Deep Reinforcement Learning with Automated Label Extraction from Clinical Reports Accurately Classifies 3D MRI Brain Volumes

Joseph Nathaniel Stember1 · Hrithwik Shalu2

Received: 22 December 2021 / Revised: 2 March 2022 / Accepted: 20 April 2022 / Published online: 13 May 2022
© The Author(s) under exclusive licence to Society for Imaging Informatics in Medicine 2022

Abstract

Image classification is probably the most fundamental task in radiology artificial intelligence. To reduce the burden of acquiring and labeling data sets, we employed a two-pronged strategy. We automatically extracted labels from radiology reports in Part 1. In Part 2, we used the labels to train a data-efficient reinforcement learning (RL) classifier. We applied the approach to a small set of patient images and radiology reports from our institution. For Part 1, we trained sentence-BERT (SBERT) on 90 radiology reports. In Part 2, we used the labels from the trained SBERT to train an RL-based classifier. We trained the classifier on a training set of 40 images. We tested on a separate collection of 24 images. For comparison, we also trained and tested a supervised deep learning (SDL) classification network on the same set of training and testing images using the same labels. Part 1: The trained SBERT model improved from 82 to 100% accuracy. Part 2: Using Part 1’s computed labels, SDL quickly overfitted the small training set. Whereas SDL showed the worst possible testing set accuracy of 50%, RL achieved 100% testing set accuracy, with a p-value of $4.9 \times 10^{-4}$. We have shown the proof-of-principle application of automated label extraction from radiological reports. Additionally, we have built on prior work applying RL to classification using these labels, extending from 2D slices to entire 3D image volumes. RL has again demonstrated a remarkable ability to train effectively, in a generalized manner, and based on small training sets.

Keywords Deep reinforcement learning · Automated label extraction · 3D MRI brain volumes

Introduction

Classification is widely recognized as one of if not the most important and fundamental tasks in radiology artificial intelligence (AI) [1–3]. As is the case for most radiology AI tasks, the literature on AI radiology classification is dominated by supervised deep learning (SDL). SDL entails an expert user/radiologist manually annotating a large set of images with class labels, typically requiring hundreds or thousands of annotated images for adequate training. SDL often also requires data augmentation to increase the number of images. However, labeling images, particularly so many, is time-consuming and tedious.

To summarize, two key hindrances in SDL are the following requirements:

1. Investigators with domain knowledge, usually diagnostic radiologists, must hand annotate all images.
2. Researchers must retrieve and annotate large numbers of images, typically hundreds to thousands, often requiring data augmentation.

Reinforcement learning (RL), a term that we will use interchangeably with deep reinforcement learning (DRL), may provide solutions to the above challenges. Researchers have recently successfully applied DRL to localizing landmarks or lesions in a range of image types and modalities [4–9]. Notably for example, DRL has shown success in localizing breast lesions [10], lung nodules [11], and anatomic landmarks on cardiac MRI [12, 13] as well as enabling vessel centerline tracing [14]. DRL has also addressed the tasks of segmentation [15–21] and classification [22].

We have, in recent work, employed DRL to classify 2D MRI brain images [24]. In so doing, they were able to
reduce the number of training set images by one to two orders of magnitude. In particular, training a 2D DRL-based image classifier on a mere 30 training set images, the approach achieved perfect accuracy classifying normal versus tumor-containing scans when applied to a separate testing set of 30 images [24].

However, for realistic clinical deployment, we wish to apply classification to full 3D image volumes. Furthermore, we would like to address item one from above; the aforementioned 2D proof-of-principle [24] still required radiologist manual annotation of class labels, although greatly reduced in number to the order of tens. If we could avoid altogether the necessity for manual annotation, this would greatly speed research and allow for new classification models to be trained quickly as the need arises, either for clinical deployment or for research applications.

In this work, we address both of the key challenges posed above, specifically for the task of binary classification, doing so for full 3D image volumes. In Part 1, we extracted image labels automatically from the radiology reports using attention-based natural language processing (NLP). In Part 2, we used the labels generated from Part 1 to train a DRL-based classifier for 3D image volumes of patients who have non-central nervous system primary cancer with either normal MRI brain scans (in the sense of no intracranial lesions) or with scans showing metastatic spread to the brain.

For metastasis-positive patients, the number of metastases ranged from one to over thirty/“innumerable.” Metastases could be infratentorial or supratentorial in location.

We started by randomly selecting 80 normal and 71 metastasis-containing patients/scans. These were initially selected based on the reports but verified on imaging review by neuroradiologist JNS. Upon downsizing from $512 \times 512$ or $256 \times 256$ to $64 \times 64$ pixel slices, many of the tumor-containing images with only smaller metastases were no longer readily classifiable by inspection. We selected the 32 scans for which metastases were still evident to the neuroradiologist. For class balance, we chose equal numbers of tumor-positive and tumor-negative images for training and testing. The training set thus obtained consisted of 40 images, 20 normal, and 20 tumor-containing. The testing set consisted of 24 scans, 12 normal, and 12 tumor-containing.

Random Image Noise

We added random noise to the image slices based on recent results showing improved stability [24]. The intuition was to “strengthen” the Deep Q Network (DQN) to recognize underlying structure in the images beneath the noise. We posit that random noise leverages the more thorough environmental sampling that the Markov Decision Process (MDP) offers.

In particular, to confer stability to our model, we added “salt and pepper” noise to the training set images for use during the MDP. We randomly selected pixels to be converted to black (pixel intensity 0) or white (255). We generated noise for each 2D image slice, replacing from 100 to 200 pixels. The latter represented an upper bound pixel replacement proportion of 5%, low enough to avoid fundamentally altering the image so that it remained identifiable. We show an example of a 2D image slice with noise in Fig. 1.

Methods

Data Collection and Pre-Processing

We obtained a waiver of informed consent and approval from the Memorial Sloan Kettering Cancer Center (MSKCC) Institutional Review Board. After identifying candidate studies through a search of MSKCC radiology archives (mPower, Nuance Communications, Inc.) of MRI brain with and without contrast studies performed between January 1st, 2012 and December 30th, 2018, we extracted two data sets of MRI brain scans, including the reports in a spreadsheet. The extracted data included reports generated by 18 attending-level neuroradiologists at MSKCC. Exclusion criteria included the following:

- T1-post-contrast images not 5 mm in slice thickness.
- Non-metastatic brain abnormalities, such as large meningiomas, primary brain tumors, or post-operative state.
- Cases in which neuroradiologist JNS (2 years of experience) could not readily identify metastases after downsampling of images to $64 \times 64$ pixels (for reasons described below).
Part 1: Automated Label Extraction Using Natural Language Processing

Attention-Based String Encoders, BERT, RoBERTa, and SBERT

We endeavored to extract labels automatically from radiology reports. To do so, we employed a form of the powerful language representation model known as Bidirectional Encoder Representations from Transformers (BERT) [25]. BERT uses a transformer model for encoding and decoding; here, we use only the encoding portion. BERT’s word encoding is essentially a vector that represents all aspects of a word. In addition to representing meaning(s) and connotations of words, the encoding contains information about the word based on the contexts in which it appears. Namely, positional encoding and the attention mechanism [26] imbue the ultimate word encodings with information about other words appearing in the same string (e.g., sentence or paragraph), including the order in which each word appears [27]. Google has pre-trained BERT on a huge corpus of written text using masked language prediction and next sentence prediction.

Robustly Optimized BERT Approach (RoBERTa) [28] is an attention-based word-embedding algorithm like BERT. It has near-identical network architecture to BERT. However, it was trained only for masked language prediction, it has more trainable parameters, and Google has trained it on a larger corpus of text. RoBERTa comes pre-trained on 160 GB of text, including English Language Wikipedia and BookCorpus, containing over 11,000 books. As such, RoBERTa has demonstrated superior performance compared to BERT on a wide array of natural language processing (NLP) tasks. Due to this success, we employed RoBERTa for the current project.

The builders of BERT/RoBERTa intended to represent words for sequence-to-sequence tasks such as translation. However, they did not design them for representational learning of sentences, phrases, or paragraphs. Since we sought to capture the overall sentiment of a report impression section, we required a mechanism to encode vectors with such information. Sentence BERT (SBERT) [29] is an adaption of BERT/RoBERTa that is more efficient at extracting the sentiment of whole sentences. It combines the word encodings from BERT/RoBERTa via dense interpolation/pooling, maintaining word meanings while minimizing representational vector size. We downloaded the pre-trained version of RoBERTa/SBERT via the Python package, sentence-transformers 0.3.0.

Application of SBERT with Pre-Trained Weights and Transfer-Learning of SBERT to Radiology Reports

As mentioned above, RoBERTa, using pre-trained weights, generated the word encodings stacked into matrix form. Then the SBERT dense pooling of these provided matrices, the rows of which represented encoded report impressions.

We now distinguish the testing set for Part 1, the natural language portion of this study, from that of Part 2. The testing set for Part 1 is the set of 64 report impressions associated with the cases/image volumes used in Part 2. In Part 2, we partition into 40 training and 24 testing set images. Both the Part 2 training and testing sets are equally split between normal and metastasis-containing, providing class balance. The predicted labels from Part 1 comprise the class labels in Part 2 for both the training and testing set scans.

Applying SBERT with only the pre-trained RoBERTa parameters to these 64 report impressions gave 100% accuracy for tumor-free images and 64% accuracy for those with brain metastases. The result makes intuitive sense because negative reports tend to be shorter, less variable, more declarative, and more easily identifiable. Two typical examples:

- “Normal study”
- “No intracranial metastasis or acute abnormality”

The report impressions for positive cases, in contrast, are typically longer and more variable. They are more imbued with the particular vocabulary used by neuroradiologists. We would not expect this jargon to follow the type of language in the corpora from which the pre-trained weights derive. We hence set out to improve this accuracy by transfer learning specific to radiology reports.
To avoid mixing up the report impressions from the images we were analyzing for classification with the impressions on which we transfer-learned SBERT, we used a separate set of 90 impressions for the latter task. Half of these were for lesion-free brain images, and the other half were for metastasis-containing brain MRIs, ensuring class balance.

To further train, i.e., perform transfer learning with SBERT on our radiological report impression training set, we generated all possible combinations of impressions, labeling these as the same or different meaning/category. Since this amounts to choosing all unique pairs out of a set of 90 elements (90 choose 2), the total number of generated pairs is given by:

\[
N_{\text{pairs}} = \binom{90}{2} = 4,005
\]  

(1)

Hence, we generated a sizeable training set from a relatively modest number of report impressions. Training SBERT on our impressions took 57 min and 22 s. After transfer learning on the radiology-specific jargon and prose of the reports, SBERT achieved 100% accuracy for both normal and metastasis-containing cases. Hence, the improvement provided by transfer learning was that SBERT learned the more nuanced and radiology-specific jargon of metastasis.

We can state this result a bit more formally if we denote by \( l_{\text{radiologist}} \) the consensus class label between original clinically interpreting/reporting neuroradiologist and researching neuroradiologist (2 years of experience). We can also denote the class label predicted by the transfer-learned SBERT as \( l_{\text{SBERT}} \). We can signify the size of Part 2 report impressions/images as \( N_{\text{total}} = 64 \), given the aforementioned 100% accuracy of SBERT further trained on our radiology report corpus, we have:

\[
\{l_{\text{radiologist}}\}_{i=1}^{N_{\text{total}}} = \{l_{\text{SBERT}}\}_{i=1}^{N_{\text{total}}}
\]  

(2)

Then, given their equivalence, we may henceforth refer to the true class label for image \( \mathcal{M} \) as \( l_{\mathcal{M}} \). Again, we note that the testing set of report impressions in Part 1 corresponds to the training and testing sets of Part 2. In the latter, we use \( l_{\text{SBERT}} \) as the labels accompanying the states that are primarily comprised of the image volumes to predict image class.

**Part 2: Reinforcement Learning Using Labels from Part 1 to Classify 3D Images into Normal and Tumor-Containing**

**Reinforcement Learning Environment, the Definition of States, Actions, and Rewards**

As in all RL formulations, we must define our environment, itself defined by the agent, possible states, actions, and rewards.

**Agent**

The agent in any RL formulation is an intuitive but formally murky concept. Essentially, the agent performs actions in various states and receives certain rewards. The agent’s goal is to act according to a policy (prescribed set of actions) that maximizes the total expected cumulative reward received. Our and the agent’s goal is to learn the optimal policy for the agent to follow.

**States**

We define the state as a tuple containing an image volume and a scalar quantity that indicates whether the previous class prediction (normal vs. tumor-containing) was correct. We denote this prediction correctness by \( \text{pred}_\text{corr} \). It is defined by:

\[
\text{pred}_\text{corr} = \begin{cases} 
1, & \text{if prediction is correct} \\
0, & \text{if prediction is wrong} 
\end{cases}
\]  

(3)

Or equivalently:

\[
\text{pred}_\text{corr} = \delta_{a,l_{\mathcal{M}}},
\]  

(4)

where \( \delta \) is the Kronecker delta function and "a" is the predicted action. Hence, we can write the state \( s \) more formally as:

\[
s = \{ \mathcal{M}, \text{pred}_\text{corr} \},
\]  

(5)

where \( \mathcal{M} \) is the 64 \( \times \) 64 \( \times \) 36-voxel matrix representing the image volume. \( \mathcal{M} \) is a \( T1 \)-weighted post-contrast image volume with a slice thickness of 5 mm. The dimensions of \( \mathcal{M} \) have to be uniform for all images to allow for 3D convolutions:

- We downsize axial slices into a uniform size to make all matrices the same size and render the 3D convolutional backpropagation more computationally tractable. In particular, slices are either 512 \( \times \) 512 or 256 \( \times \) 256 pixels, and we down-sample all to 64 \( \times \) 64 pixels.
- Additionally, axial slices range in number from 28 to 36. We add zero matrices of size 64 \( \times \) 64 pixels to the caudal part of the images so that all have \( z \)-dimension of 36. This is essentially padding in the \( z \)-direction.

Again, the uniform dimensions of 36 \( \times \) 64 \( \times \) 64 allow for 3D convolutions used in the Deep Q network, described later.
**Actions**

The two possible actions, \( a_1 \) and \( a_2 \), are simply the prediction of whether image \( M \) is normal (in the sense of no tumor present) or tumor-containing; i.e., the actions \( \mathcal{A} \) are given by:

\[
\mathcal{A} = \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \text{predict normal} \\ \text{predict tumor} \end{pmatrix}
\]  

(6)

**Rewards**

To encourage correct class predictions, we employ a reward that incentivizes correct predictions. Hence, we provide a reward of +1 if the prediction is correct and penalize with a reward of −1 for a wrong prediction. We can define the reward \( r \) in terms of prediction correctness, \( \text{pred\_corr} \):

\[
\begin{align*}
  r &= \begin{cases} 
  -1, & \text{if } \text{pred\_corr} = 0 \\
  +1, & \text{if } \text{pred\_corr} = 1.
\end{cases}
\end{align*}
\]  

(7)

**Action-Value Function (Q)**

A fundamentally important quantity in RL is that of a value function, in our case taking the form of the action-value function, denoted by \( Q(s, a) \). \( Q(s, a) \) essentially tells us the expected total cumulative reward that the agent would receive by selecting a particular action given a specific state, then acting in an optimal (on-policy) manner afterward until the end of the current episode. We define an episode as the number of states after which the agent begins anew in a different initial state. We can restate the action-value function’s definition as the expected cumulative reward for taking action \( a \) while in state \( s \) and then following a given policy (set protocol for selecting actions) until the episode’s end. More formally, given a policy \( \pi \), the corresponding action-value, \( Q^\pi(s, a) \), is defined by:

\[
Q^\pi(s, a) = E_{\pi} \left\{ R_t \mid s_t = s, a_t = a \right\} = E_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\}
\]

(8)

where \( R_t \) is the total cumulative reward starting at time \( t \) and \( E_{\pi} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right\} \) is the expectation for \( R_t \) upon selecting action \( a \) in state \( s \) and subsequently picking actions according to \( \pi \). The discount factor \( \gamma \) represents the trade-off between immediate rewards (“instant gratification”) with rewards later on (“delayed gratification”). We set it equal to 0.99, as we have in prior work.

The action-value function is critically important because by maximizing this quantity, we can ultimately reach an optimal policy that produces the desired behavior, correctly predicting image class.

**Deep-Q Network**

To predict the actions our agent will take, we use the Deep-Q Network (DQN), as prior research has done [23, 30–32]. We illustrate the DQN architecture in Fig. 2.

Our DQN consists of two arms/branches that join toward the end. In one branch, the image volume serves as the input, undergoing a 3D convolution. An additional convolutional layer is flattened and followed by three fully connected layers.

In a separate, parallel pathway, \( \text{pred\_corr} \) is passed to a flattened layer, which is then concatenated with the last fully connected layer of the image volume network branch. This concatenated layer then connects to a two-node output.

The two-node output represents the two possible action-values, \( Q(s, a_1) \) and \( Q(s, a_2) \), that result from taking actions \( a_1 \) and \( a_2 \), respectively, from state \( s \). Again, since we wish to maximize the total cumulative reward, we should maximize \( Q(s, a) \). We do so simply by selecting the \( \text{argmax}_a(Q) \), thereby picking the action that maximizes the expected cumulative reward.

We should note at this point that selecting \( \text{argmax}_a(Q) \) represents “on-policy” action selection. As we will see and has been described in prior work [23].

Indeed, an initial conundrum/catch-22 is that we wish to train our network to approximate the optimal policy’s \( Q \) function, but at first, we have no idea what that optimal policy is. We can only start to learn it in pieces by sampling from the environment via off-policy exploration and temporal difference learning.

**Temporal Difference Learning**

While the DQN allows us to select actions for our agent to take, we need to learn the best policy via the process of taking actions and receiving rewards. The rewards in particular tell us about our environment. We have to store these “experiences” of the agent to understand the environment better. By doing so in tandem with the use of DQN, we can train the parameters of the DQN to build it into a reliable approximator for the action-value function \( Q(s, a) \). However, the optimal policy, and thus best possible action-value function, is a moving target that we approach through sampling.

Then, using the DQN \( Q \) function approximator for an ever-improving policy’s \( Q \) function, we can better explore and increasingly exploit what our agent knows about the environment to sample the environment more efficiently. The result of this process is that the DQN becomes a better...
and better approximator for not only a \( Q(s, a) \) function, but ultimately in theory the “global” optimal \( Q(s, a) \) function, denoted by \( Q^* \). In practice, the exploration continues throughout our training since the goal is to find “local” optimal policies that produce high testing set accuracy.

In general, an episode of training proceeds as follows:

We select one of the training set images at random. We guess the initial predicted class with equal probability for normal or tumor-containing. We set the initial value of \( \text{pred}_{corr} \) to zero by default. We choose the agent’s actions from the DQN’s outputs via the epsilon-greedy algorithm for all subsequent steps of the episode. We have used the approach and described it in more detail in prior DRL work [23, 30–32]. However, briefly, the method selects actions initially more at random to explore the environment. As the agent learns about the environment and the best policy to follow, the degree of randomness decreases, and the agent increasingly chooses the optimal action predicted by the DQN, i.e., \( \text{argmax}_a \{ Q(a = 0), Q(a = 1) \} \). In other words, earlier in training, we emphasize exploration, while later in training, we increasingly employ exploitation.

We use similar epsilon-greedy algorithm parameters as in the referenced DRL work, namely:

\[
\begin{align*}
\epsilon &= 0.7 \\
\epsilon_{\text{min}} &= 1 \times 10^{-4} \\
\Delta \epsilon &= 1 \times 10^{-4},
\end{align*}
\]

where \( \epsilon \) is the initial exploration rate, which decreases during training at a rate of \( \Delta \epsilon \) down to a minimum value of \( \epsilon_{\text{min}} \).

Again, although RL in theory always converges toward an optimal global policy, this is typically not practically feasible. However, that is acceptable given our goal of identifying “locally” optimal policies that produce high-accuracy testing set predictions. In fact, the training duration of 2000 episodes results in a final \( \epsilon = 0.5 \). Even at the end of our training, the agent performs random environmental exploration as much as exploitation.

In recent work [23], we have shown that multi-step episodes provide more thorough environmental sampling than single-step episodes. We showed that multi-step episodes, implemented as Markov Decision Processes (MDP), produce locally optimal policies with higher testing accuracy. Hence, we employ the MDP in the present work.

We proceed as follows: once we take the action from the previous step, this will bring us, at time \( t \), from the state \( s_t \) to \( s_{t+1} \) and will provide a reward \( r_t \). We can store all of these values in a tuple called a transition. As described in prior RL work [23, 30–32], we store these up to a maximum memory size called the replay memory buffer. Here, to conserve space and save more transitions in the replay memory buffer, we store state information solely based on the training image number and correctness of the prior and just-made predictions, \( \text{pred}_{corr}_{t-1} \) and \( \text{pred}_{corr}_t \), respectively. We also store the index of the training image, \( I_{\text{image}} \), which is all we need to retrieve the actual image volume \( \mathcal{M} \) and class label \( l_{\mathcal{M}} \). As such, the transition at time \( t \), denoted by \( T_t \), is \( T_t = (\text{pred}_{corr}_{t-1}, I_{\text{image}}, a_t, r_t, \text{pred}_{corr}_t) \). Then the training batch for the DQN is selected randomly from the
replay memory buffer, consisting of 15,000 transitions, stored as rows in a transition matrix.

Earlier work with 2D image slice classification [23] included information about the previous step as a color overlay on grayscale images. To save space by not storing the image itself with color overlay, we simply record the value for prediction correctness, \( \text{pred\_corr} \) (0 if the prior prediction was incorrect, 1 if it was correct). The default starting value for \( \text{pred\_corr} \) in the initial state is zero. Then, if our agent has made the right prediction, i.e. taken the correct action, we set \( \text{pred\_corr} = 1 \); otherwise, it remains zero. We illustrate this Markov decision process (MDP) in Figs. 3 and 4.

These figures also show an additional modification of the Markov decision process used here: while the initial state in the episode contains the noiseless grayscale image volume, subsequent steps use states with random noise added to the image slices, as in Fig 1.

The MDP, as shown in Figs. 3 and 4, proceeds as five steps per episode of training. At each step, we acquire data that samples the environment and which we use to train the DQN. Each step of the MDP produces pieces of data about the environment storable as transitions.
The environment provides the agent with a reward of +1 for taking the correct action/class prediction and penalizes the agent with a reward of −1 for a wrong action/prediction. We demonstrate the reward scheme in Figs. 3 and 4.

**Training: Deep-Q Network Architecture, Parameters, Hyperparameters**

Having verified 100 % accuracy for label extraction, we trained an RL approach based on deep Q networks (DQNs) and TD(0) temporal difference learning, as in our recent work classifying 2D image slices [23]. To use TD(0), we employed the five-step Markov decision process illustrated in Figs. 3 and 4.

As shown in Fig. 2, for our DQN, we used a two-input architecture. On one branch, the image volumes, again of size 64 × 64 × 36 voxels, were fed into a 3D convolutional neural network as input. The 3D convolutional kernels were of size 5 × 5 × 5. As per usual, we initially randomized the weights in the kernels as per the standard Glorot distribution. We set the stride and padding at (2, 2, 2) and (1, 1, 1), respectively. The first convolutional layer has 32 channels, followed by a 64-channel layer. We applied Elu activation after each convolutional step. We then flattened the second convolutional and passed it to a fully connected layer of size 512 nodes, itself connected to a 256- then 64-node fully connected layer.

Separately, the scalar pred_corr, defined in Eqs. (3) and (4), was connected to a 64-node fully connected layer. We then concatenated this layer with the 64-node fully connected layer from the image convolutions. In this manner, we infused the DQN with information about the image itself and whether the preceding image class prediction was correct.

Finally, we connected the concatenated 128-node layer to two output nodes, Q(a = 0) and Q(a = 1). Most of the hyperparameters we used for training are similar to or the same as in prior RL work [23, 30–32]:

- Loss: mean squared error
- Adam optimization
- Learning rate: 1 × 10⁻⁴
- Batch size: 24
- Training time: 2000 episodes

For each episode of training, the image/case was selected randomly from the training set. We ran individually through all of the testing set images for predictions on the testing set after each set of 10 training episodes.

Total training time was roughly 10 Google Colab GPU hours. We coded the approach in Python, using PyTorch for network design.

**Testing of Reinforcement Learning**

We applied the DQN to each testing set input state, comprising the testing set image and an initial pred_corr defaulted to zero. We applied the DQN for one step of forward passage. Then, we computed the predicted class as \(\text{argmax}_a(Q(s, a))\).

**Supervised Deep Learning (SDL) Classification for Comparison**

To compare SDL and RL-based classification, we trained the same 90 training set images with a supervised deep learning (SDL) CNN with architecture essentially identical to the convolutional branch of the DQN from Fig. 2. The CNN also consisted of 3D convolutional layers followed by Elu activation employing 5 × 5 × 5 filters. As for the DQN, we followed with three fully connected layers. The network output a single node. We passed the node through a sigmoid activation function given the direct binary nature of the CNN’s prediction (normal vs. tumor-containing). The loss used here was binary cross-entropy. Other training hyperparameters were the same as for the DQN. We trained the SDL CNN for 100 epochs.

**Results**

**Part 1**

As stated above, transfer-learning of SBERT to radiology reports resulted in 100 % accurate label predictions based solely on the set of reports corresponding to the images analyzed in Part 2. Hence, we were able to use these as gold standard labels in Part 2.

**Part 2**

SDL classification, after training, predicted only class, tumor-containing, for the entire testing set, for an accuracy of 50 %. Hence, SDL was unable to discern a difference between normal and tumor-containing images. Essentially, the trained network could not distinguish between the two classes and defaulted to the “second” class (label of one for metastasis-containing images).

RL achieved an accuracy of 100 %. This difference in accuracy between SDL and RL, displayed in Fig. 5, is statistically significant, with a \(p\)-value of 4.9 × 10⁻³. We computed the \(p\)-value via McNemar’s test, which accounts for frequencies of false positives and false negatives and is widely used to compare machine learning classifiers [33]. We also note that, due to the class balance between normal
and tumor-containing images in the testing set, percent accuracy by itself is an appropriate and sufficient statistical measure for model performance.

**Discussion**

**Part 1: NLP Automated Label Prediction**

We have shown that an SBERT-based NLP algorithm can produce outstanding results when trained with the specific vocabulary and phrasing of oncologic neuroradiology. The trained network enables automated label extraction directly from radiology reports. When coupled with image extraction, this can allow for partial or fully automated training of classification algorithms, with little-to-no requirements from the user/researcher.

When we applied the model with weights trained on more general text corpora such as Wikipedia, they were perfectly accurate for normal scan report impressions. However, they did worse for report impressions detailing brain metastases. The underperformance probably arose from the fact that, in our experience, there are fewer ways for radiologists to communicate normalcy, for example, “normal scan” or “no intracranial metastasis.” The language is simple, straightforward, and does not vary much between interpreters. However, reporting a positive finding like new or increased metastases involves many possible scenarios and ways of phrasing the findings. The vernacular of radiologists becomes more prevalent, and the variability between radiologist phrasing increases. Hence, these were the scans for which SBERT improved dramatically by training specifically on radiology report impressions.

**Part 2: Image Classification**

We have extended our prior work [23], here building a DRL classifier for fully 3D image volumes and applying it to scans from our institution. The statistically significant improvement for DRL’s 100% testing set accuracy vs. the 50% of SDL fits with the recent results showing much better performance of DRL when trained on small training sets. RL can generalize more by exploring the data in a more exhaustive way that mimics how humans learn. In contrast, SDL employs a fragile “memorization” process that does not generalize well to, i.e., “breaks” in new situations.

Some important limitations apply to this work. Due to the considerable computational cost of 3D convolutions, we had to downsize our images twofold to fourfold, producing pronounced spatial and contrast resolution loss. Future work will seek to speed up the training process and eschew the downsizing, noting that brain metastases are often rather small and subtle, and in actual deployment, downsizing is not an option.

**Author Contribution** Both authors contributed to the study conception and design, data curation/processing, and all components of the analysis. Both authors contributed to the manuscript. Both read and approved the final/submitted version of the manuscript.

**Funding** We gratefully acknowledge support from the following sources: American Society of Neuroradiology Research Grant in Artificial Intelligence, Canon Medical Systems USA, Inc./Radiological Society of North America Research Seed Grant, Memorial Sloan Kettering Cancer Center Radiology Developmental Project Fund.
Declarations

Conflict of Interest The authors have pursued a provisional patent and conversion to full patent based on work including that described here.

References

1. Mcbee, M. P. et al. Deep learning in radiology. Academic radiology 25, 1472–1480 (2018).
2. Saba, L. et al. The present and future of deep learning in radiology. European journal of radiology 114, 14–24 (2019).
3. Mazurowski, M. A., Buda, M., Saha, A. & Bashir, M. R. Deep learning in radiology: An overview of the concepts and a survey of the state of the art with focus on MRI. Journal of magnetic resonance imaging 49, 939–954 (2019).
4. Parekh, V. S., Braverman, V., Jacobs, M. A., et al. Multitask radiological modality invariant landmark localization using deep reinforcement learning in Medical Imaging with Deep Learning. Proceedings of the Third Conference on Medical Imaging with Deep Learning, in Proceedings of Machine Learning Research (2020), 588–600.
5. Alansary, A. et al. Evaluating reinforcement learning agents for anatomical landmark detection. Medical image analysis 53, 156–164 (2019).
6. Ghesu, F.-C. et al. Multi-scale deep reinforcement learning for real-time 3D-landmark detection in CT scans. IEEE transactions on pattern analysis and machine intelligence 41, 176–189 (2017).
7. Zhou, S. K., Le, H. N., Luu, K., Nguyen, H. V. & Ayache, N. Deep reinforcement learning in medical imaging: A literature review. arXiv preprint arXiv:2103.05115 (2021).
8. Al A. W. & Yun, I. D. Partial policy-based reinforcement learning for anatomical landmark localization in 3D medical images. IEEE transactions on medical imaging 39, 1245–1255 (2019).
9. Blair, S. I. A. S. A., White, C. & Moses, L. D. D. Localization of lumbar and thoracic vertebrae in 3D CT datasets by combining deep reinforcement learning with imitation learning (2018).
10. Maicas, G., Carneiro, G., Bradley, A. P., Nascimento, J. C. & Reid, I. Deep reinforcement learning for active breast lesion detection from DCE-MRI. International conference on medical image computing and computer-assisted intervention (2017), 665–673.
11. Ali, I. et al. Lung nodule detection via deep reinforcement learning. Frontiers in oncology 8, 108 (2018).
12. Jang, Y. & Jeon, B. Deep Reinforcement Learning with Explicit Spatio-Sequencal Encoding Network for Coronary Ostia Identification in CT Images. Sensors 21, 6187 (2021).
13. Codari, M. et al. Deep reinforcement learning for localization of the aortic annulus in patients with aortic dissection. International Workshop on Thoracic Image Analysis (2020), 94–105.
14. Zhang, P., Wang, F. & Zheng, Y. Deep reinforcement learning for vessel centerline tracing in multi-modality 3D volumes. International Conference on Medical Image Computing and Computer-Assisted Intervention (2018), 755–763.
15. Winkel, D. J. et al. Validation of a fully automated liver segmentation algorithm using multi-scale deep reinforcement learning and comparison versus manual segmentation. European journal of radiology 126, 108918 (2020).
16. Winkel, D. J., Breit, H.-C., Weikert, T. J. & Stieltjes, B. Building large-scale quantitative imaging databases with multi-scale deep reinforcement learning: initial experience with whole-body organ volumetric analyses. Journal of Digital Imaging 34, 124–133 (2021).
17. Li, Z. & Xia, Y. Deep reinforcement learning for weakly-supervised lymph node segmentation in CT images. IEEE Journal of Biomedical and Health Informatics 25, 774–783 (2020).
18. Yin, S., Han, Y. & Li, S. Left Ventricle Contouring in Cardiac Images Based on Deep Reinforcement Learning. arXiv preprint arXiv:2106.04127 (2021).
19. Si, X. et al. Multi-step segmentation for prostate MR image based on re-inforcement learning in Medical Imaging 2020: Image-Guided Procedures. Robotic Interventions, and Modeling 11315 (2020), 113152R.
20. Xiong, J. et al. Edge-Sensitive Left Ventricle Segmentation Using Deep Reinforcement Learning. Sensors 21, 2375 (2021).
21. Zhang, D., Chen, B. & Li, S. Sequential conditional reinforcement learning for simultaneous vertebral body detection and segmentation with modeling the spine anatomy. Medical Image Analysis 67, 101861 (2021).
22. Kooi, T. et al. A comparison between a deep convolutional neural network and radiologists for classifying regions of interest in mammography. In-ternational Workshop on Breast Imaging (2016), 51–56.
23. Stember, J. & Shalu, H. Deep reinforcement learning-based image classification achieves perfect testing set accuracy for MRI brain tumors with a training set of only 30 images. arXiv preprint arXiv:2102.02895 (2021).
24. Stember, J. Comparison of Contextual Bandits versus Markov Decision Process Reinforcement Learning for MRI brain classification (Aug. 2021).
25. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018).
26. Vaswani, A. et al. Attention is all you need. arXiv preprint arXiv:1706.03762 (2017).
27. Jawahar, G., Sagot, B. & Seddah, D. What does BERT learn about the structure of language? ACL 2019–57th Annual Meeting of the Association for Computational Linguistics (2019).
28. Liu, Y. et al. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019).
29. Reimers, N. & Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv preprint arXiv:1908.10084 (2019).
30. Stember, J. & Shalu, H. Deep reinforcement learning to detect brain lesions on MRI: a proof-of-concept application of reinforcement learning to medical images. arXiv preprint arXiv:2008.02708 (2020).
31. Stember, J. & Shalu, H. Unsupervised deep clustering and reinforcement learning can accurately segment MRI brain tumors with very small training sets. arXiv preprint arXiv:2012.13321 (2020).
32. Stember, J. N. & Shalu, H. Reinforcement learning using Deep Q Networks and Q learning accurately localizes brain tumors on MRI with very small training sets. arXiv preprint arXiv:2010.10763 (2020).
33. Dieterich, T. G. Approximate statistical tests for comparing supervised classification learning algorithms. Neural computation 10, 1895–1923 (1998).

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.