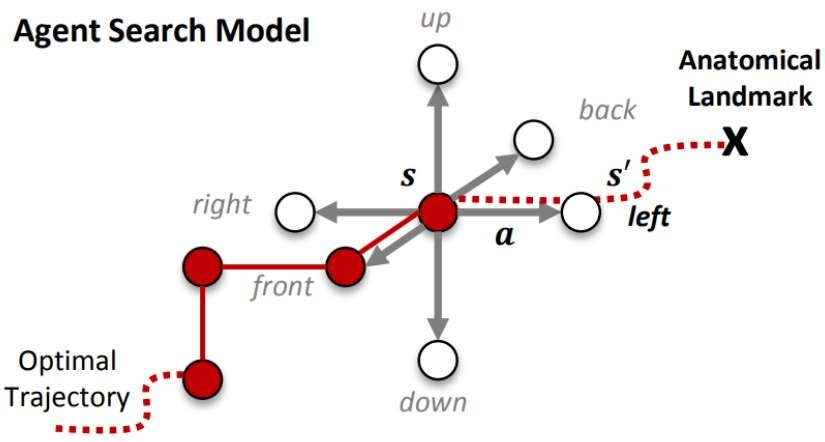
Discrete Action Space

- The agent is allowed to take a total of 6 actions.

Modeling the Agent

Discrete Action Space

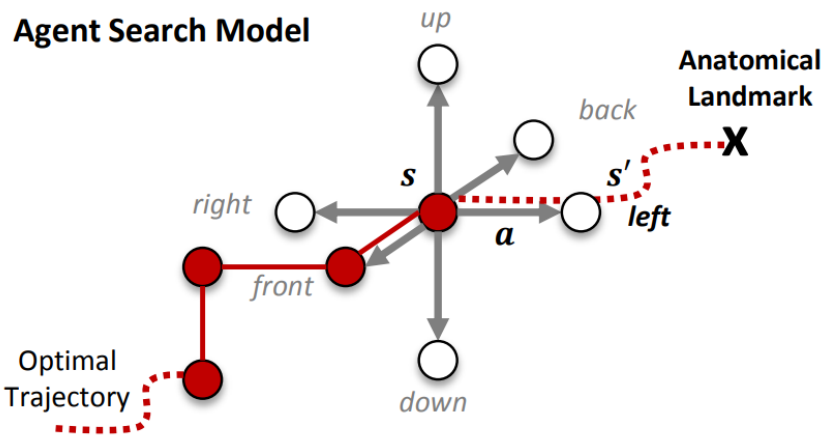
- The agent is allowed to take a total of 6 actions.



Modeling the Agent

Discrete Action Space

- The agent is allowed to take a total of 6 actions.
- It can move up, down, left, right, front or back of the 3D volume.



Reward Function

Defining Reward Function

- The reward function for this problem is simply:

Reward Function

Defining Reward Function

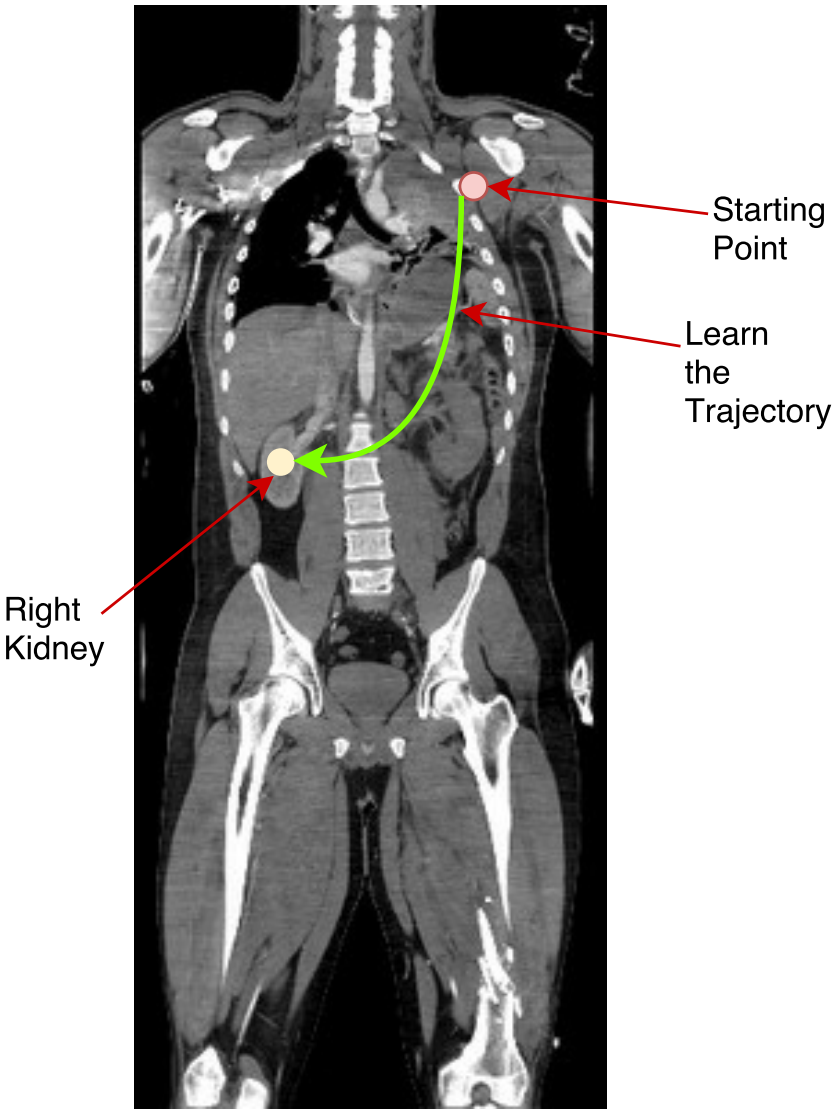
- The reward function for this problem is simply:

$$r_k = ||\vec{p}_k - \vec{p}_{GT}||$$

- Euclidean Distance between the current position, and the ground truth position.

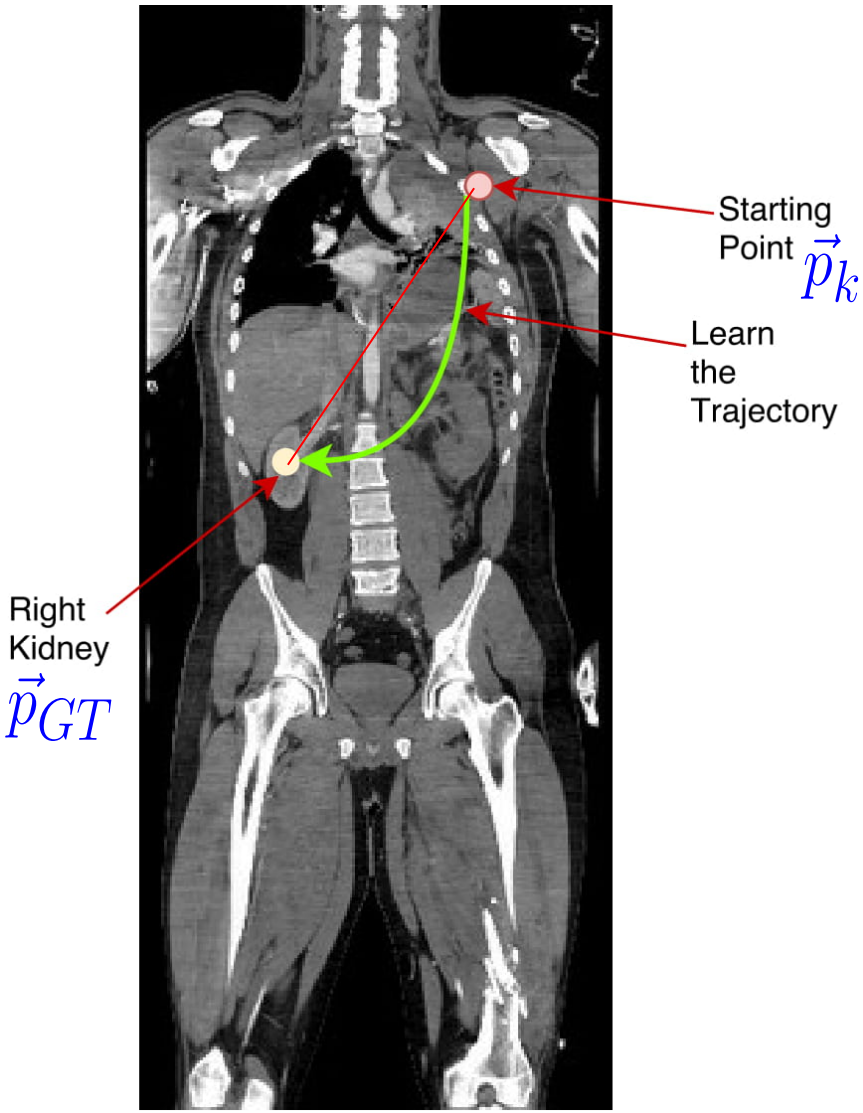
Reward Function

Defining Reward Function



Reward Function

Defining Reward Function



Outline

Drawing Parallels

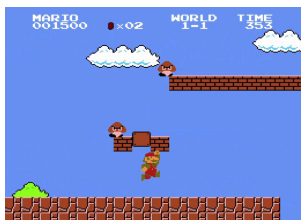
Game Playing RL vs Object Detection RL: Policy

Connect the earlier terminology to this proposed terminology.

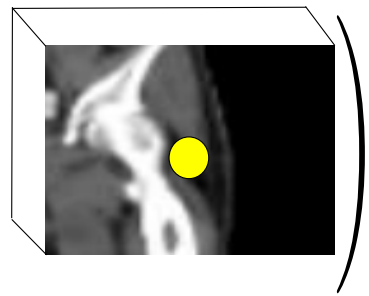
Drawing Parallels

Game Playing RL vs Object Detection RL: Policy

Connect the earlier terminology to this proposed terminology.

$$\pi\left(\text{  }\right) = \text{“Jump”}$$

Current State

$$\pi\left(\text{  }\right) = \text{“Up”}$$

Current State

In both cases, Policy Function is a Convolutional Neural Network (2D for Atari and 3D for CT).

Drawing Parallels

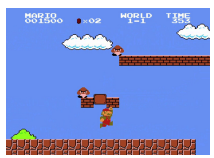
Game Playing RL vs Object Detection RL: Reward

Connect the earlier terminology to this proposed terminology.

Drawing Parallels

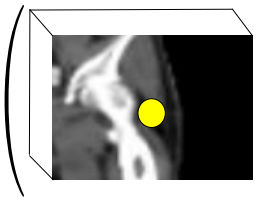
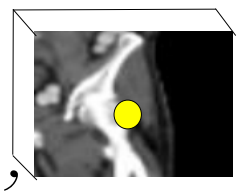
Game Playing RL vs Object Detection RL: Reward

Connect the earlier terminology to this proposed terminology.

$$\mathcal{R}\left(\text{  }, \text{“Jump”}\right) = 10$$

Current State

Reward function defined by Environment

$$\mathcal{R}\left(\text{  }, \text{  }, \text{“Left”}\right) = ||\vec{p}_{new} - \vec{p}_{GT}||$$

Current State | New State | Action

Reward function defined by User

The reward function in this work was defined as distance between the agent's position and ground truth position.

Drawing Parallels

Game Playing RL vs Object Detection RL: Trajectory

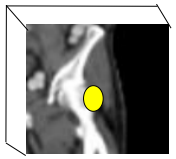
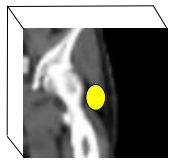
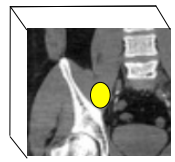
Connect the earlier terminology to this proposed terminology.

Drawing Parallels

Game Playing RL vs Object Detection RL: Trajectory

Connect the earlier terminology to this proposed terminology.

$$\tau = \Big(\text{  } \text{“Jump”, 10} \; , \; \text{  } \text{“Forward”, 10} \; , \; \text{  } \; , \; , \; , \; , \; \text{  } \text{“Jump”, 10} \Big)$$

$$\tau = \Big(\text{  } \text{“Right”, 10} \; , \; \text{  } \text{“Down”, 8} \; , \; \text{  } \text{“Back”, 5} \; , \; , \; , \; , \; \text{  } \text{“Up”, 2.5} \Big)$$

The trajectory vector stores all “states” or 3D patch volumes that it visited, and the corresponding actions that the agent took.

Optimization Function

Traditional vs Deep

- In traditional RL, we use

$$Q^*(s, a) = \mathbb{E}_{s'}\left(r + \gamma max_{a'} Q^*(s', a')\right)$$

- As objective function, also called the Bellman Equation.
- However, due to high dimensional nature of state space (state if defined in terms of position in the 3D image), this function is unsuitable.

Optimization Function

Traditional vs Deep

- In traditional RL, we use

$$Q^*(s, a) = \mathbb{E}_{s'}\left(r + \gamma max_{a'} Q^*(s', a')\right)$$

- As objective function, also called the Bellman Equation.
- However, due to high dimensional nature of state space (state if defined in terms of position in the 3D image), this function is unsuitable.
- Solution?

Optimization Function

Traditional vs Deep

- In traditional RL, we use

$$Q^*(s, a) = \mathbb{E}_{s'}\left(r + \gamma max_{a'} Q^*(s', a')\right)$$

- As objective function, also called the Bellman Equation.
- However, due to high dimensional nature of state space (state if defined in terms of position in the 3D image), this function is unsuitable.
- Solution?
- Use a Deep Convolutional Neural Network as a non-linear parameterization of Q^*.

Optimization Function

Traditional vs Deep

- In traditional RL, we use
$$Q^*(s, a) = \mathbb{E}_{s'}\left(r + \gamma max_{a'} Q^*(s', a')\right)$$
- As objective function, also called the Bellman Equation.
- However, due to high dimensional nature of state space (state if defined in terms of position in the 3D image), this function is unsuitable.
- Solution?
- Use a Deep Convolutional Neural Network as a non-linear parameterization of Q^*.
- This is called Deep Q-Network (DQN).

Optimization Function

Bellman Optimality with MSE

- A Deep Q-Network can be trained in a RL setup using iterative approach.

Optimization Function

Bellman Optimality with MSE

- A Deep Q-Network can be trained in a RL setup using iterative approach.
- Minimizes the mean squared error based on the Bellman optimality equation.

Optimization Function

Bellman Optimality with MSE

- A Deep Q-Network can be trained in a RL setup using iterative approach.
- Minimizes the mean squared error based on the Bellman optimality equation.

$$\hat{\theta}^{(i)} = argmin_{\theta^{(i)}} \mathbb{E}_{s,a,r,s'} \left[\left(y - Q(s, a; \theta^{(i)})^2 \right) \right]$$

Optimization Function

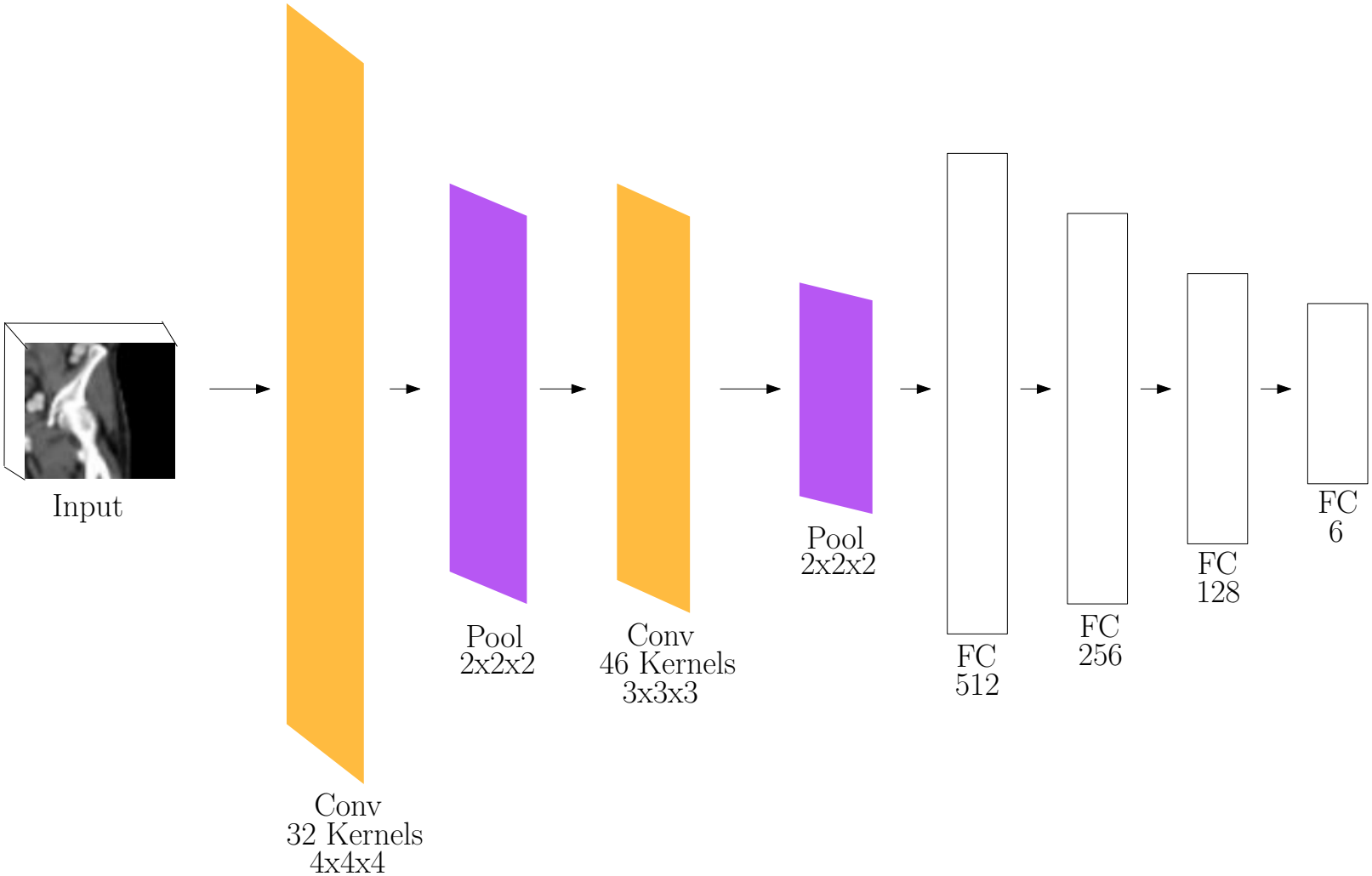
Bellman Optimality with MSE

- A Deep Q-Network can be trained in a RL setup using iterative approach.
- Minimizes the mean squared error based on the Bellman optimality equation.

$$\hat{\theta}^{(i)} = argmin_{\theta^{(i)}} \mathbb{E}_{s,a,r,s'} \left[\left(y - Q(s, a; \theta^{(i)})^2 \right) \right]$$

- where $y = r + \gamma max_{a'} Q(s', a'; \theta^{(i)})$

Network Architecture



Network Architecture

Outline

Outline

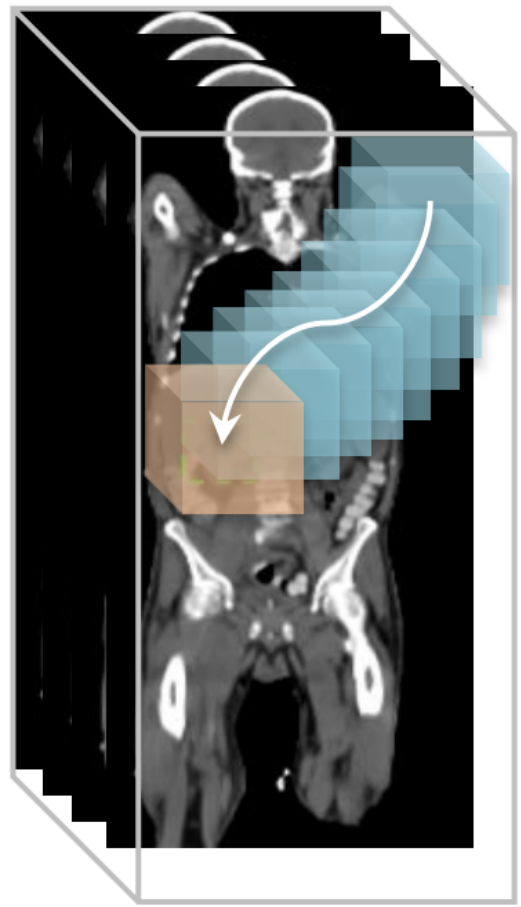
Major Limitation

Major Limitation

- An obvious limitation of the approach is the fixed box size of the agent, as shown:

Major Limitation

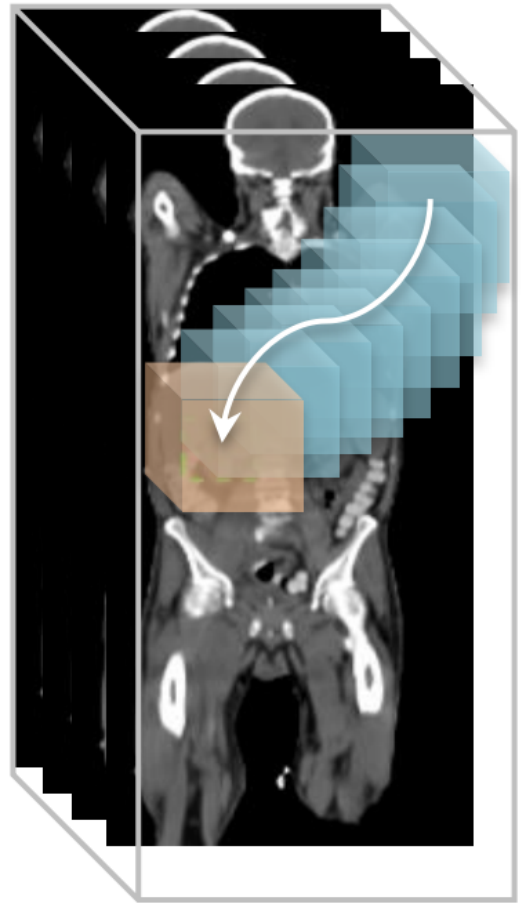
- An obvious limitation of the approach is the fixed box size of the agent, as shown:



Learned Search-Path

Major Limitation

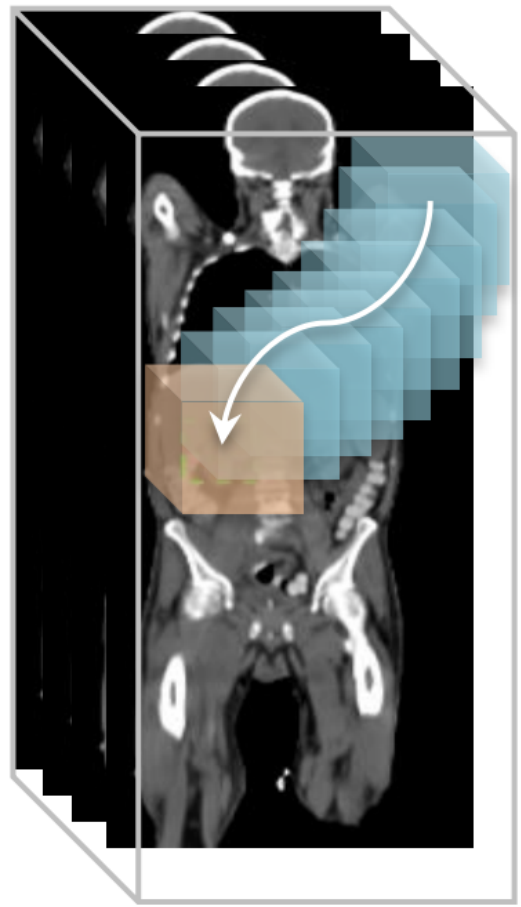
- An obvious limitation of the approach is the fixed box size of the agent, as shown:
- What if the size of object is much smaller than the box?



Learned Search-Path

Major Limitation

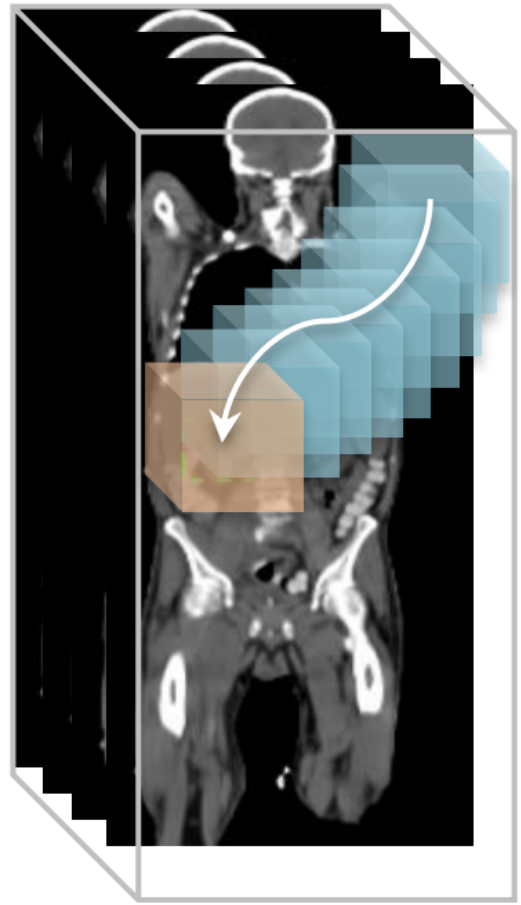
- An obvious limitation of the approach is the fixed box size of the agent, as shown:
- What if the size of object is much smaller than the box?
  - Localization is useless if the box is too large.



Learned Search-Path

Major Limitation

- An obvious limitation of the approach is the fixed box size of the agent, as shown:
- What if the size of object is much smaller than the box?
  - Localization is useless if the box is too large.
- Need to make the system scale-invariant.

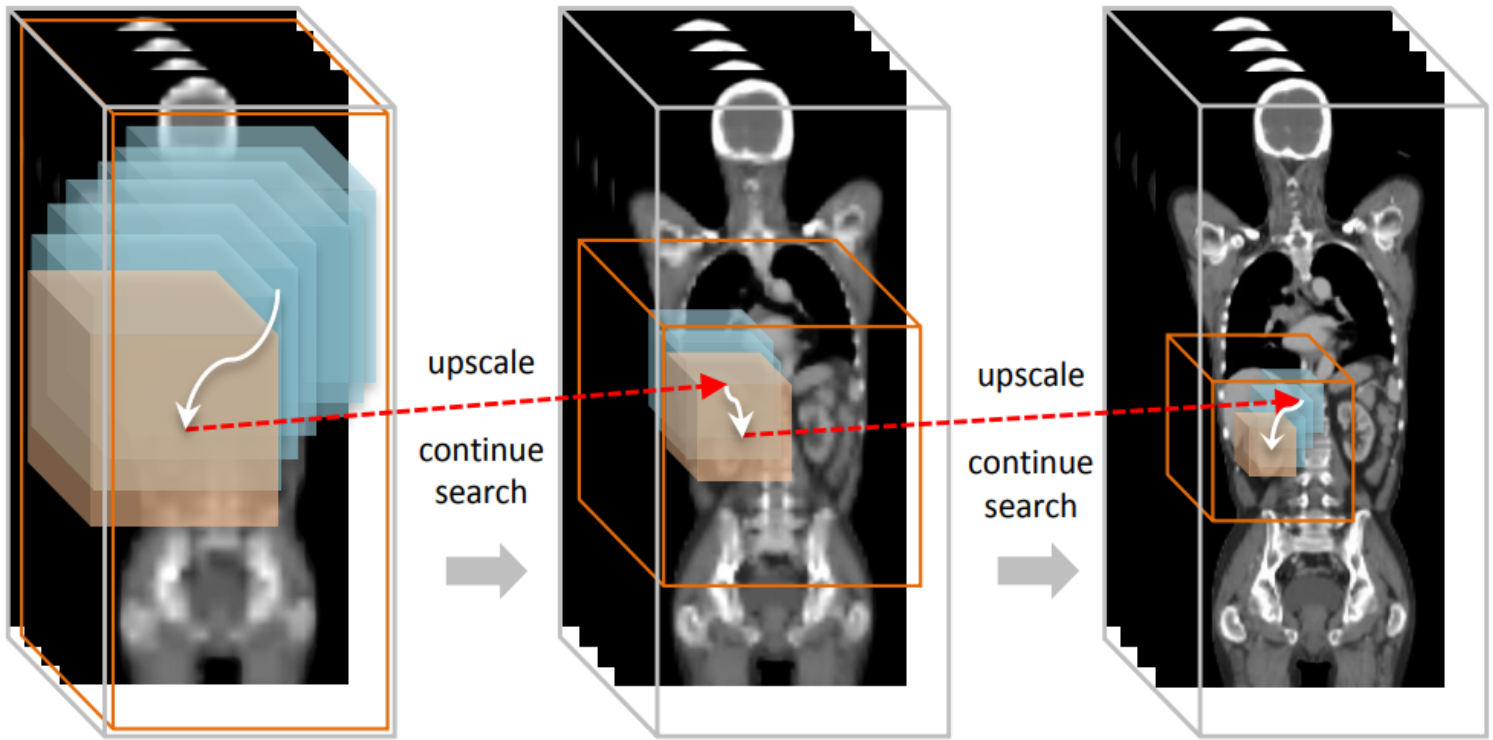


Learned Search-Path

Outline

Multi-Scale Search Strategies

Visualization



Multi-Scale Search Strategies

Solution to Scale-invariance

- Solution: Search on multiple scales of the input 3D volume.
- Obtain a finite number of scaled 3D volumes, starting from coarsest (16mm) to finest (4mm).
- Search algorithm:
  - Start the search from the coarsest scale (lowest resolution).
  - If the agent converges[1], change to finer (higher resolution) scale, and start the search again from the convergence point.
  - Repeat until convergence on finest scale is reached.

[1] discussed in the next slide

Multi-Scale Search Strategies

Convergence Criteria

- The convergence criteria for the agent answers the question: When does the agent know it has found the object of interest?
- Answer: Detect small oscillatory-like cycles in the agent's behaviour.
- If a cycle is detected, save the position, and move to next scale level to continue search.
  - If the scale is finest, return the current position as the position of object of interest.

Multi-Scale Search Strategies

Increased Complexity in Scale Space

- However
- Searching through multiple scale spaces increases the complexity in learning of policy function (CNN).
- The CNN would have to learn
  - how to distinguish an object of interest from background
  - how to navigate the image space by taking actions
  - how to do the above 2 tasks in multiple scales.
- This would make the learning intractable.
- Solution: Authors propose to use multiple CNNs for multiple scales.

Multi-Scale Search Strategies

Increased Complexity in Scale Space

Separate CNNs for separate scale 3D images.

Outline

Comparison

Algorithms compared against

Comparison

Algorithms compared against

- The proposed method was compared against:

Comparison

Algorithms compared against

- The proposed method was compared against:
  - Probabilistic Boosting Trees

Comparison

Algorithms compared against

- The proposed method was compared against:
  - Probabilistic Boosting Trees
  - Extremely Randomized Trees with Hough Regression

Comparison

Algorithms compared against

- The proposed method was compared against:
  - Probabilistic Boosting Trees
  - Extremely Randomized Trees with Hough Regression
  - Overfeat, adapted to 3D data

Comparison

Algorithms compared against

- The proposed method was compared against:
  - Probabilistic Boosting Trees
  - Extremely Randomized Trees with Hough Regression
  - Overfeat, adapted to 3D data
  - 3D Deep Learning with Filter Decomposition

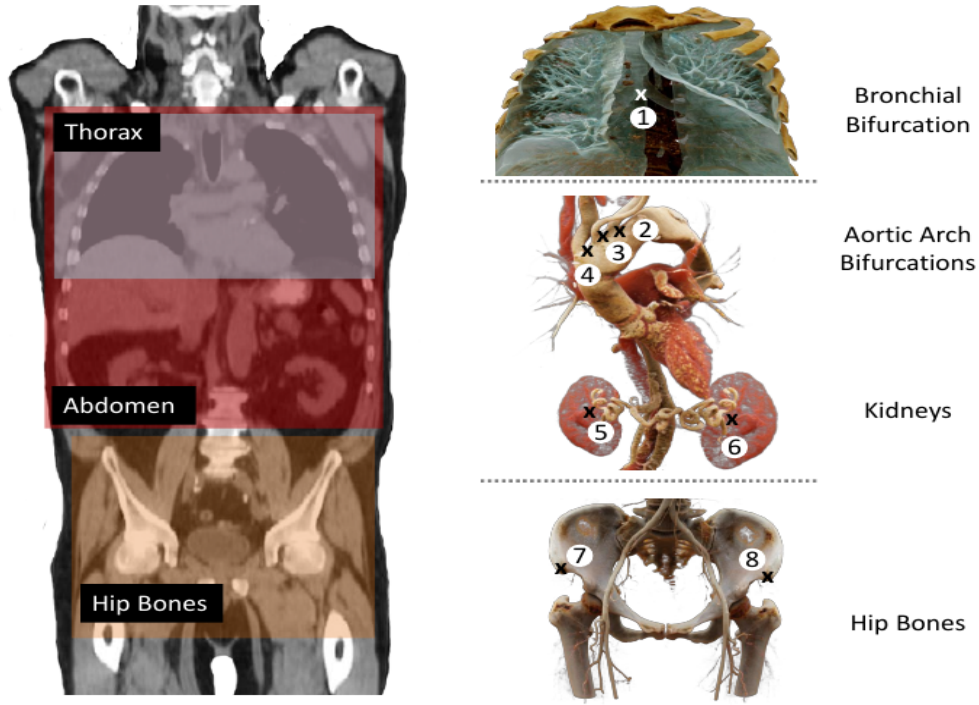
Comparison

Algorithms compared against

- The proposed method was compared against:
  - Probabilistic Boosting Trees
  - Extremely Randomized Trees with Hough Regression
  - Overfeat, adapted to 3D data
  - 3D Deep Learning with Filter Decomposition
  - Scanning with Cascaded Sparse-Adaptive DNN

Dataset

- 1487 3D-CT Volumes from 532 patients.
- Contains different field of views:
  - Cardiac CT Scans
  - CT scans of legs, pelvis, head, neck
  - Scans of thorax



Metrics

- **Failure Percentage Rate**
  - If the agent doesn't come within:
    - 30mm for the center of each kidney.
    - 10mm for the remaining landmarks
  - It is deemed a failure of detection.

Metrics

- **Failure Percentage Rate**
  - If the agent doesn't come within:
    - 30mm for the center of each kidney.
    - 10mm for the remaining landmarks
  - It is deemed a failure of detection.
- **Accuracy**
  - According to the number of times the agent detected the structure successfully (using above definition).

Metrics

- **Failure Percentage Rate**
  - If the agent doesn't come within:
    - 30mm for the center of each kidney.
    - 10mm for the remaining landmarks
  - It is deemed a failure of detection.
- **Accuracy**
  - According to the number of times the agent detected the structure successfully (using above definition).
- **Speed**
  - Raw speed of detection on their hardware platform

Quantitative Results

Performance Compared to Other Published Works

Solution	Dataset Size (Data/Patients)	Accuracy (mm)	Speed (seconds)
Zhan *et al.* [27]	18/18 CT	4.72	4
Fenchel *et al.* [35]	31/31 MR	22.4	20
Criminisi *et al.* [12]	100/– CT	17.60	1
Pauly *et al.* [32]	33/33 MR	14.95	0.8
Cuingnet *et al.* [11]	233/89 CT	10.5	2.8
Donner *et al.* [10]	20/20 CT	5.25	120
Criminisi *et al.* [31]	400/– CT	13.50	4
Chu *et al.* [30]	10/10 CT	1.90 [1]	30
Potesil *et al.* [37]	83/83 CT	4.70	N/A
de Vos *et al.* [24]	100/– CT	4.80	10
Ours	**1487/532** CT	**4.19** [2]	**0.061**

[1] Evaluated only on vertebrae localization with strong priors.
[2] With no failures of clinical significance. All other solutions did not provide any information in this respect.

Quantitative Results

Speed Compared to Methods Discussed in the Previous Slide

DETECTION TIME per Volume [**seconds**]

Method	Platform	Median	Min	Max
PBT [3]	CPU 8-core	1.051	0.14	16.20
ExtRTrees [10]	CPU 8-core	4.714	0.272	44.535
Overfeat [1] [22]	GPU Titan X	2.175	0.331	14.86
3D-DL [18]	CPU 10-core	57.034	7.161	548.23
SADNN [2] [1]	GPU GTX 1080	0.471	0.064	3.029
Ours	CPU 8-core	0.061	0.035	0.155
Ours	GPU Titan X	**0.033**	**0.018**	**0.085**

[1,2] Evaluated on finest isotropic resolution of 4 mm.

Quantitative Results

Individual Anatomy Detection Comparison with Other Methods

			Failed Cases			Accuracy (excl. failed cases)	
Structure Type	**Landmark**	**Method**	**% Failed**	**Median**	**Max**	**Mean ± STD**	**Median**
Bone	Hip Bone Right Front Corner	PBT [3]	4.35%	38.89	141.88	3.05 ± 1.60	2.77
		ExtRTrees [10]	8.07%	20.27	460.22	5.01 ± 2.40	4.95
		Overfeat [22]	9.31%	35.64	231.29	4.55 ± 1.98	4.31
		3D-DL [18]	0.62%	10.17	10.17	2.84 ± 1.36	2.53
		SADNN [1]	0%	–	–	3.50 ± 1.63	3.37
		Ours	**0%**	–	–	$\mathbf{2.80} \pm 1.46$	**2.53**
	Hip Bone Left Front Corner	PBT [3]	3.75%	14.30	28.57	4.03 ± 1.79	3.86
		ExtRTrees [10]	6.25%	13.43	17.44	5.13 ± 2.78	5.08
		Overfeat [22]	3.75%	127.52	242.05	4.19 ± 1.80	4.15
		3D-DL [18]	2.50%	262.81	513.81	3.14 ± 1.49	2.98
		SADNN [1]	1.25%	12.57	12.57	4.44 ± 1.75	4.43
		Ours	**0%**	–	–	$\mathbf{3.07} \pm 2.14$	**2.85**

Critique

Some Critical Observations

- Authors reimplemented a method without a main subsystem.
- Runtime comparison was performed on heterogeneous systems.
- Self imposed accuracy threshold.
  - If the agent is within 10mm of the body part, it is deemed as successful detection.
  - No analysis of reducing/increasing this threshold to report varying results.
  - Although the authors cite clinical reasons to choose this threshold.
- Starting point of the agent is heuristically chosen to be close to the actual expected location.
  - Makes sense because searching for right kidney does not have to start from the head.
  - However, no results reported about runtime speed and accuracy if the chosen point is randomized.

Critique

Some Critical Observations

Pros

- A brilliantly written paper.
- Highly polished analysis, clear and concise handling of various apparent limitations.

Conclusion

- Authors proposed a new formulation of the object detection problem in 3D CT Scans.
- Posed the problem as Deep Reinforcement Learning based agent learning to navigate the image space and identifying markers.
- Used Deep Q-Learning with 3D Convolutional Neural Network as policy function.
- Reported 0% Failure Percentage Rate for their approach.
- Significant orders of speed-up in detection process, compared to state-of-art methods.

Thank You for listening!

Outline

Starting Point of the Agent

How to set agent's starting point?

- Also, the starting point of the agent is set to a position $\vec{p_0}$ using the expected position of the object of interest.
- This position is sampled from the training data.
- For example, for right kidney, it is known that it is on the right side of the abdomen. Hence set the starting position close to that region.

Object Not in Scan?

How to find out if object is not in scan?

Object Not in Scan?

How to find out if object is not in scan?

- In any object detection framework, the system or agent should be able to handle the case where the object is not present in the image (or scene).

Object Not in Scan?

How to find out if object is not in scan?

- In any object detection framework, the system or agent should be able to handle the case where the object is not present in the image (or scene).
- In this work, a very simple heuristic approach is taken to find out if the object is not present in the scene.

Object Not in Scan?

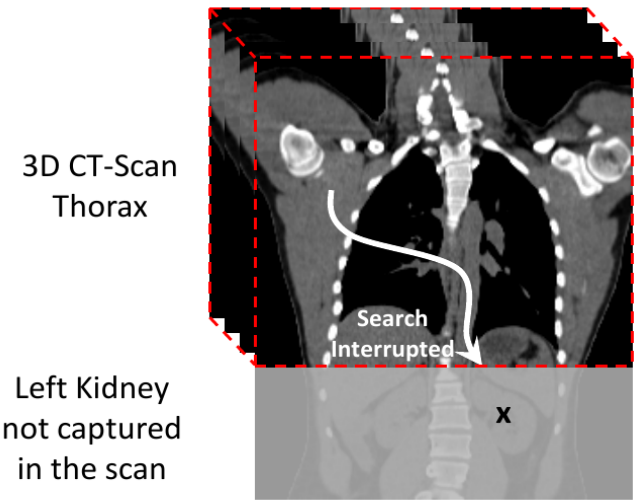
How to find out if object is not in scan?

- In any object detection framework, the system or agent should be able to handle the case where the object is not present in the image (or scene).
- In this work, a very simple heuristic approach is taken to find out if the object is not present in the scene.
- Approach: Detect if the agent leaves the image space while scanning.

Object Not in Scan?

How to find out if object is not in scan?

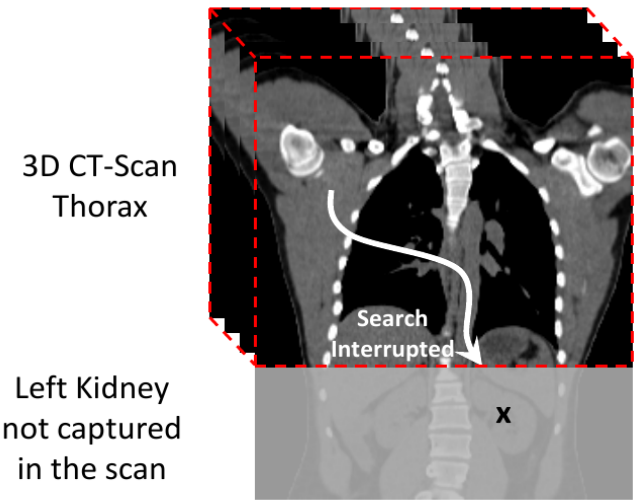
- In any object detection framework, the system or agent should be able to handle the case where the object is not present in the image (or scene).
- In this work, a very simple heuristic approach is taken to find out if the object is not present in the scene.
- Approach: Detect if the agent leaves the image space while scanning.



Object Not in Scan?

How to find out if object is not in scan?

- In any object detection framework, the system or agent should be able to handle the case where the object is not present in the image (or scene).
- In this work, a very simple heuristic approach is taken to find out if the object is not present in the scene.
- Approach: Detect if the agent leaves the image space while scanning.
- Although simple, the approach is shown to work atleast for 90%+ negative cases, which show the robustness.



Exploration vs Exploitation

Overview

- Learning the action-value function Q^* enables the agent to effectively search for objects in the image.

Exploration vs Exploitation

Overview

- Learning the action-value function Q^* enables the agent to effectively search for objects in the image.
- However there is a need to balance exploration by the agent against exploitation.

Exploration vs Exploitation

Overview

- Learning the action-value function Q^* enables the agent to effectively search for objects in the image.
- However there is a need to balance exploration by the agent against exploitation.
- The authors propose to use three techniques to ensure adequate exploration of the environment.

ϵ-greedy Approach

ϵ-greedy Approach

- A variable $\epsilon \in [0, 1]$ controls the randomness in the exploration.

ϵ-greedy Approach

- A variable $\epsilon \in [0, 1]$ controls the randomness in the exploration.
- In simple words:

ϵ-greedy Approach

- A variable $\epsilon \in [0, 1]$ controls the randomness in the exploration.
- In simple words:
- During training, sample the actions from

ϵ-greedy Approach

- A variable $\epsilon \in [0, 1]$ controls the randomness in the exploration.
- In simple words:
- During training, sample the actions from
  - Uniform Distribution, with probability ϵ, or

ϵ-greedy Approach

- A variable $\epsilon \in [0, 1]$ controls the randomness in the exploration.
- In simple words:
- During training, sample the actions from
  - Uniform Distribution, with probability ϵ, or
  - Deterministically from the current policy , with probability $1 - \epsilon$.

ϵ-greedy Approach

- A variable $\epsilon \in [0, 1]$ controls the randomness in the exploration.
- In simple words:
- During training, sample the actions from
  - Uniform Distribution, with probability ϵ, or
  - Deterministically from the current policy , with probability $1 - \epsilon$.
- This value is annealed linearly from 1.0 to 0.05 during the training.

Experience Replay

Experience Replay

- Ensure training stability by decorrelating the training samples using experience replay.

Experience Replay

- Ensure training stability by decorrelating the training samples using experience replay.
- During training, memorize the trajectories in $M = [T_1, T_2, T_3 ...]$.

Experience Replay

- Ensure training stability by decorrelating the training samples using experience replay.
- During training, memorize the trajectories in $M = [T_1, T_2, T_3 ...]$.
- Uniformly sample from this list of trajectories.

Experience Replay

- Ensure training stability by decorrelating the training samples using experience replay.
- During training, memorize the trajectories in $M = [T_1, T_2, T_3 ...]$.
- Uniformly sample from this list of trajectories.
- Use those to train the policy (deep convolutional neural network).

Adaptive Episode Length

Adaptive Episode Length

- Further accelerate the training using an adaptive episode length.

Adaptive Episode Length

- Further accelerate the training using an adaptive episode length.
- During training, linearly increase the number of trajectories sampled from the memory, and train using those.

Adaptive Episode Length

- Further accelerate the training using an adaptive episode length.
- During training, linearly increase the number of trajectories sampled from the memory, and train using those.
- That is, don't gather new experience all through the training.

Adaptive Episode Length

- Further accelerate the training using an adaptive episode length.
- During training, linearly increase the number of trajectories sampled from the memory, and train using those.
- That is, don't gather new experience all through the training.
- As the training progresses, use more experiences of the past rather than making new ones.

Adaptive Episode Length

- Further accelerate the training using an adaptive episode length.
- During training, linearly increase the number of trajectories sampled from the memory, and train using those.
- That is, don't gather new experience all through the training.
- As the training progresses, use more experiences of the past rather than making new ones.
- This reduces training time by 30%.