Scaling the deep RL on landmark localization

Viska Wei
swei20@jhu.edu

***Abstract*—In recent years, increasingly successful applications of machine learning to the field of medical imaging strongly enhance the confidence in building an AI-powered clinical decision-support systems. Especially, deep reinforcement learning has emerged as one of the best techniques in developing the system for its outstanding performance at landmark localization related tasks. However, these RL applications to date still face many challenges: (1) State space with large cardinalities and high dimensions are emerging; (2) Clinical data are often geo-distributed, direct aggregation and training become impossible due to memory constraints and transfer costs. (3) Free movement of sensitive and confidential medical data is restricted by clinic's regulations and state's law. (4) Trained RL agents can only perform a single task, such as kidney localization, failed to generalize to multitask. In this paper, we address these challenges by building a count sketch preprocessor to the existing deep reinforcement learning framework [4].**

We show this augmentation of deep RL to be fully parallel, scale linearly in time, sublinear in memory and communication, making it possible to obtain lower embedding of high dimensional state space of many millions, potentially billions of data points, spread across hospitals and clinics around the globe. We demonstrate the power of our method on one midsize dataset: cancer data with 52 million 35-band pixels from multiplex images of tumor biopsies. We believe this distributed deep reinforcement learning framework for privacy-preserving medical data training can pave way to AI-powered clinical decision-support systems

***Index Terms*—deep Q learning, landmark localization, state space embedding, count sketch,**

I. Introduction

The field of medical imaging is moving fast towards a collaborative space between human experts and AI. In particular, deep reinforcement learning has emerged as one of the best techniques for landmark localization across multiple imaging researches. However, there're still challenges remain to be tackled.

A. Large and High Dimensional State-Action Space

Firstly, Data sets with very large cardinalities are emerging (i.e one slice of a cancer biopsy image contains 52M pixel by 35 layers). Standard procedure for analyzing any cells/tissues involves first staining the tissue with 7-12 different fluorescent markers/dyes and then observing tissue with 5-7 different broad-band filterers and 20nm wide narrow-band filters. As a result, each pixel in the tissue image is in $5 \times 7 = 35$ to $7 \times 12 = 84$ - dimensional space. Keeping a state space of such high dimensional data is computationally expensive. Note that one of the key features in landmark localization is experience replay that typically keeps a replay buffer containing millions of state-action-reward-nextState tuple. The tuple would be living in a space with $O(10^2)$ - dimensions. In worse case, simply loading this data to memory would be problematic, let along training a neural net with it.

Hypothetically, to survive the memory constraint, we can embed the high dimensional state space into lower dimensions before feeding the data into a deep RL algorithm. For instance, we could use principal component analysis (PCA) to get a 2-4D lower dimensional embedding of the data, thus a simpler representation of the states. Yet, doing so would significantly hinder the learning process due to the degeneracy of states, limitation of linear embeddings and most importantly the curse of dimensionality.

Another idea would be using non-linear embeddings like tSNE [2] or UMAP [3]. These dimensional reduction tools use local relationships between points to create a low-dimensional mapping, which renders better results than linear ones like PCA. In recent years, non-linear embeddings have became a common practice to navigate data scientists towards better understanding of local and global structures within the dataset while working with more comprehensible two- or three dimensional plots. The problem with these powerful embedding tools is that they scale poorly in time and memory. While recent optimizations showed successful handling of 10,000 data points, scaling beyond million points is still challenging. Their typical run time scale with n^2. Even recent improvements to $n \log(n)$, along with new parallel techniques, cannot handle data in millions, let alone in billions or trillions.

To summarize, although most of the high dimensional medical imaging do have strong clustering with high density contrasts, forming a highly representative lower embedding state or state-action space is still challenging.

B. Geo-distributed Data

Secondly, medical data today are often geo-distributed. Overfitting occurs when each clinics or institution conducts computations with its own local data. Moreover, data imbalance and other associated learning issues might occur as the amount of data in each platform are not equal [5]. If the data happened to be very large, not only the cost to transfer is high, the aggregation is also impossible since all data must reside on the same server for that.

C. Private and Sensitive Data

Meanwhile, due to policies and regulations, free movement of data between clinics especially sensitive and private data is also challenging. In some countries or states, gathering and transporting patient data between research centers and hospitals is strictly restricted by laws. Such concerns for the training of RL agent in segregated large-scale data are already present in current Covid-19 research, e.g. aggregating mobility data between cellular providers in different countries.

D. *Generalization of RL Agent*

Currently, the models developed using deep RL for landmark localization have been limited to a single application. As a result, to develop a radiological decision support system, hundreds and thousands of deep learning models, each targeting a specific task or organ, need to be trained. This is neither memory nor time efficient. A more realistic approach towards an AI-powered diagnostic system is to train a single adaptable and generalizable agent that can perform multiple tasks.

E. *Distributed Deep RL Framework*

To address these challenges, we combine the ideas in two amazing papers. We propose that by scaling the "multitask modality invariant deep reinforcement learning framework" developed in [4] with a "Sketch & Scale" pipeline in [6], we could handle at ease the big, high dimensional, geo-distributed, private medical data of multiple tasks/organs and imaging parameters. With this augumented RL framework, we lay the foundation to AI-powered medical decision support system.

II. Model

In this section we present the vanilla 2D single-agent deep Q-learning(DQN) landmark localization algorithm. All images are sourced from this paper [4].

A. *Vanilla Landmark Localization Algorithm*

Environment The environment is the radiological image (Figure 1).

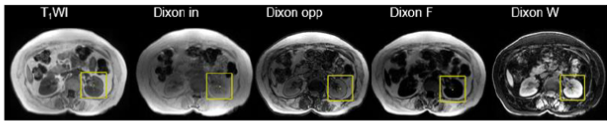

Fig. 1. Radiological Image of kidney with different imaging parameters

State Space The state is the sequence of areas within the image that are delineated by the bounding box. Our training samples have a frame history of length of 4, and the cropped image in the bounding box region with size of 45 by 45,

$$S = H \times W \times C, \quad H = W = 45, \quad C = 4$$

Action Space The action is to move the bounding box in one direction. For 2D, we can move it up, down, right or left. The action space is thus a tuple of position in four direction.

$$A = (x+, x-, y+, y-)$$

Reward The reward is the change in Euclidean distance from the center of the bounding box to the landmark. It's positive if the box is moving closer to the landmark and negative if away. As shown in figure 2, the red box is the target bounding box and the yellow agent bounding box is moving towards it.

Agent The task is performed by a single agent with deep Q learning.

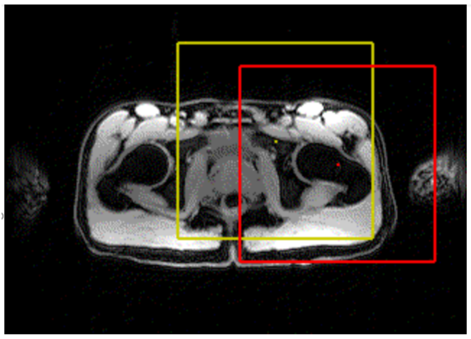

Fig. 2. Single-agent DQN locating landmark of left trochanter in 2D slices

B. *Deep Q Learning*

Recall that the Q-learning algorithm, a variant of TD learning, attempts to learn a policy for maximizing the expected total future reward, deep Q-leaning achieve the same goal utilizing powerful neural network.

$$Q(s, a) = \max_{\pi} E\left[r_t + \gamma r_{t+1} + \ldots \mid s_t = s, a_t = a, \pi\right]$$

Frequently applied on large and continuous state spaces, deep RL uses neural nets, such as convolutional neural networks (CNN) and recurrent Neural networks (RNN), to identify the current state and predict the best possible action. By approximating the optimal state-action value function, the deep RL algorithms can tackle problems that seems to be impossible to learn from scratch (i.e play video games). The application of deep RL algorithms in landmark localization is to learn to locate different anatomical landmarks with high accuracy, computational efficiency, and robustness.

C. *Experience Replay*

Experience replay is originally proposed to make more efficient use of observed experiences.

a) Waste of Experience: Consider the basic online Q-learning algorithm where we interact with the environment and at each time step, we obtain a state action reward next state tuple. We learn from it and then discarded, moving on to the next tuple in the following time step. That is wasteful as We could possibly learn more from these experienced tuples if we stored them somewhere. Moreover, some states are rare to come by and some actions can by costly. We expect the training to converge faster with higher accurate if rare experience are recalled. As a result, we store each experienced tuple in so-called replay buffer as we are interacting with the environment. From the buffer, we sample a small batch of tuples in order to learn faster and better. As we learn from individual tuples multiple times and recall rare occurrences, we optimize the deep RL by making better use for experience.

b) Correlated Updates: Another problem is the correlated updates where the next state is highly correlated with the action taken. This correlation is catastrophic for it could cause network weights to oscillate or to diverge catastrophically, which results in an unstable and ineffective policy. To summarize, experience replay build a database of samples and then learning a mapping from them. In some sense, it's turning the reinforcement learning problem to a supervised learning problem. We can even improve the experience replay by prioritizing experience tuples that are rare or more important.

c) Importance Sampling: One commonly applied techniques to improve experience replay is to sample from importance instead of random. The lower frequency events which is the rare experiences should be valued higher. This can be achieved by giving high priority to experiences with high TD error δ_{TD}. In addition, the learning rate should be small for the high TD error experience to cancel the bias from large sampling. However, this method fails to solve the potential memory problems caused by large, high-dimensional state space.

D. Count Sketch

To tackle memory-related problem, one of the best algorithm is the count sketch algorithm. We map this experience replay procedure to a streaming heavy-hitters problem. In simple words, instead of storing all experiences, only a sketch of the experiences is stored. The count sketch data structure is typically composed of four major operations: initialization, update, estimate and merge. [1].

```
1:  function init($R$, $C$):
2:      init $R \times C$ table of counters $S$ (with zeros)
3:      init bucket hashes: $\{h_1^r : [n] \to [C]\}_{r=1}^R$
4:      init sign hashes: $\{h_2^r : [n] \to \pm 1\}_{r=1}^R$
5:  function update($s_i$)):
6:      for $r$ in $1 \dots R$ :    $S[r, h_1^r(s_i)]$ += $h_2^r(s_i)$
7:  function estimate($i$):
8:      init length $R$ array estimates
9:      for $r$ in $1 \dots R$:   estimates$[r] = h_2^r(i) S[r, h_1^r(i)]$
10:     return median(estimates)
11: function merge($S_1, S_2$):
12:     return $S_1 + S_2$
```

The size of the sketch matrix is chosen to optimize the number of hash collisions. The procedure is repeated for every item. Once finished, we obtain a hash table which is the low dimensional approximate representation of the frequency histogram. The heavy hitters, the cells with high cardinality will have a high count in each of their corresponding buckets (one in each row). However, the counts will be different, because of hash collisions. By taking the median of the counts over the rows we can get a good estimate of the true cardinality, and we can extract the heavy items. Intuitively, buckets with high absolute value correspond to frequent items. And several rows eliminates the false-positives. If we chose the matrix size correctly, then the median will well represent the actual count.

In short, count sketch is a low dimensional embedding which is linear in time, and approximate item frequencies in logarithmic storage space. Using this approach, sketches of data at different locations can be computed in place, and only the hash table move to the final aggregation site. This not only saves huge amounts of data movement, but also protect data privacy as the hashing is not invertible.

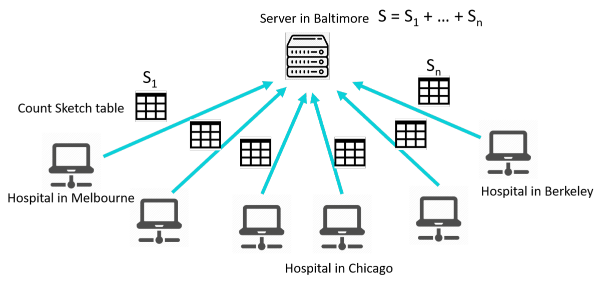

Fig. 3. Count Sketch uses sublinear memory; can be aggregated; preserves privacy

III. Implementation

In this section, we evaluate the performance of our trained agents. All algorithms are implemented in Python 3.8.3 and are carried out on Intel(R) Xeon(R) Gold 6226 CPU and Tesla V100 16GB GPU.

A. Multitask Modality Invariant Deep RL Framework

In this section, we reproduce results from the this paper [4]. The 2D single-agent DQN in paper is trained on 57 MRI data. One of the breakthrough in their paper is that they manage to train a model that is adaptable and generalizable to multiple different tasks. Before them, the applications of deep RL in landmark localization have been restricted to a single task. Consequently, training multiple deep RL agents across different body regions and imaging parameters would not only time and memory demanding, but would also be challenging to translate in a clinical workflow.

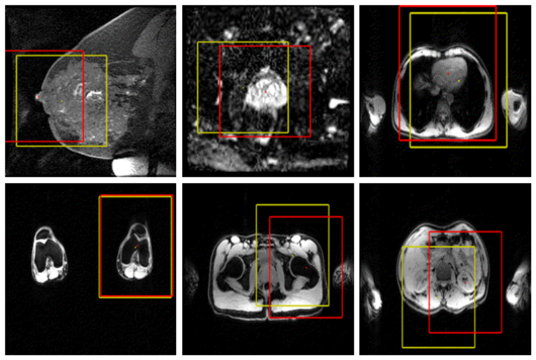

Fig. 4. Examples of a single 2D agent locating different landmarks in 2D slices.

For example shown in Figure 4, the single agent can locate different landmarks in 2D slices for breast nipple, prostate, left kidney, left tro-khanter, left knee cap, and heart with negligible error and reasonable convergence time. Traditional method would have to train 6 different tasks using supervised training by an expert. As a result, in the development of a medical decision support system, whether an agent can be adaptive and generalized to multi-task/organ plays a crucial role. We explain the training in details here.

a) Convolutional Neural Network (CNN): A convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. In Figure 5, we show the CNN architecture used in our deep RL framework. The medical images are first processed by four 2D convolutional layers followed by maxpooling and ReLU activation. This allows the system to exploit spatial relationships. In addition, since four sequential channels are stacked and provided as input, these convolutional layers also extract some temporal properties across those channels. After that, four fully connected layers with leaky ReLU are implemented. THe last fully-connected linear layer outputs a vector of action values. The loss function used is the huber loss

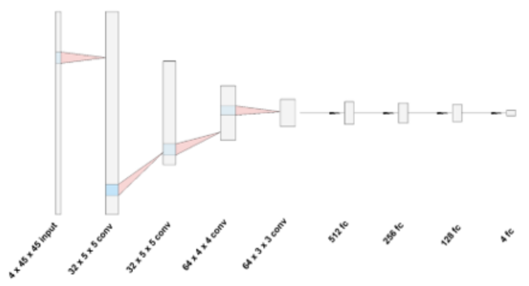

Fig. 5. CNN architecture in 2D single-agent DQN framework

which is less sensitive to outliers in data than the mean square error. After computing the loss function, Policy network's weights are updated on each step using stochastic gradient descent. Techniques like gradient clipping and learning rate scheduler are adopted as well.

b) Deep Q-Network (DQN): The training process uses a target network (DQN-T), and a policy network (DQN-P), both with the same CNN architecture. Initializing both networks with same weights, we start the training by feeding DQNP the first state s. Once obtain the the predicted Q-value of state s and action a, we then the next state s' to DQNT and obtain the Q-value of next state s' and next action a'. The target Q-value is $r + \gamma Q\left(s', a'\right)$, the current reward plus discounted future rewards. Thus, the losses propagate back are,

$$Loss = Huber\left(r + \gamma Q\left(s', a'\right) - Q(s, a)\right)$$

The actions are chosen under the ϵ–greedy policy with decreasing ϵ. Finally Policy network's weights are updated to target network's weights only every 2500 steps. This is done to stabilize training.

c) Hyper-parameter Tuning: We pick the batch size to be 48, learning rate to be 10^{-4}. The discounted parameter gamma is set to be 0.9. The optimizer we used is Adam. Techniques like gradient clipping and learning rate scheduler are also implemented. The buffer capacity is set to 10^6.

B. Sketch and Scale Pipeline

Training such DQN agent requires lots of data. Acquiring, sharing and storing such big data becomes problematic in practice. In this paper [6], they introduce a novel framework: Sketch and Scale (SnS) to tackle those challenges altogether. The idea is to leverage a Count Sketch data structure to compress the data on the edge nodes, aggregates the reduced size sketches on the master node. With small sacrifice in accuracy, frequency of items can be queried in a time and memory efficient way. Adopting this pipeline, We can extract a lower dimensional embedding of the state space that is highly representative. Furthermore, this pipeline can be applied to sketch the experiences (S, A, R, S') - tuple in the replay buffer. The frequency of each tuple can be quickly queried thus making importance sampling feasible in a memory shortage setting. We verify these assumptions on cancer immunotherapy data with 54 million pixels in 35-bands from multiplex images of tumor biopsies. We expect the pipeline to output a highly representative subset of the original data.

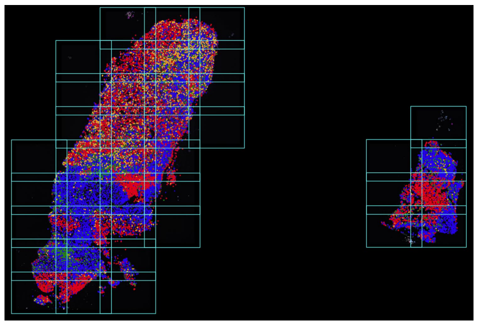

Fig. 6.

a) Cancer Immunotherapy: The cancer biopsy shown in Figure 6 contains tumor cells, marked with red color, surrounded by non-tumor cells. We cut it into 40 images, each image contains 1.5M pixels and 35 layers. We expect the data to have a lower embedding with approximately 8 degrees of freedom, representing 8 cell types. Our goal is to see what level of clustering can be detected at the pixel level, and whether clustering can be used to create an embedding with a clear separation between tumor and other cell types.

b) Sketching Results: We run the Count Sketch algorithm and create an ordered list of the top 20,000 heavy hitters. The top HH has 204,901 points, while the $20{,}000^{th}$ rank has only 180. The cumulative fraction of the top 20,000 heavy hitters is 84.11%. Meaning that the heavy hitters are forming a highly representative sample. The sketch matrix is 16x200,000. We then feed the top 20K HH to UMAP, and generate the top two coordinates. We find 10 clusters in the data. Adjacent cluster have been aggregated into three groups (tumor, immune and other), shown in different colors of Figure 7. These show an excellent agreement with labels generated for nuclei using an industry standard segmentation software. To sum, we successfully embed the originally state space with 35 dimensions to a binary space: Tumor-1; Other-0.

c) Algorithm Performance: The scaling performance follows the expected linear trend, once the upfront costs of setting up the data structures become small compared to the stream processing. The cost for 1B data points is less

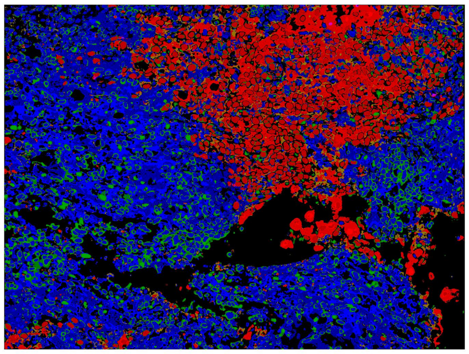

Fig. 7. The embedded state space with only two dimensions: tumor (red), non-tumor (immune(green) and others(blue)).

than 2 minutes, clearly demonstrating the advantages of the streaming count-sketch. Since only the sketch table is stored, the algorithm is sublinear in space. From this quick sanity check, we conclude that Count Sketch processor can find low-dimensional embedding of state-action space in a time and memory efficient way.

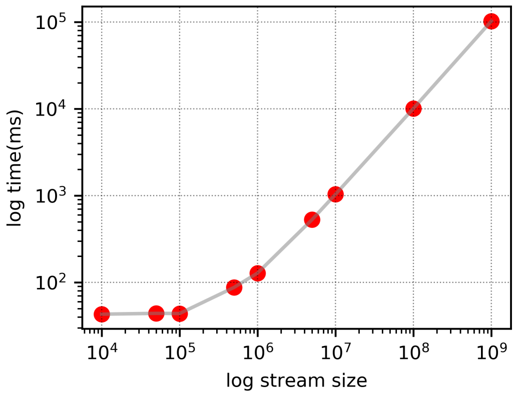

Fig. 8. The scaling of the run time for sketch table of size 10x20,000 vs. the size of the data, up to a billion points. For the higher cardinalities the scaling is very linear.

IV. Evaluation

In this section, we combine the deep RL framework developed in [4] and the Sketch & Scale pipeline in [6]. We evaluate this distributed deep RL framework on a large-sized dataset from cancer immunotherapy where each tumor biopsy containing 52M pixel and 35 layers. Currently we have more than 45,000 biopsies created, with 100,000 more in the queue, resulting in hundreds of billions of total pixels. Approaching a pixel-wide analysis of such a data will only be possible through highly scalable algorithms.

Furthermore, the training can only be possible if we maintain a simpler, yet highly representative embedding of the original high dimensional state space. That is instead of feeding the deep RL framework with a state specified with 35 parameters, we use Sketch & Scale pipeline to derive a binary representation with 1 indicating tumor and 0 otherwise.

The computations for such embedding were extremely fast, essentially I/O limited. Processing the sketch scale of billions points takes only dozen minutes on a single V100 GPU, using a single stream for the I/O. We can do much better by parallelizing the I/O to saturate compute power of all the CUDA kernels in the GPU.

We then feed four sequential slices of the tumor biopsies in binary representation to the deep RL framework for landmark localization. The experience buffer is optimized as following. We first quantize the state, action and reward space as shown in Figure 9. Second, as the experience tuple arriving one by one, we represent each tuple with its bin ID in the quantized space.

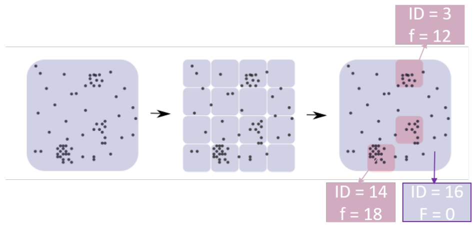

Fig. 9. Quantization of state-action-reward space to optimize experience replay

Due to memory constraint, it is infeasible to maintain a counter for each item, thus we use the memory-efficient count sketch algorithm. By definition, the count sketch can compute a $(1 \pm \epsilon)$-approximation of frequency F_k.

$$P\left(|X - F_k| \geqslant \epsilon F_k\right) < \delta \text{ i.e. } X \sim (1 \pm \epsilon) F_k$$

We keep only unique items in the replay buffer and sample from it based on the approximated frequency. Furthermore, these sketches of experience replay sorted from low to high frequency at separate hospitals can be aggregated together and sampled by frequency to survive overfitting. This not only saves huge amounts of data movement, but also diminishes potential data privacy concerns, as the approximate hashing is not invertible, i.e. hides all identifiable information. The only constraint is that the hashing functions and the sketch matrix sizes must be the same for all threads. To conclude, the 2D DQN agent manages to locate the landmark in reasonable time with minor errors.

V. Conclusion

We present a distributed deep RL framework that is aimed at tackling existing challenges of an AI-powered medical decision support system. We demonstrate how deep RL framework equipped with a CNN that takes in a time sequence of the cropped image in the bounding box and outputs Q-value for each possible action can achieve the generalization of landmark localization tasks. We verify that the Sketch & Scale pipeline can reduce the cardinality of extreme sized data sets with moderate dimensions while preserving the clusterting properties, thus rendering a highly representative and simpler state space. Moreover, we show that this embedding pipeline scales linear in time and sublinear in memory. Especially, we show that the frequency list and the heavy hitters of billion-ordered data set can be extracted in two minutes. As the accuracy of most landmark localization reinforcement learning algorithms depends on the their capability to store and recall experience tuples, we hypothesized how this frequency sketching techniques can be utilized to improve the experience replay. On one hand, the algorithm might achieve faster convergence if experience tuples are sampled based on how rare they occurred (low frequency tuples). On the other hand, Sketches of experience replay can be computed on geo-distributed data sets across the globe, and later be aggregated together in a secure and affordable way. As the sketches are irreversible, it overcomes the problem rising from institutional and national policies limiting the free movement of the data across boundaries and between research centers and hospitals.

We demonstrated the utility of this approach on 50M pixels of cancer images. We found that the high cardinality and high dimensional space states after Sketch & Scale pipeline can be embedded into a binary space. Moreover, the labeling results were in excellent agreement with the ground truth. The entire process takes only a few seconds. We then feed four sequential slices of the embedded data into the deep RL framework for landmark localization. The agent can find the landmarks in time with negligible errors. To conclude, we combined the two techniques that address different challenges in the field of medical imaging. We demonstrate the potential of the distributed deep RL framework of landmark localization. We expect this sketch-augmented deep RL framework to lay the foundation for an AI-powered medical decision support system.

References

[1] Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. In *International Colloquium on Automata, Languages, and Programming* (2002), Springer, pp. 693–703.

[2] Maaten, L. v. d., and Hinton, G. Visualizing data using t-sne. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[3] McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[4] Parekh, V., Bocchieri, A., Braverman, V., and Jacobs, M. Multitask radiological modality invariant landmark localization using deep reinforcement learning. In *Proceedings of the Third Conference on Medical Imaging with Deep Learning* (Montreal, QC, Canada, 06–08 Jul 2020), vol. 121 of *Proceedings of Machine Learning Research*, PMLR, pp. 588–600.

[5] Shokri, R., and Shmatikov, V. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security* (2015), pp. 1310–1321.

[6] Wei, V., Ivkin, N., Braverman, V., and Szalay, A. Sketch and Scale: Geo-distributed tSNE and UMAP. *arXiv e-prints* (Nov. 2020), arXiv:2011.06103.