DQN for Detecting Brain Hemorrhage

Authors: Eric Yang, Suraj Godithi

Introduction:

Intracranial hemorrhage is a severe condition that requires immediate diagnosis and localization to reduce the likelihood of death or long-term effects like neurological deterioration. CT imaging is the standard tool for detecting hemorrhages in the brain, but accurate interpretation of CT scans requires expert knowledge. Our project explores how reinforcement learning can be applied to automate the localization of brain hemorrhages in 2D CT scans.

Convolutional neural networks are inherently difficult to interpret, and U-Nets—while a specialized formulation of CNNs for segmentation—still require large amounts of labeled data to perform well, much like standard convolutional models. Reinforcement learning allows an agent to actively interact with the CT slice by iteratively moving and resizing a bounding box to localize in on the hemorrhage. This model can yield more interpretable results, as the agent's trajectory toward the hemorrhage is observable and influenced by a tunable reward system.

Building on prior work such as Caicedo and Lazebnik's research on active object localization with deep reinforcement learning, we focused on training a Deep Q-Network (DQN) agent to localize hemorrhages through iteratively moving and adjusting a bounding box. By training the agent to move, zoom, and reshape the bounding box over CT slices, and guiding its learning through tunable reward functions, we aimed to enable the model to learn not just what to detect, but also how to search for it effectively.

Methodology:

Baseline U-Net:

To begin our project, we first attempted to reproduce a U-Net-based segmentation model as a benchmark for brain hemorrhage localization as provided for by Kaggle. Hoping that this could be a comparatively less coding-intensive way of starting the project and a good way to introduce ourselves to the format and practical handling of segmentation datasets, we wound up running into unexpected issues as mismatches between modules and Python led to unexpected behavior—rather than detecting hemorrhages, the model often reconstructed general brain structure. In response, we downgraded Python and TensorFlow versions to match the original notebook, and re-implemented portions of the training pipeline to fine-tune the model on our specific dataset. Early on, we also made the decision to focus solely on CT slices that already contained hemorrhages, inspired by the work of Stember and Shalu, which was the only paper we encountered applying deep reinforcement learning to medical imaging. This allowed us to simplify the task and focus the DQN and U-Net models on learning localization, rather than full classification. As a result, we slightly modified the original Kaggle source code to restrict

training and evaluation to positive cases only. Early layers were frozen to retain low-level features, while the rest of the model was retrained to help it adapt to our dataset, which differed from the original used during pre-training. We evaluated performance using standard segmentation metrics: Dice loss, which emphasizes similarity between predicted and ground truth masks, and the Jaccard coefficient, which measures the ratio of shared area to total combined area.

Deep Q-Network:

Core Network:

For the design of our Deep Q-Network, we drew inspiration from several foundational works in reinforcement learning for vision tasks, including Caicedo and Lazebnik's *Active Object Localization with Deep Reinforcement Learning*, Sameeni and Li's *Object Detection with Deep Reinforcement Learning*, and Mnih et al.'s formative work *Playing Atari with Deep Reinforcement Learning*. These papers informed our choices in network architecture, action space design, reward formulation, and exploration strategies.

Our agent interacts with 2D CT slices by manipulating a bounding box through a discrete set of nine actions: move up, down, left, right, zoom in, zoom out, squash horizontally, squash vertically, and trigger. The bounding box is represented by four values: x_1, y_1, x_2, y_2 defining the top-left and bottom-right corners of the rectangular region being predicted. Each action (except trigger, which simply ends the episode) directly modifies these coordinates by a proportion α of the current width or height. Here, α is a tunable hyperparameter between 0 and 1 which controls the relative magnitude of each transformation. Transformations are applied by adding or subtracting α times the box width ($w = x_2 - x_1$) or height ($h = y_2 - y_1$) to the respective coordinates, depending on the intended effect. After each transformation, the coordinates are clipped to stay within image boundaries and sorted to ensure a valid box. Following Caicedo and Lazebnik, we primarily used $\alpha = 0.2$, which was noted to provide a good trade-off between localization speed and accuracy though we also experimented with $\alpha = 0.1$, hypothesizing that it might allow for finer adjustments which may be important in detecting smaller hemorrhages.

State representations also follow a similar structure to that proposed by Caicedo and Lazebnik, where each state is defined as a tuple (o, h). The visual component o is implemented as a 4096-dimensional feature vector extracted from the currently observed region. Each region is resized to 224 × 224 pixels and passed through a pretrained VGG16 network, from which we extract the output of the first fully connected layer (fc1) to serve as the agent's visual input. The history component h is implemented as a sequence of the last h actions (defaultly 10), each encoded as a one-hot 9-dimensional binary vector, and concatenated with o to form the full state. Unlike o, which was fixed, h was treated as a tunable component of the state representation, and our experiments included configurations both with and without it—motivated in part by findings from Samiei and Li, who observed that such binary histories hindered model performance.

Our agent's decision-making is driven by a Deep Q-Network (DQN), a neural network that approximates the action-value function Q(s, a), estimating the expected cumulative reward for taking action *a* in state *s*. The model follows a fully connected architecture with two hidden layers, each containing 1024 units followed by ReLU activation functions, followed by an output layer with 9 units—one for each possible action. The state representation (*o*, *h*) is provided as input to the network.

Training and Hyperparameters:

To train the DQN, we use an experience replay mechanism, following the approach introduced by Mnih et al. and further utilized by Samiei and Li. Transitions collected during agent-environment interactions are stored in a replay buffer and sampled uniformly to form mini-batches used for training. This technique helps decorrelate updates and improve sample efficiency. For each transition in the batch, the target Q-value is computed using the Bellman equation: the immediate reward plus the discounted maximum Q-value of the next state. The network is then updated by minimizing the mean squared error (MSE) between the predicted and target Q-values using the Adam optimizer with a learning rate of $1e^{-4}$.

We treated batch size as a tunable hyperparameter, initially testing values such as 32 and 64, and ultimately using 300 in our final setup to achieve more stable learning. Similarly, exploration was governed by an ε-greedy strategy, where the agent selects a random action with probability ε and the action with the highest predicted Q-value otherwise. We experimented with both linear and exponential schedules for reducing ε over the course of training. In the linear case—consistent with the approach used by Mnih et al.—ε was annealed over 10000 steps to steadily encourage exploration early and exploitation later. In our exponential variant, ε was decayed multiplicatively by a factor δ at the end of each epoch. δ was treated as a tunable hyperparameter, typically initialized at 0.95. In both settings, ε was reduced to a minimum value of 0.05 to preserve occasional exploration during later training.

We also experimented with different state configurations. While the visual component o remained fixed, the history component h was treated as a tunable element. Our experiments included configurations both with and without this component—motivated in part by findings from Samiei and Li, who observed that such binary histories hinder model performance when concatenated with high-dimensional CNN features.

Reward Functions:

Reward design is one of the most critical components of training a reinforcement learning agent, particularly in Deep Q-Networks, where the agent's behavior is directly shaped by the feedback it receives from the environment. In our setting, effective reward shaping was essential for encouraging precise localization and discouraging trivial or inefficient strategies. As such, we experimented with a variety of reward functions throughout development.

A central metric used throughout our reward design process was Intersection over Union (IoU), a standard evaluation measure for object localization. IoU is computed as the ratio of the

overlapping area between the predicted bounding box and the ground truth box to the area of their union. IoU values range from 0 (no overlap) to 1 (perfect match), and they provide a continuous, interpretable measure of localization accuracy.

Our initial reward structure was based on the design proposed by Caicedo and Lazebnik, where the agent receives a binary reward at each step depending on whether the predicted bounding box improves in Intersection over Union (IoU) with the ground truth. Specifically, a reward of +1 is granted if the IoU increases from one state to the next, and −1 otherwise. The "trigger" action, which terminates the episode, uses a different reward scheme: a fixed positive reward is granted if the final IoU exceeds a predefined threshold τ, and a negative penalty is applied otherwise. Following Caicedo and Lazebnik, we initialized $\tau = 0.6$, but we also experimented with lower thresholds (such as $\tau = 0.3$ and 0.25), motivated by the small size of many hemorrhagic lesions, which made it difficult for the agent to consistently achieve high IoU scores.

In addition to the binary IoU-based reward described by Caicedo and Lazebnik, we implemented and tested several alternative reward functions to better guide the agent's learning. One variant was a continuous IoU difference reward, where the agent received a reward equal to the change in IoU between the current and previous box. This formulation was intended to reward the agent proportionally to how much its prediction improved, offering more nuanced feedback than the binary +1/−1 scheme.

Later on, we began running into issues of the agent using the same move over and over again. Initially, this seemed to create overly large bounding boxes, so we introduced an area-penalized reward that combined IoU with a penalty term proportional to the normalized area of the predicted box.

As the model was refined, guard against the model repeatedly getting too small, we implemented multiple approaches. In a rudimentary approach, we used Manhattan distance, rewarding the agent based on the reduction in total L1 distance between the predicted and ground truth box corners. If the agent moved too far "inwards", meaning that it moved the top-left corner too far to the bottom right beyond the ground truth, or moved the bottom-right corner too far up beyond the ground truth mask, the agent was doubly penalized in hopes of harshly discouraging zooming too far in or becoming a line.

Lastly, we designed a coordinate-based reward function that provided discrete feedback for each coordinate (x_1, y_1, x_2, y_2), rewarding movement toward the ground truth and penalizing overextension or misalignment. Regardless of the shaping function used during box transformation steps, all reward variants included a separate scheme for the "trigger" action, which granted a fixed positive reward if the final IoU exceeded a threshold τ and a penalty otherwise. This structure ensured consistent termination behavior across reward types while allowing flexibility during the agent's exploration phase.

Evaluation:

Evaluation was initially framed as a binary classification metric indicating whether the predicted region (by the RL agent or U-Net) exhibited any nonzero overlap with the ground truth segmentation mask. This was motivated by early design considerations in which the RL agent operated as a small convolutional kernel traversing the image, potentially producing artificially inflated Intersection-over-Union (IoU) scores due to its limited spatial footprint. To mitigate this bias, the binary overlap criterion was considered a more robust evaluation proxy. However, as the RL agent's performance failed to reach parity with the U-Net segmentation baseline, a unified evaluation framework was deprioritized. Instead, RL performance was tracked independently via episodic reward curves, training loss trajectories, and qualitative assessments over randomly selected test samples.

Results:

U-Net:

The U-Net model demonstrated strong performance in segmenting hemorrhagic regions from CT scans, despite being trained on a limited dataset of only 28 images. As shown in Figure 1, the predicted masks closely aligned with the ground truth, capturing both the shape and location of the lesions in most cases. Using the binary evaluation metric described earlier—scoring each image as correct if there was any overlap between the predicted and ground truth masks—the U-Net achieved a score of 210 out of 318 images, indicating reliable generalization even under data-constrained conditions.

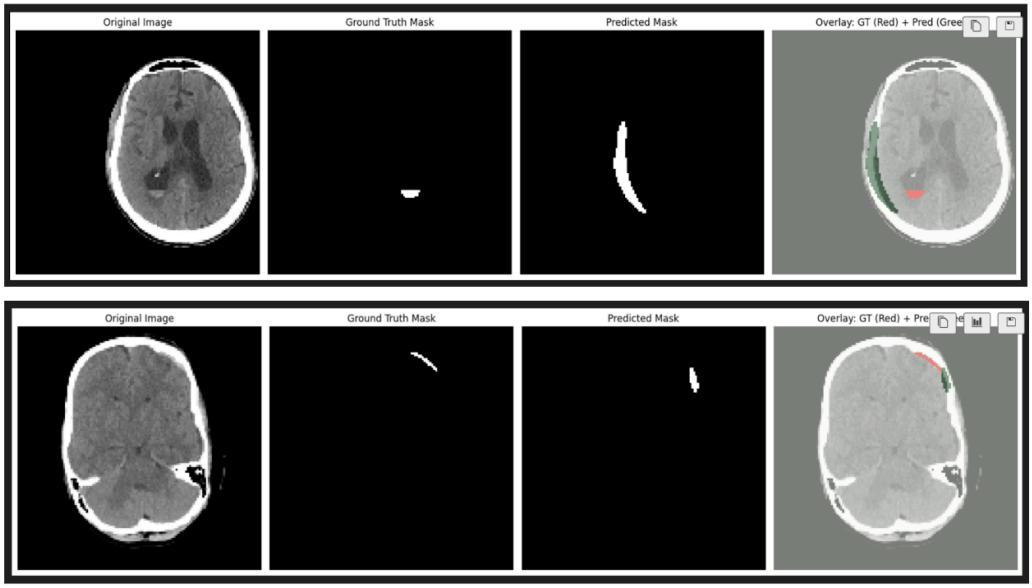

Figure 1: U-Net using dice loss predicting hemorrhage mask on CT scan, (from left to right), original image, the ground truth mask alone, followed by predicted mask from U-Net segmentation, and finally an overlay of everything on the original image.

Deep Q Network:

Binary Reward and IoU Difference:

We began by implementing a binary reward scheme for the reinforcement learning agent, drawing inspiration from the approaches used in Caicedo and Lazebnik. In this setup, the agent received a reward of +1 if an action increased the Intersection-over-Union (IoU) with the ground truth mask, and −1 otherwise. Consistent with these prior works, we adopted a history length of 10 past actions, a reward threshold of $\tau = 0.6$ for triggering the terminal action, and a discount factor $\gamma = 0.99$, but an exponential epsilon decay schedule with $\delta = 0.95$—mainly because this combination was simple to implement and was our first intuition for decaying epsilon over time.

Despite these design choices, the agent's performance was suboptimal. It often predicted extremely small bounding boxes that were far from the ground truth (see Figure 2). Loss plots showed a sharp decline early in training (also in Figure 2), which initially led us to suspect issues with the reward formulation as we reasoned the binary reward scheme offered no gradient of feedback as it treated marginally incorrect actions the same as drastically incorrect ones.

To address the limitations of the binary reward scheme, we experimented with an alternative reward function based on the difference in Intersection-over-Union (IoU) between consecutive steps. The idea was to provide a more continuous and informative signal—rewarding the agent proportionally to how much it improved its IoU with the ground truth, rather than simply whether it improved at all. This approach preserved the same hyperparameters used in the binary reward setup, including the history size, epsilon decay, and trigger threshold.

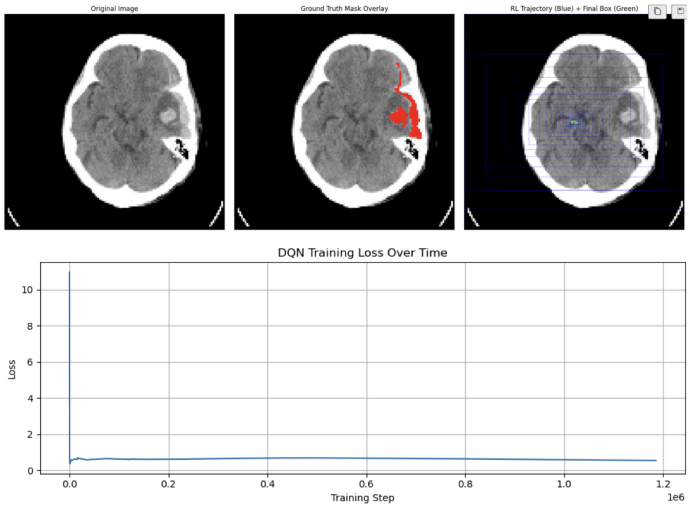

Figure 2: *Top:* Example output from the binary reward model trained over 15 epochs using the full test dataset. The CT scan is shown alongside overlays of the ground truth mask (red), initial bounding boxes (blue), and final predicted box (green). Key parameters include a history size $h = 10$, exponential epsilon decay with a decay rate $\delta = 0.95$, discount factor $\gamma = 0.99$, step size $\alpha = 0.2$, and a trigger threshold of $\tau = 0.6$. *Bottom:* Mean squared error (MSE) loss plotted over training steps.

However, this modification led to unexpected behavior. The agent consistently predicted very large bounding boxes as shown in Figure 3, often encompassing most of the image. Upon inspecting training logs which print the given reward at each step, we observed that some images with large hemorrhages allowed the agent to achieve high IoU with minimal movement, while others with small lesions led to disproportionately large penalties. This imbalance revealed a core limitation in both binary and raw IoU-based schemes—the lack of consistent scale sensitivity—ultimately motivating the design of more structured reward functions in later stages.

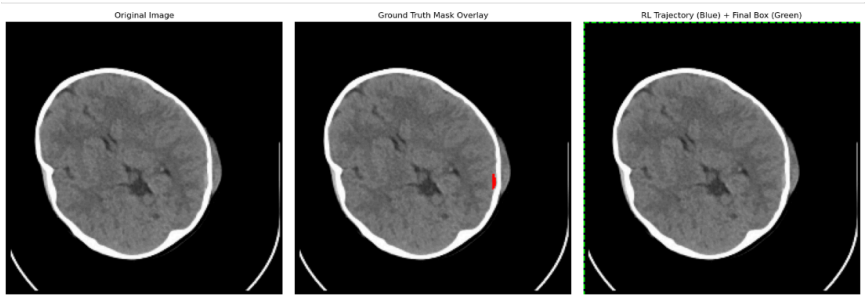

Figure 3: Example output from the IoU difference reward structure, trained on the full dataset. Key parameters include $h = 10$, $\gamma = 0.99$, $\alpha = 0.2$, exponential epsilon decay $\delta = 0.95$, and a trigger threshold of $\tau = 0.6$.

Scale-Aware Reward Design

To address the persistent issue of the agent predicting bounding boxes that were either excessively large or degenerated into small boxes, we developed a series of reward structures aimed at incorporating scale sensitivity into the learning process. These modifications were motivated by the shortcomings observed with binary and IoU difference-based rewards, which failed to consistently guide the agent toward precise and well-scaled localization.

Our initial strategy was an IoU Penalty Area reward, which retained the core principle of rewarding based on the IoU with the ground truth but introduced a scaling factor inversely proportional to the predicted box area. The rationale was to gently discourage the agent from defaulting to full-image or zoomed-out predictions, while still allowing smaller, more focused boxes to be rewarded more generously.

A more refined variation was the IoU Difference + Area and Aspect Ratio Penalty reward, which explicitly penalized the agent when the predicted box diverged too much in scale or shape from the ground truth mask. This was calculated using the logarithmic difference between the predicted and true box areas, as well as their aspect ratios. The goal here was to not only encourage overlap but also enforce spatial consistency, rewarding the agent for matching both the size and the geometry of the lesion.

In these experiments, we kept the hyperparameters—such as history size, epsilon decay, and trigger threshold—consistent to isolate the impact of the reward structure itself. However, despite these motivations, we found the agent often overcorrected and converged to long, narrow boxes ("lines"), suggesting that additional constraints or architectural adjustments might still be needed. Initially, we did some rudimentary experimentation, lowering the “step-size” α to 0.1 in

hopes of allowing the agent to make finer, more precise adjustments—particularly around small lesions—but ultimately found the results indistinguishable from those obtained with $\alpha = 0.2$ A similar outcome occurred with the discount factor γ, which we briefly tested at 0.9 instead of 0.99 in an effort to reduce the influence of long-term rewards. In both cases, the changes had minimal effect on training behavior, and we reverted to using $\alpha = 0.2$ and $\gamma = 0.99$ consistently in favor of tuning other components of the model.

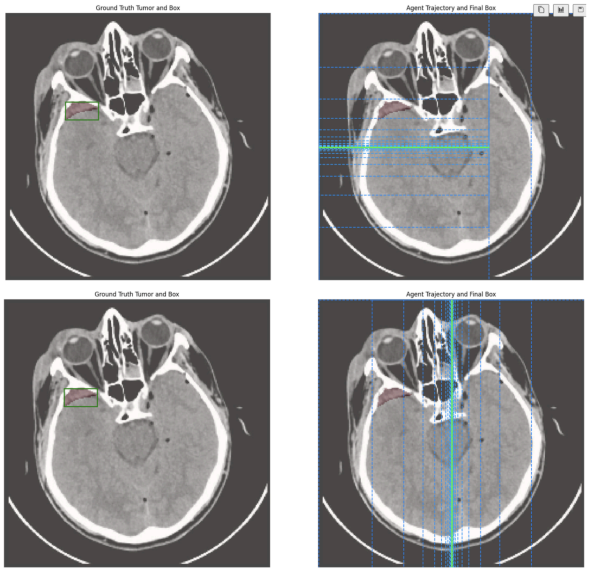

Figure 4: Representative outputs from the penalty-based reward structure. Each row displays a CT scan with the ground truth mask overlaid in red (*left*) and the agent's bounding box trajectory from initial (blue) to final (green) prediction (*right*). Key parameters are the same as previous figures.

Architectural Refinement and Hyperparameter Tuning

As the agent's behavior continued to diverge from our expectations, we shifted focus away from reward design and toward examining the training pipeline itself. Rather than introducing further reward modifications, we explored whether improvements could be made through architectural changes and targeted hyperparameter tuning. During this process, we identified several limitations in the initial DQN implementation that impacted learning stability and overall performance. Notably, although a replay buffer was present, transitions were processed individually rather than in batches, leading to noisy gradient updates and inconsistent convergence. In the revised version, we corrected this by enabling true batch-based training from the replay buffer, allowing for smoother optimization and improved sample efficiency. We also incorporated multi-GPU support, introduced a warm-up phase (a period of non-training during which the agent only collected experience) for more stable early exploration, and refined the experience sampling pipeline. These modifications significantly improved the consistency of training signals and helped the agent converge more reliably across runs.

Additionally, we returned to the original binary reward structure proposed by Caicedo and Lazebnik, both to simplify debugging and because of its demonstrated effectiveness in prior work. This provided a controlled baseline for a series of focused hyperparameter tuning experiments. We began by exploring adjustments known to have a high impact in machine

learning workflows—such as data augmentation (limited to horizontal flips and rotations) and tuning the batch size (set to either 32 or 64). These experiments were conducted on small mini-splits of the dataset to enable rapid iteration. As shown in Figure 5, however, neither augmentation nor batch size had a meaningful effect on convergence or final performance. The agent continued to predict degenerate bounding boxes, suggesting that the underlying learning dynamics—not the input distribution or batch shape—were the primary issue.

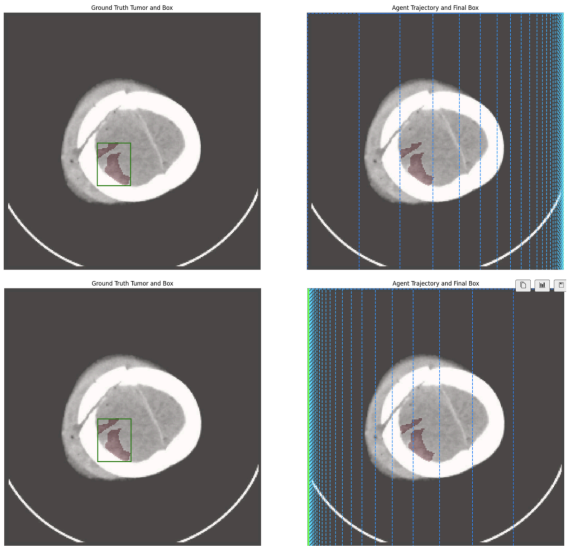

Figure 5: *Top*: Output from a model trained with a batch size of 32 and exponential epsilon decay on a mini-split of the dataset (the shown image is part of the training set). *Bottom*: Output on the same image from a model trained with a batch size of 64 and light data augmentation, using the same mini-split.

This prompted a re-examination of core assumptions in the model's design. One key realization was that the action history, initially set to 10 following Caicedo and Lazebnik's paper (likely inspired by Mnih et al.'s Atari DQN), was somewhat unnecessary in our task. Unlike Atari games where temporal dependencies like momentum are critical, our task could be modeled as a Markov Decision Process (MDP)—where the optimal action depends only on the current state. This insight, echoed in findings by Samiei and Li, led us to reduce the history length to 4 or remove it entirely in later runs, which improved performance consistency.

Additionally, we evaluated different epsilon decay schedules. Our initial use of exponential decay applied per epoch felt intuitive, but we later realized it wasn't the conventional approach; for instance, Mnih et al. achieved strong results using a linear decay tied to the number of training steps. Therefore, we implemented and compared a linear decay strategy, but found it to be largely ineffective in our setting—the agent frequently zoomed out and failed to localize the target altogether. In contrast, our original exponential scheme enabled the agent to steadily refine its predictions and was the only strategy that produced usable results, as shown in Figure 6. This was further supported by training metrics in Figure 7, where exponential decay led to a clear upward trend in total reward per episode, unlike the linear decay schedule, which showed little to no improvement over time. For both experiments, we also shifted to training on a single

image as a sanity check, aiming to confirm that the model was fundamentally capable of learning the desired behavior.

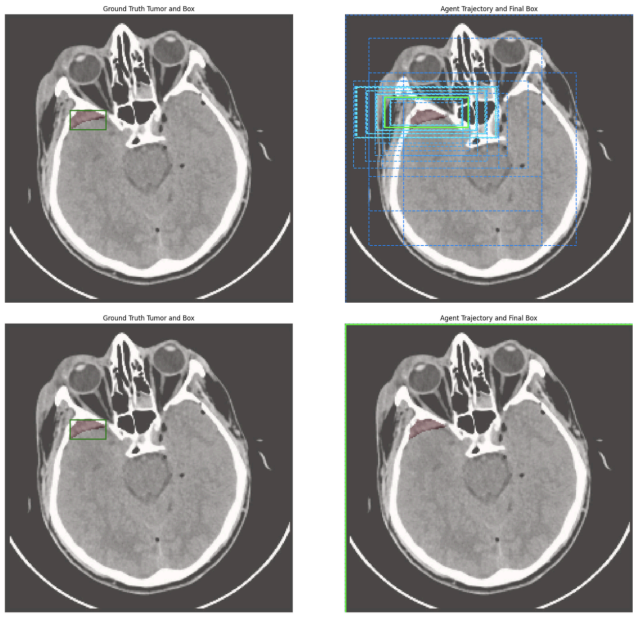

Figure 6: Comparison of epsilon decay strategies on the same training image. *Top*: Exponential epsilon decay $\delta = 0.9$. *Bottom*: Linear decay over 10,000 steps. In both cases, the agent's initial bounding box is shown in blue, gradually transitioning to green as it approaches the final prediction.

However, we still noticed issues with the agent's ability to consistently reach the episode termination threshold τ, even in cases where it successfully overfit a single image. This suggested that the original τ value may have been too strict and poorly aligned with the variability in hemorrhage sizes across the dataset. Specifically, the default setting of $\tau = 0.6$ disproportionately favored large lesions—allowing the agent to terminate episodes with minimal movement—while making success on smaller lesions overly difficult. To better calibrate what constituted a "reasonable" IoU, we generated several bounding boxes with increasing degradation from the ground truth and computed their average IoUs across a set of 10 images. Although Figure 8 shows only one representative example, the printed IoU values reflect averages across this broader subset and helped guide our decision-making. Based on these results, we reduced τ to approximately 0.2 and added a small bonus reward for achieving an IoU greater than 0.6. This adjustment provided a more balanced and scale-aware reward signal, leading to more consistent behavior across diverse lesion sizes.

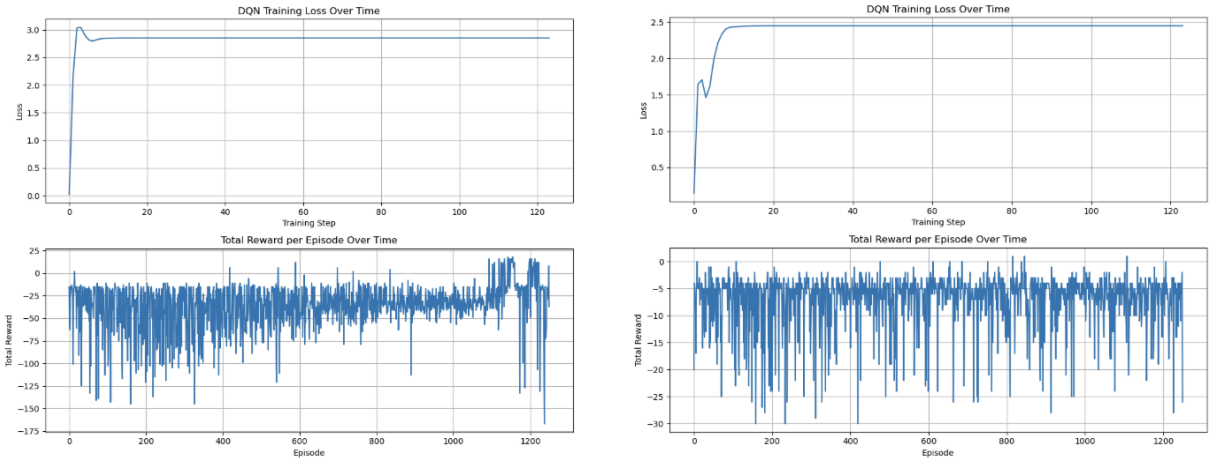

Figure 7: Comparison of training dynamics under different epsilon decay strategies. The left column corresponds to exponential decay ($\delta = 0.9$), and the right column to linear decay over 10,000 steps. For both strategies, the top plot depicts loss per training step, while the bottom plot shows total reward per episode.

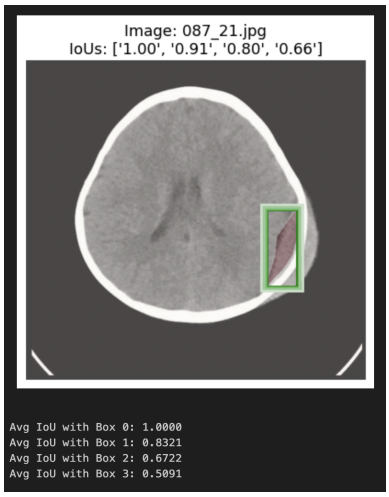

Figure 8: Visualization of IoU sensitivity using bounding boxes of varying quality. Box 0 (dark green) represents the ground truth. Boxes 1 through 3 progressively degrade in quality by shifting 5 pixels farther from the ground truth with each step, shown in increasingly lighter shades of green. The printed average IoU for each box is computed over a set of 10 images.

Geometric Reward Functions

In the final phase of experimentation, we shifted toward designing reward functions that directly leveraged the geometric structure of our bounding box representation—defined by two (x, y) coordinate pairs, one in the top-left corner and the other in the bottom-right. We hypothesized that this explicit spatial formulation could offer a middle ground between the coarse binary or IoU-based rewards and the overly volatile reward dynamics encountered earlier.

The first approach we implemented was the Manhattan Distance reward. At each step, the agent was rewarded for reducing the total Manhattan distance between the predicted and ground truth box corners—summing the absolute differences for all four coordinates. To discourage box collapse, we introduced an additional penalty if any part of the box crossed inward past the

ground truth, such as when the top-left corner moved too far down and right, or the bottom-right corner moved too far up and left. During termination, large rewards or penalties were applied based on the final IoU, with high bonuses for accurate localization and harsh penalties for poor overlap or timeout. While this reward structure introduced substantial variance due to potentially large numerical differences, it enabled the agent to accumulate steadily increasing episode rewards and generalize well to new images, as shown in Figure 9.

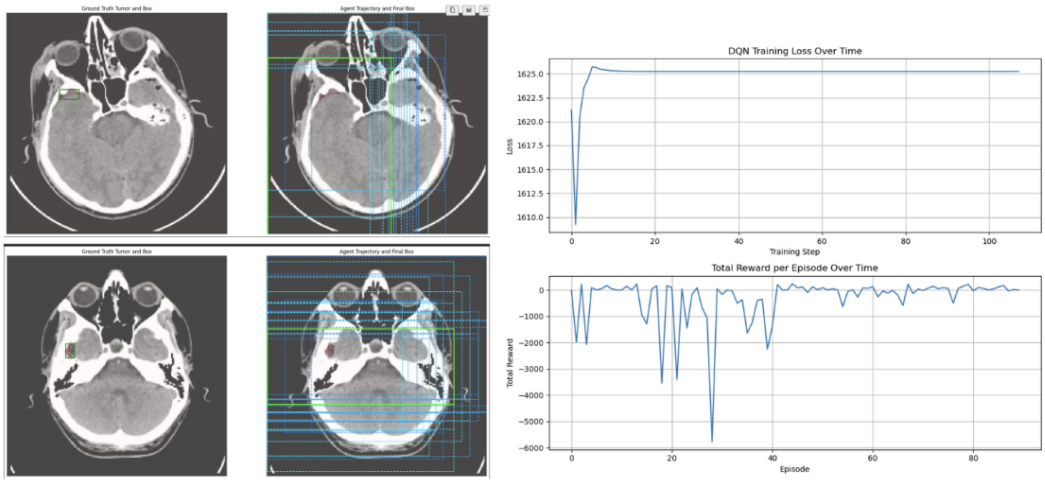

Figure 9: Performance of the Manhattan distance reward system, trained over 25 epochs with exponential epsilon decay ($\delta = 0.9$), $h = 4$, $\alpha = 0.2$, $\gamma = 0.99$, and batch size = 300. *Left*: Two example outputs illustrating hemorrhage localization using the Manhattan-based reward. *Right*: Training metrics, with total reward per episode shown on the bottom and loss per training step shown on the top.

To counteract the instability observed in the Manhattan-based formulation, we designed a second reward function, referred to as “coordinates”. Rather than using a summed distance, this method evaluated each component of each coordinate individually: the agent received a positive reward for moving closer to the corresponding ground truth value, a penalty for moving farther away, and an additional penalty if it overcorrected past the ground truth boundary. Exact matches were given additional rewards, encouraging precision. This formulation resulted in more stable per-step behavior and lower overall loss during training, and performed well on smaller subsets of the data (results not shown). However, we observed that the agent often fell into a repetitive "zoom-in/zoom-out" pattern, possibly due to setting $h = 0$, which removed the action history from the state representation. behavior is visualized in Figure 10, and was further reflected in reward plots, where episode rewards fluctuated significantly even in later training stages.

Together, these geometric reward functions provided more interpretable, coordinate-level feedback while reinforcing the importance of explicitly encoding spatial reasoning into reward design. Although Manhattan distance proved more effective in terms of reward accumulation and generalization, both approaches offered meaningful improvements and a strong foundation for future refinement.

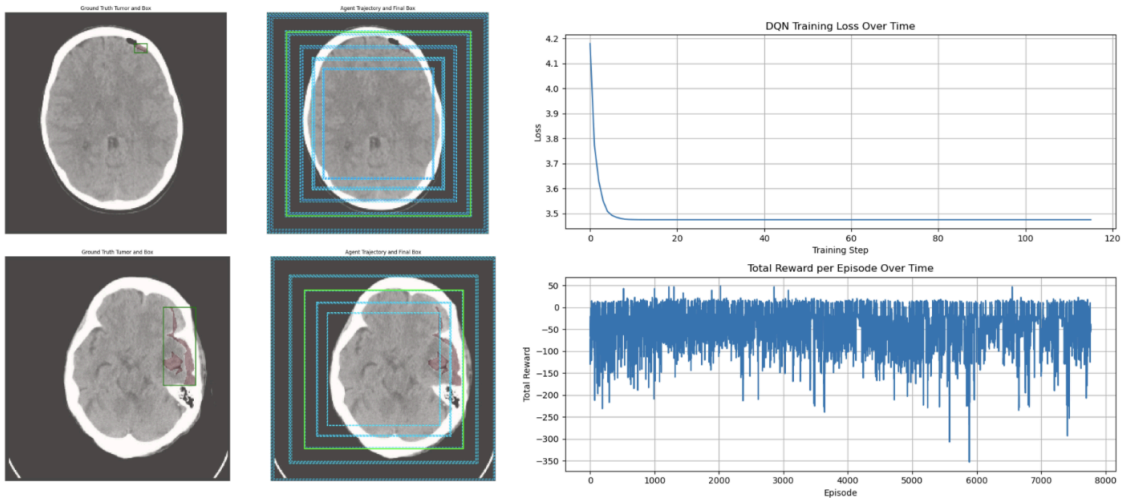

Figure 10: Performance of the Coordinates reward system, trained over 35 epochs with exponential epsilon decay ($\delta = 0.9$), $h = 4$, $\alpha = 0.2$, $\gamma = 0.99$, and batch size = 300. *Left*: Training metrics, with total reward per episode shown on the bottom and loss per training step shown on the top. *Right*: Two example outputs illustrating the agent's zoom-in/zoom-out behavior during hemorrhage localization using the Coordinates-based reward.

Conclusion:

Our experiments demonstrate that both supervised and reinforcement learning approaches come with significant trade-offs when applied to medical object localization. While the U-Net segmentation model provided a useful baseline, it was far from perfect, frequently missing small lesions and likely would have benefited from far more training data. Conversely, the Deep Q-Network (DQN) offered a way to train with less annotated data but required substantial computational time, architectural tuning, and reward shaping to achieve usable results. In the end, neither method offered a definitive solution. There is no free lunch in this domain: one must either invest heavily in data curation for supervised methods or endure long and often unstable training cycles for reinforcement learning.

Training the DQN was particularly demanding. Despite running experiments for days at a time, our agent likely still had room to improve with additional epochs. Reinforcement learners do not converge as cleanly or predictably as their supervised counterparts—the same model might perform well one day and regress the next due to noisy updates and stochastic training dynamics. These challenges underscore that success in reinforcement learning depends not just on algorithmic understanding, but on thoughtful adaptation to the problem domain—particularly in high-stakes settings like medical imaging, where data is limited and interpretability is critical.

A key insight from our work is that simplicity and generalizability in reward design matter more than cleverness or complexity. While some of our early reward structures attempted to shape the agent toward specific box shapes or penalize nuanced geometric deviations, these often introduced brittle behavior that didn't generalize across cases. In contrast, reward functions like our coordinates-based scheme provided clean and interpretable feedback across all four

bounding box coordinates and proved to be more robust. These simpler structures didn't try to encode ideal behavior explicitly but instead created flexible learning signals that worked well across lesion types and image scales.

On the whole, RL remains compelling in its promise: in theory, it can learn to solve almost any task, including those where explicit supervision is difficult to define. However, that power comes with a cost—particularly when it comes to generalization. Throughout this project, we often found that it was easier to teach a human how to read CT scans than to coax an agent into learning the same thing. The RL agent doesn't understand anatomy, symmetry, or spatial context the way a human does, and that lack of inductive bias severely limits its ability to transfer learning across images. Thus, we hypothesize that performance in this setting may depend as much on the quality of the feature extractor as on the design of the DQN itself.

We also reflected on the promise of imitation learning—a technique where agents learn from demonstrations or expert trajectories. While reinforcement learning is theoretically capable of surpassing human strategies, it becomes vastly more sample-efficient when guided by examples of what "good behavior" looks like. Our project did not incorporate any such guidance—no expert demonstrations, no gaze-tracking data, and no precomputed optimal actions. In contrast, many of the prior works we referenced, including those we modeled our framework after, benefited from access to these kinds of auxiliary signals. This difference may help explain why our replication efforts did not yield similarly strong results, despite mirroring their architectures and training procedures.

In hindsight, there are many avenues left to explore. We considered implementing Double Q-learning to mitigate overestimation bias, and debated experimenting with other reward formulations, such as comparing L1 and L2 distance, but ran out of time. These and other ideas remain promising directions for future work. Ultimately, our project underscores both the potential and the limitations of deep reinforcement learning in medical contexts. RL is a powerful tool—but one that requires careful design, extensive iteration, and a deep understanding of both the task and the learning system itself.

Code:

All code is accessible through: github.