Adaptive partial scanning transmission electron microscopy with reinforcement learning

IOP Publishing Mach. Learn.: Sci. Technol. 2 (2021) 045011 https://doi.org/10.1088/2632-2153/abf5b6

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
22 December 2020

REVISED
1 March 2021

ACCEPTED FOR PUBLICATION
7 April 2021

PUBLISHED
19 July 2021

PAPER

Adaptive partial scanning transmission electron microscopy with reinforcement learning

Jeffrey M Ede

Department of Physics, University of Warwick, Coventry CV4 7AL, United Kingdom

E-mail: j.m.ede@warwick.ac.uk

Keywords: adaptive scans, compressed sensing, deep learning, electron microscopy, reinforcement learning

Supplementary material for this article is available online

Abstract

Compressed sensing can decrease scanning transmission electron microscopy electron dose and scan time with minimal information loss. Traditionally, sparse scans used in compressed sensing sample a static set of probing locations. However, dynamic scans that adapt to specimens are expected to be able to match or surpass the performance of static scans as static scans are a subset of possible dynamic scans. Thus, we present a prototype for a contiguous sparse scan system that piecewise adapts scan paths to specimens as they are scanned. Sampling directions for scan segments are chosen by a recurrent neural network (RNN) based on previously observed scan segments. The RNN is trained by reinforcement learning to cooperate with a feedforward convolutional neural network that completes the sparse scans. This paper presents our learning policy, experiments, and example partial scans, and discusses future research directions. Source code, pretrained models, and training data is openly accessible at https://github.com/Jeffrey-Ede/adaptive-scans.

1. Introduction

Most scan systems sample signals at sequences of discrete probing locations. Examples include atomic force microscopy [1, 2], computerized axial tomography [3, 4], electron backscatter diffraction [5], scanning electron microscopy [6], scanning Raman spectroscopy [7], scanning transmission electron microscopy (STEM) [8] and x-ray diffraction spectroscopy [9]. In STEM, the high current density of electron probes produces radiation damage in many materials, limiting the range and types of investigations that can be performed [10, 11]. In addition, most STEM signals are oversampled [12] to ease visual inspection and decrease sub-Nyquist artefacts [13]. As a result, a variety of compressed sensing [14] algorithms have been developed to enable decreased STEM probing [15]. In this paper, we introduce a new approach to STEM compressed sensing where a scan system is trained to piecewise adapt partial scans [16] to specimens by deep reinforcement learning (RL) [17].

Established compressed sensing strategies include random sampling [18–20], uniformly spaced sampling [19, 21–23], sampling based on a model of a sample [24, 25], partials scans with fixed paths [16], dynamic sampling to minimize entropy [26–29] and dynamic sampling based on supervised learning [30]. Complete signals can be extrapolated from partial scans by an infilling algorithm, estimating their fast Fourier transforms [31] or inferred by an artificial neural network (ANN) [16, 23]. In general, the best sampling strategy varies for different specimens. For example, uniformly spaced sampling is often better than spiral paths for oversampled STEM images [16]. However, sampling strategies designed by humans usually have limited ability to leverage an understanding of physics to optimize sampling. As proposed by our earlier work [16], we have therefore developed ANNs to dynamically adapt scan paths to specimens. Expected performance of dynamic scans can always match or surpass expected performance of static scans as static scan paths are a special case of dynamic scan paths and performance varies for different static scan paths [16].

Exploration of STEM specimens is a finite-horizon partially observed Markov decision process (POMDP) [32, 33] with sparse losses: a partial scan can be constructed from path segments sampled at each step of the POMDP and a loss can be based on the quality of an scan completion generated from the partial scan with an ANN. Most scan systems support custom scan paths or can be augmented with a field programmable gate array (FPGA) [34, 35] to support custom scan paths. However, there is a delay before a scan system can execute or is ready to receive a new command. Total latency can be reduced by using both fewer and larger steps, and decreasing steps may also reduce distortions due to cumulative errors in probing positions [34] after commands are executed. Command execution can also be delayed by ANN inference. However, inference delay can be minimized by using a computationally lightweight ANN and inferring future commands while previous commands are executing.

Markov decision processes (MDPs) can be optimized by recurrent neural networks (RNNs) based on long short-term memory (LSTM) [36, 37], gated recurrent unit (GRU) [38], or other cells [39–41]. LSTMs and GRUs are popular as they solve the vanishing gradient problem [42] and have consistently high performance [40]. Small RNNs are computationally inexpensive and are often applied to MDPs as they can learn to extract and remember state information to inform future decisions. To solve dynamic graphs, an RNN can be augmented with dynamic external memory to create a differentiable neural computer (DNC) [43]. To optimize a MDP, a discounted future loss, Q_t, at step t in a MDP with T steps can be calculated from step losses, L_t, with Bellman's equation,

$$Q_t = \sum_{t'=t}^{T} \gamma^{t'-t} L_{t'}\,, \tag{1}$$

where $\gamma \in [0, 1)$ discounts future step losses. Equations for RL are often presented in terms of rewards, e.g. $r_t = -L_t$; however, losses are an equivalent representation that avoids complicating our equations with minus signs. Discounted future loss backpropagation through time (BPTT) [44] enables RNNs to be trained by gradient descent [45]. However, losses for partial scan completions are not differentiable with respect to (w.r.t.) RNN actions, $(a_0, \ldots, a_{T-1})$, controlling which path segments are sampled.

Many MDPs have losses that are not differentiable w.r.t. agent actions. Examples include agents directing their vision [46, 47], managing resources [48], and playing score-based computer games [49, 50]. Actors can be trained with non-differentiable losses by introducing a differentiable surrogate [51] or critic [52] to predict losses that can be backpropagated to actor parameters. Alternatively, non-differentiable losses can be backpropagated to agent parameters if actions are sampled from a differentiable probability distribution [46, 53] as training losses given by products of losses and sampling probabilities are differentiable. There are also a variety of alternatives to gradient descent, such as simulated annealing [54] and evolutionary algorithms [55], that do not require differentiable loss functions. Such alternatives can outperform gradient descent [56]; however, they usually achieve similar or lower performance than gradient descent for deep ANN training.

2. Training

In this section, we outline our training environment, ANN architecture and learning policy. Our ANNs were developed in Python with TensorFlow [57]. Detailed architecture and learning policy is in supplementary information (available online at stacks.iop.org/MLST/2/045011/mmedia). In addition, source code and pretrained models are openly accessible from GitHub [58], and training data is openly accessible [12, 59].

2.1. Environment

To create partial scans from STEM images, an actor, μ, infers action unit vectors, $\mu(h_t)$, based on a history, $h_t = (a_0, o_1^i, \ldots, a_{t-1}, o_t)$, of previous actions, a, and observations, o. To encourage exploration, $\mu(h_t)$ is rotated to a_t by Ornstein–Uhlenbeck (OU) [60] exploration noise [61], ε_t,

$$a_t = \begin{bmatrix} \cos\varepsilon_t & -\sin\varepsilon_t \\ \sin\varepsilon_t & \cos\varepsilon_t \end{bmatrix} \mu(h_t) \tag{2}$$

$$\epsilon_t = \theta(\varepsilon_{\text{avg}} - \varepsilon_{t-1}) + \sigma W \tag{3}$$

where we chose $\theta = 0.1$ to decay noise to $\varepsilon_{\text{avg}} = 0$, a scale factor, $\sigma = 0.2$, to scale a standard normal variate, W, and start noise $\varepsilon_0 = 0$. OU noise is linearly decayed to zero throughout training. Correlated OU exploration noise is recommended for continuous control tasks optimized by deep deterministic policy gradients (DDPGs) [49] and recurrent deterministic policy gradients (RDPGs) [50]. Nevertheless, follow-up

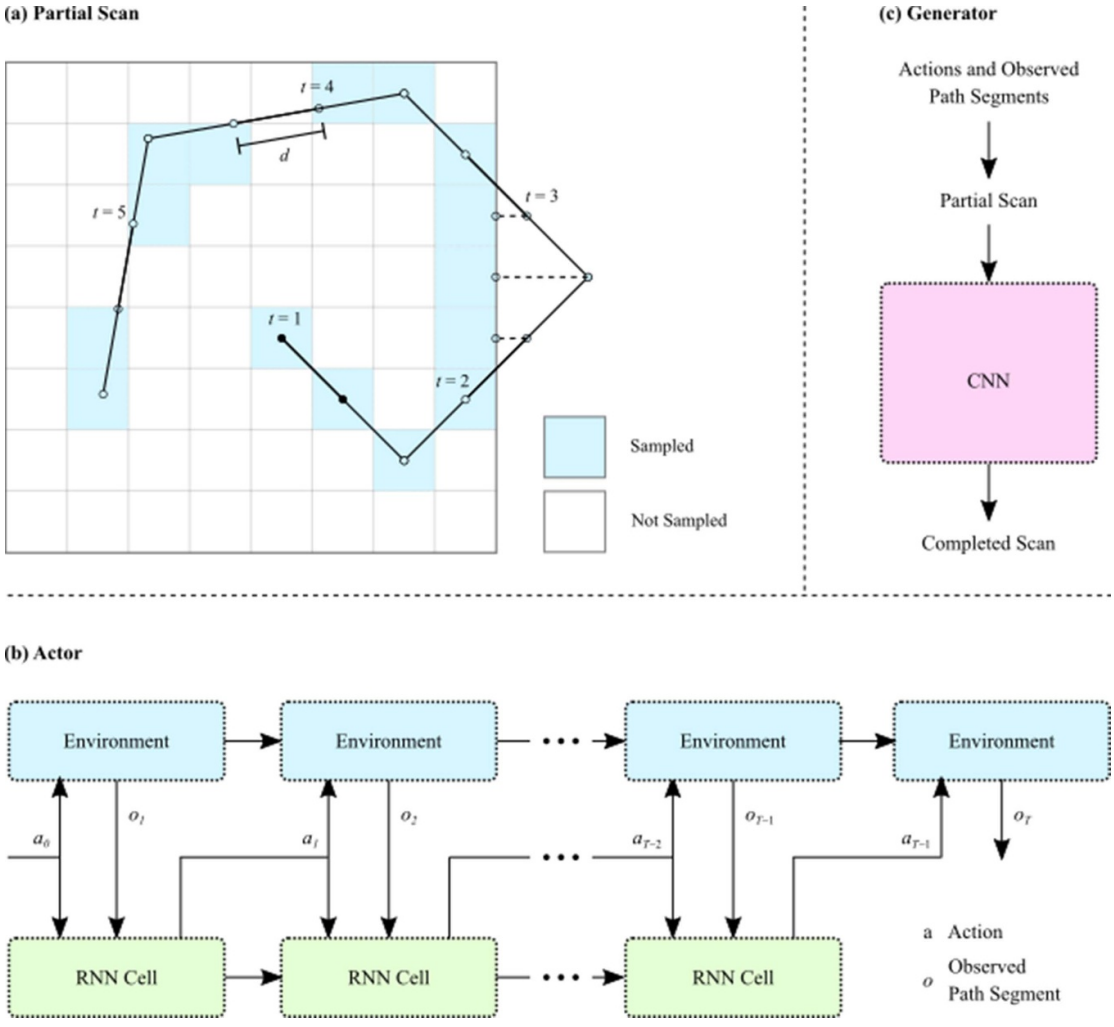

Figure 1. Simplified scan system. (a) An example 8 × 8 partial scan with $T = 5$ straight path segments. Each segment in this example has three probing positions separated by $d = 2^{1/2}$ px, and their starts are labelled by step numbers, t. Partial scans are selected from STEM images by sampling image pixels nearest probing positions, even if a nominal probing position is outside an imaging region. (b) An actor RNN uses its previous state, action, and an observed path segment to choose the next action at each step. (c) A partial scan constructed from actions and observed path segments is completed by a generator CNN.

experiments with twin delayed deep deterministic policy gradients (TD3) [62] and distributed distributional deep deterministic policy gradients (D4PG) [63] have found that uncorrelated Gaussian noise can produce similar results.

An action, a_t, is the direction to move to observe a path segment, o_t, from the position at the end of the previous path segment. Partial scans are constructed from complete histories of actions and observations, h_T. A simplified partial scan is shown in figure 1(a). In our experiments, partial scans, s, are constructed from $T = 20$ straight path segments selected from 96 × 96 STEM images. Each segment has 20 probing positions separated by $d = 2^{1/2}$ px and positions can be outside an image. The pixels in the image nearest each probing position are sampled, so a separation of $d \geq 2^{1/2}$ simplified development by preventing successive probing positions in a segment from sampling the same pixel. A separation of $d < 2^{1/2}$ would allow a pixel to sampled more than once by moving diagonally, potentially incentivising orthogonal scan motion to sample more pixels.

Following our earlier work [16, 23, 64], we select subsets of pixels from STEM images to create partial scans to train ANNs for compressed sensing. Selecting a subset of pixels is easier than preparing a large, carefully partitioned and representative dataset [65, 66] containing experimental partial scan and full image pairs, and selected pixels have realistic noise characteristics as they are from experimental images. However, selecting a subset of pixels does not account for probing location errors varying with scan shape [34]. We use a Warwick Electron Microscopy Dataset containing 19 769 32-bit 96 × 96 images cropped and downsampled from full images [12, 59]. Cropped images were blurred by a symmetric 5 × 5 Gaussian kernel with a 2.5 px standard deviation to decrease any training loss variation due to varying noise characteristics. Finally, images, I, were linearly transformed to normalized images, I_N, with minimum and maximum values of −1

and 1. To test performance, the 19 769 images were split, without shuffling, into a training set containing 15 815 images and a test set containing 3954 images.

2.2. Architecture

For training, our adaptive scan system consists of an actor, μ, target actor, μ', critic, Q, target critic, Q', and generator, G. Simplified actor and generator architecture is shown in figures 1(b) and (c). To minimize latency, our actors and critics are computationally inexpensive deep LSTMs [67] with a depth of 2 and 256 hidden units. Our generator is a convolutional neural network (CNN) [68, 69]. A recurrent actor selects actions, a_t and observes path segments, o_t, that are added to an experience replay [70], R, containing 10^5 complete histories of actions and observations. Partial scans, s, are constructed from histories sampled from the replay to train a generator to complete partial scans, $I_G^i = G(s^i)$. The actor and generator cooperate to minimize generator losses, L_G, and are the only networks needed for inference.

Generator losses are not differentiable w.r.t. actor actions used to construct partial scans i.e. $\partial L_G/\partial a_t = 0$. Following RDPG [50], we therefore introduce recurrent critics to predict losses from actor actions and observations that can be backpropagated to actors for training by BPTT. Actor and critic RNNs have the same architecture, except actors have two outputs to parameterize actions whereas critics have one output to predict losses. Target networks [49, 71] use exponential moving averages of live actor and critic network parameters and are introduced to stabilize learning. For training by RDPG, live and target ANNs separately replay experiences. However, we propagate live RNN states to target RNNs at each step as a precaution against any cumulative divergence of target network behaviour from live network behaviour across multiple steps.

2.3. Learning policy

To train actors to cooperate with a generator to complete partial scans, we developed cooperative recurrent deterministic policy gradients (CRDPG, algorithm 1). This is an extension of RDPG to an actor that cooperates with another ANN to minimize its loss. We train our networks by ADAM [72] optimized gradient descent for $M = 10^6$ iterations with a batch size, $N = 32$. We use constant learning rates $\eta_\mu = 0.0005$ and $\eta_Q = 0.0010$ for the actor and critic, respectively. For the generator, we use an initial learning rate $\eta_G = 0.0030$ with an exponential decay factor of $0.75^{5m/M}$ at iteration m. The exponential decay envelope is multiplied by a sawtooth cyclic learning rate [73] with a period of $2M/9$ that oscillates between 0.2 and 1.0. Training takes two days with an Intel i7-6700 CPU and an Nvidia GTX 1080 Ti GPU.

We augment training data by a factor of eight by applying a random combination of flips and 90° rotations, mapping $s \to s'$ and $I_N \to I'_N$, similar to our earlier work [16, 23, 64, 74]. Our generator is trained to minimize mean squared errors,

$$L_G = \mathrm{MSE}(G(s'), I_N)\,, \tag{12}$$

between scan completions, $G(s')$, and normalized target images, I_N. Generator losses decrease during training as the generator learns, and may vary due to loss spikes [64], learning rate oscillations [73] or other training phenomena. Normalizing losses can improve RL [75], so we divide generator losses used for critic training by their running mean,

$$L_{\mathrm{avg}} \leftarrow \beta_L L_{\mathrm{avg}} + \frac{1-\beta_L}{N}\sum_i^N L_G\,, \tag{13}$$

where we chose $\beta_L = 0.997$ and L_{avg} is updated at each training iteration.

Heuristically, an optimal policy does not go over image edges as there is no information there in our training environment. To accelerate convergence, we therefore added a small loss penalty, $E_t = 0.1$, at step t if an action results in a probing position being over an image edge. The total loss at each step is

$$L_t = E_t + \delta_{tT}\frac{\mathrm{clip}(L_G)}{L_{\mathrm{avg}}}\,, \tag{14}$$

where $\mathrm{clip}(L_G)$ clips losses used for RL to three standard deviations above their running mean. This adaptive loss clipping is inspired by adaptive learning rate clipping [64] and reduces learning destabilization by high loss spikes. However, we expect that clipping normalized losses to a fixed threshold [71] would achieve similar results. The Kronecker delta, δ_{tT}, in equation (14) is 1 if $t = T$ and 0 otherwise, so it only adds the generator loss at the final step, T.

To estimate discounted future losses, Q_t^{rl}, for RL, we use a target actor and critic,

Algorithm 1. Cooperative recurrent deterministic policy gradients (CRDPG).

Initialize actor, μ, critic, Q, and generator, G, networks with parameters ω, θ and ϕ, respectively.
Initialize target networks, μ' and Q', with parameters $\omega' \leftarrow \omega$, $\theta' \leftarrow \theta$, respectively.
Initialize replay buffer, R.
Initialize average generator loss, L_{avg}.
for iteration $m = 1, M$ **do**
  Initialize empty history, h_0.
  for step $t = 1, T$ **do**
    Make observation, o_t.
    $h_t \leftarrow h_{t-1}, a_{t-1}, o_t$ (append action and corresponding observation to history).
    Select action, a_t, by computing $\mu(h_t)$ and applying exploration noise, ε_t.
  end for
  Store the sequence $(a_0, o_1, ..., a_{T-1}, o_T)$ in R.
  Sample a minibatch of N histories, $h_T^i = (a_0^i, o_1^i, ..., a_{T-1}^i, o_T^i)$, from R.
  Construct partial scans, s^i, from h_T^i.
  Use generator to complete partial scans, $I_G^i = G(s^i)$.
  Compute step losses, $(L_1^i, ..., L_T^i)$, from generator losses, L_G^i, and over edge losses, E_t^i,

$$L_t^i = E_t^i + \delta_{tT}\frac{\text{clip}(L_G^i)}{L_{\text{avg}}}, \tag{4}$$

  where the Kronecker delta, δ_{tT}, is 1 if $t = T$ and 0 otherwise, and $\text{clip}(L_G^i)$ is the smaller of L_G^i and three standard deviations above its running mean.
  Compute target values, $(y_1^i, ..., y_T^i)$, with target networks,

$$y_t^i = L_t^i + \gamma Q'(H_Q^i, a_t^i, o_{t+1}^i, \mu'(H_\mu^i, a_t^i, o_{t+1}^i)), \tag{5}$$

  where H_Q^i and H_μ^i are hidden states of live networks after computing $Q(h_t^i, a_t^i)$ and $\mu(h_t^i)$, respectively.
  Compute critic update (using BPTT),

$$\Delta\omega = \frac{1}{NT}\sum_i^N\sum_t^T (y_t^i - Q(h_t^i, a_t^i))\frac{\partial Q(h_t^i, a_t^i)}{\partial\omega}. \tag{6}$$

  Compute actor update (using BPTT),

$$\Delta\theta = \frac{1}{NT}\sum_i^N\sum_t^T \frac{\partial Q(h_t^i, a_t^i)}{\partial\mu(h_t^i)}\frac{\partial\mu(h_t^i)}{\partial\theta}. \tag{7}$$

  Compute generator update,

$$\Delta\phi = \frac{1}{N}\sum_i^N \frac{\partial L_G^i}{\partial\phi}. \tag{8}$$

  Update the actor, critic, and generator by gradient descent.
  Update the target networks and average generator loss,

$$\omega' \leftarrow \beta_\omega\omega' + (1-\beta_\omega)\omega, \tag{9}$$

$$\theta' \leftarrow \beta_\theta\theta' + (1-\beta_\theta)\theta, \tag{10}$$

$$L_{\text{avg}} \leftarrow \beta_L L_{\text{avg}} + \frac{1-\beta_L}{N}\sum_i^N (L_G^i). \tag{11}$$

end for

$$Q_t^{\text{rl}} = L_t + \gamma Q'(h_{t+1}, \mu'(h_{t+1})), \tag{15}$$

where we chose $\gamma = 0.97$. Target networks stabilize learning and decrease policy oscillations [76–78]. The critic is trained to minimize mean squared differences, L_Q, between predicted and target losses, and the actor is trained to minimize losses, L_μ, predicted by the critic,

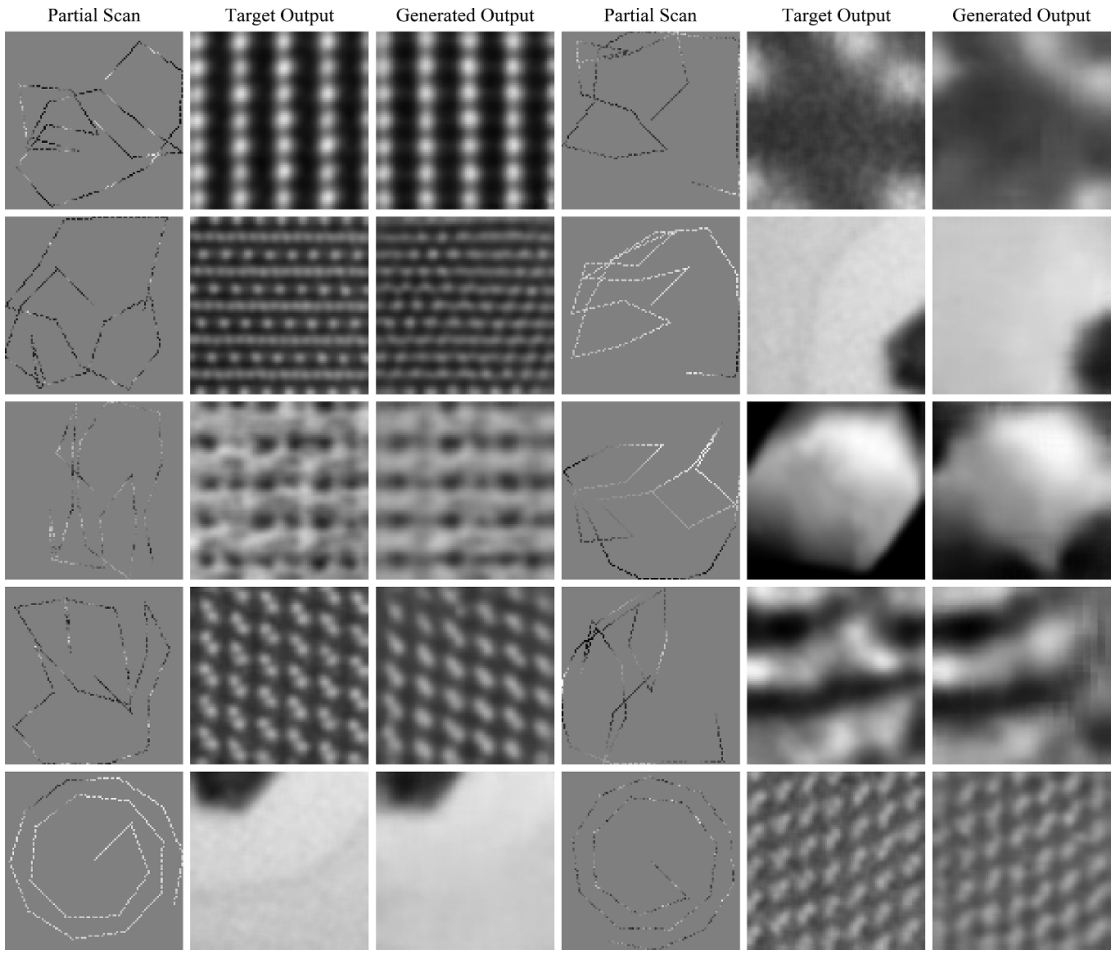

Figure 2. Examples of test set 1/23.04 px coverage partial scans, target outputs and generated partial scan completions for 96 × 96 crops from STEM images. The top four rows show adaptive scans, and the bottom row shows spiral scans. Input partial scans are noisy, whereas target outputs are blurred.

$$L_Q = \frac{1}{2T}\sum_{t=1}^{T}(y_t - Q(h_t, a_t))^2\,, \tag{16}$$

$$L_\mu = \frac{1}{T}\sum_{t=1}^{T}Q(h_t, a_t)\,. \tag{17}$$

Our target actor and critic have trainable parameters ω' and θ', respectively, that track live parameters, ω and θ, by soft updates [49],

$$\omega'_m = \beta_\omega \omega'_{m-1} + (1-\beta_\omega)\omega_m\,, \tag{18}$$

$$\theta'_m = \beta_\theta \theta'_{m-1} + (1-\beta_\theta)\theta_m\,, \tag{19}$$

where we chose $\beta_\omega = \beta_\theta = 0.9997$. We also investigated hard updates [71], where target networks are periodically copied from live networks; however, we found that soft updates result in faster convergence and more stable training.

3. Experiments

In this section, we present examples of adaptive partial scans and select learning curves for architecture and learning policy experiments. Examples of 1/23.04 px coverage partial scans, target outputs and generator completions are shown in figure 2 for 96 × 96 crops from test set STEM images. They show both adaptive and spiral scans after flips and rotations to augment data for the generator. The first actions select a path segment from the middle of image in the direction of a corner. Actors then use the first and following

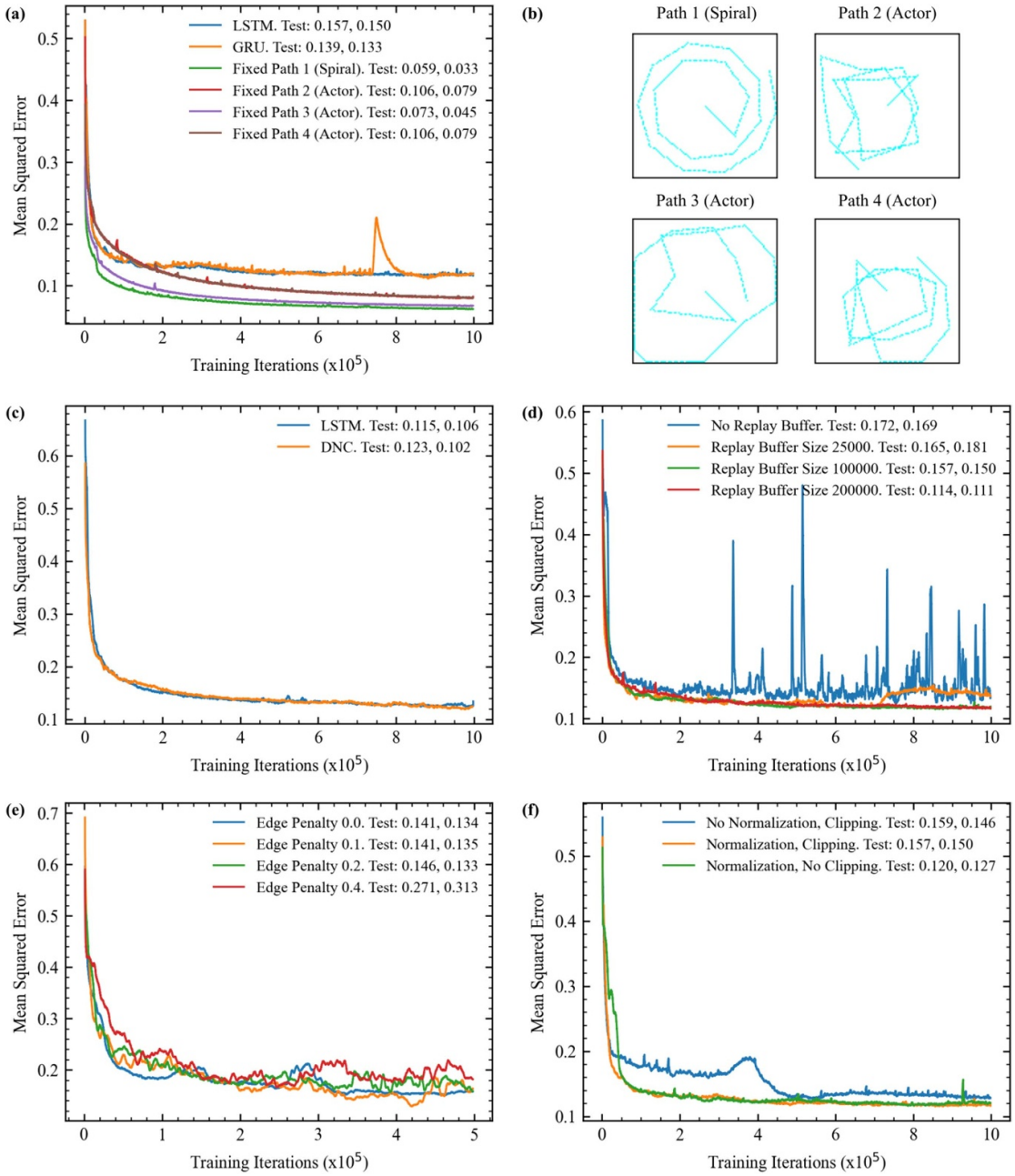

Figure 3. Learning curves for (a)–(b) adaptive scan paths chosen by an LSTM or GRU, and fixed spiral and other fixed paths, (c) adaptive paths chosen by an LSTM or DNC, (d) a range of replay buffer sizes, (e) a range of penalties for trying to sample at probing positions over image edges, and (f) with and without normalizing or clipping generator losses used for critic training. All learning curves are 2500 iteration boxcar averaged and results in different plots are not directly comparable due to varying experiment settings. Means and standard deviations of test set errors, 'Test: Mean, Std Dev', are at the ends of labels in graph legends.

observations to inform where to sample the remaining $T - 1 = 19$ path segments. Actors adapt scan paths to specimens. For example, if an image contains regular atoms, an actor might cover a large area to see if there is a region where that changes. Alternatively, if an image contains a uniform region, actors, may explore near image edges and far away from the uniform region to find region boundaries.

The main limitation of our experiments is that generators trained to complete a variety of partial scan paths generated by an actor achieves lower performance than a generate trained to complete partial scans with a fixed path. For example, figure 3(a) shows that generators trained to cooperate with LSTM or GRU actors are outperformed by generators trained with fixed spiral or other scan paths shown in figure 3(b). Spiral paths outperform fixed scan paths; however, we emphasize that paths generated by actors are designed for individual training data, rather than all training data. Freezing actor training to prevent changes in actor policy does not result in clear improvements in generator performance. Consequently, we think that improvements to generator architecture or learning policy should be a starting point for further

investigation. To find the best practical actor policy, we think that a generator trained for a variety of scan paths should achieve comparable performance to generators trained for single scan paths.

We investigated a variety of popular RNN architectures to minimize inference time. Learning curves in figure 3(a) show that performance is similar for LSTMs and GRUs. GRUs require less computation. However, LSTM and GRU inference time is comparable and GRU training seems to be more prone to loss spikes, so LSTMs may be preferable. We also created a DNC by augmenting a deep LSTM with dynamic external memory. However, figure 3(c) shows that LSTM and DNC performance is similar, and inference time and computational requirements are much higher for our DNC. We tried to reduce computation and accelerate convergence by applying projection layers to LSTM hidden states [79]. However, we found that performance decreased with decreasing projection layer size.

Experienced replay buffers for RL often have heuristic sizes, such as 10^6 examples. However, RL can be sensitive to replay buffer size [70]. Indeed, learning curves in figure 3(d) show that increasing buffer size improves learning stability and decreases test set errors. Increasing buffer size usually improves learning stability and decreases forgetting by exposing actors and critics to a higher variety of past policies. However, we expect that convergence would be slowed if the buffer became too large as increasing buffer size increases expected time before experiences with new policies are replayed. We also found that increasing buffer sized decreased the size of small loss oscillations [76–78], which have a period near 2000 iterations. However, the size of loss oscillations does not appear to affect performance.

We found that initial convergence is usually delayed if a large portion of initial actions go outside the imaging region. This would often delay convergence by about 10^4 iterations before OU noise led to the discovery of better exploration strategies away from image edges. Although 10^4 iterations is only 1% of our 10^6 iteration learning policy, it often impaired development by delaying debugging or evaluation of changes to architecture and learning policy. Augmenting RL losses with subgoal-based heuristic rewards can accelerate convergence by making problems more tractable [80]. Thus, we added loss penalties if actors tried to go over image edges, which accelerated initial convergence. Learning curves in figure 3(e) show that over edge penalties at each step smaller than $E_t = 0.2$ have a similar effect on performance. Further, performance is lower for higher over edge penalties, $E_t \geq 0.2$. We also found that training is more stable if over edge penalties are added at individual steps, rather than propagated to past steps as part of a discounted future loss.

Our actor, critic and generator are trained together. It follows that generator losses, which our critic learns to predict, decrease throughout training as generator performance improves. However, normalizing loss sizes usually improves RL [75], so we divide by their running means in equation (14). Learning curves in figure 3(f) show that loss normalization improves learning stability and decreases final errors. Clipping training losses can improve RL [71], so we clipped generator losses used for critic training to three standard deviations above their running means. We found that clipping increases test set errors, possibly because most training errors are in a similar regime. Thus, we expect that clipping may be more helpful for training with sparser scans as higher uncertainty may increase likelihood of unusually high generator losses.

4. Discussion

The main limitation of our adaptive scan system is that generator errors are much higher when a generator is trained for a variety of scan paths than when it is trained for a single scan path. However, we expect that generator performance for a variety of scans could be improved to match performance for single scans by developing a larger neural network with a better learning policy. To train actors to cooperate with generators, we developed CRDPG. This is an extension of RDPG [50], and RDPG is based on DDPG [49]. Alternatives to DDPG, such as TD3 [62] and D4PG [63], arguably achieve higher performance, so we expect that they could form the basis of a future training algorithm.Further, we expect that architecture and learning policy could be improved by AdaNet [81], Ludwig [82], or other automatic machine learning (AutoML) [83–87] algorithms as AutoML can often match or surpass the performance of human developers [88, 89]. Finally, test set losses for a variety of scans appear to be decreasing at the end of training, so we expect that performance could be improved by increasing training iterations.

After generator performance is improved, we expect the main limitation of our adaptive scan system to be distortions caused by probing position errors. Errors usually depend on scan path shape [34] and accumulate for each path segment. Non-linear scan distortions can be corrected by comparing pairs of orthogonal raster scans [90, 91], and we expect this method can be extended to partial scans. However, orthogonal scanning would complicate measurement by limiting scan paths to two half scans to avoid doubling electron dose on beam-sensitive materials. Instead, we propose that a cyclic generator [92] could be trained to correct scan distortions and provide a detailed method as supplementary information [93]. Another limitation is that our generators do not learn to correct STEM noise [94]. However, we expect that generators can learn to remove noise, for example, from single noisy examples [95] or by supervised learning [74].

To simplify our preliminary investigation, our scan system samples straight path segments and cannot go outside a specified imaging region. However, actors could learn to output actions with additional degrees of freedom to describe curves, multiple successive path segments, or sequences of non-contiguous probing positions. Similarly, additional restrictions could be applied to actions. For example, actions could be restricted to avoid actions that cause high probing position errors. Training environments could also be modified to allow actors to sample pixels over image edges by loading images larger than partial scan regions. In practice, actors can sample outside a scan region and being able to access extra information outside an imaging region could improve performance. However, using larger images may slow development by increasing data loading and processing times.

Not all scan systems support non-raster scan paths. However, many scan controllers can be augmented with an FPGA to enable custom scan paths [34, 35]. Recent versions of Gatan DigitalMicrograph support Python [96], so our ANNs can be readily integrated into existing scan systems. Alternatively, an actor could be synthesized on a scan-controlling FPGA [97, 98] to minimize inference time. There could be hundreds of path segments in a partial scan, so computationally lightweight and parallelizable actors are essential to minimize scan time. We have therefore developed actors based computationally inexpensive RNNs, which can remember state information to inform future decisions. Another approach is to update a partial scan at each step to be input to feedforward neural network (FNN), such as a CNN, to decide actions. However, we expect that FNNs are less practical than RNNs as FNNs may require additional computation to reprocess all past states at each step.

5. Conclusions

Our initial investigation demonstrates that actor RNNs can be trained by RL to direct piecewise adaption of contiguous scans to specimens for compressed sensing. We introduce CRDPG to train an RNN to cooperate with a CNN to complete STEM images from partial scans and present our learning policy, experiments, and example applications. After further development, we expect that adaptive scans will become the most effective approach to decrease electron beam damage and scan time with minimal information loss. Static sampling strategies are a subset of possible dynamic sampling strategies, so the performance of static sampling can always be matched by or outperformed by dynamic sampling. Further, we expect that adaptive scan systems can be developed for most areas of science and technology, including for the reduction of medical radiation. To encourage further investigation, our source code, pretrained models, and training data is openly accessible.

Data availability statement

The data that support the findings of this study are openly available at the following URL/DOI: https://doi.org/10.5281/zenodo.4384 766.

Acknowledgments

Thanks go to Jasmine Clayton, Abdul Mohammed, and Jeremy Sloan for internal review. The author acknowledges funding from EPSRC Grant EP/N035437/1 and EPSRC Studentship 1917382.

Competing interests

The author declares no competing interests.

ORCID iD

Jeffrey M Ede https://orcid.org/0000-0002-9358-5364

References

[1] Krull A, Hirsch P, Rother C, Schiffrin A and Krull C 2020 Artificial-intelligence-driven scanning probe microscopy *Commun. Phys.* **3** 1–8
[2] Rugar D and Hansma P 1990 Atomic force microscopy *Phys. Today* **43** 23–30
[3] New P F, Scott W R, Schnur J A, Davis K R and Taveras J M 1974 Computerized axial tomography with the EMI scanner *Radiology* **110** 109–23
[4] Heymsfield S B *et al* 1979 Accurate measurement of liver, kidney and spleen volume and mass by computerized axial tomography *Ann. Intern. Med.* **90** 185–7

[5] Schwartz A J, Kumar M, Adams B L and Field D P 2009 *Electron Backscatter Diffraction in Materials Science* vol 2 (Berlin: Springer)
[6] Vernon-Parry K D 2000 Scanning electron microscopy: an introduction *III-Vs Rev.* **13** 40–4
[7] Keren S *et al* 2008 Noninvasive molecular imaging of small living subjects using Raman spectroscopy *Proc. Natl Acad. Sci.* **105** 5844–9
[8] Tong Y-X, Zhang Q-H and Gu L 2018 Scanning transmission electron microscopy: a review of high angle annular dark field and annular bright field imaging and applications in lithium-ion batteries *Chin. Phys.* B **27** 066107
[9] Scarborough N M *et al* 2017 Dynamic x-ray diffraction sampling for protein crystal positioning *J. Synchrotron Radiat.* **24** 188–95
[10] Hujsak K, Myers B D, Roth E, Li Y and Dravid V P 2016 Suppressing electron exposure artifacts: an electron scanning paradigm with Bayesian machine learning *Microsc. Microanal.* **22** 778–88
[11] Egerton R F, Li P and Malac M 2004 Radiation damage in the TEM and SEM *Micron* **35** 399–409
[12] Ede J M 2020 Warwick electron microscopy datasets *Mach. Learn.: Sci. Technol.* **1** 045003
[13] Sub-Nyquist Artefacts A I and Effects S M 2015 *R. Soc. Open Sci.* **2** 140550
[14] Binev P *et al* 2012 Compressed sensing and electron microscopy *Modeling Nanoscale Imaging in Electron Microscopy* (Berlin: Springer) pp 73–126
[15] Ede J M 2020 Review: deep learning in electron microscopy (arXiv:2009.08328)
[16] Ede J M and Beanland R 2020 Partial scanning transmission electron microscopy with deep learning *Sci. Rep.* **10** 8332
[17] Li Y 2017 Deep reinforcement learning: an overview (arXiv:1701.07274)
[18] Hwang S, Han C W, Venkatakrishnan S V, Bouman C A and Ortalan V 2017 Towards the low-dose characterization of beam sensitive nanostructures via implementation of sparse image acquisition in scanning transmission electron microscopy *Meas. Sci. Technol.* **28** 045402
[19] Hujsak K, Myers B D, Roth E, Li Y and Dravid V P 2016 Suppressing electron exposure artifacts: an electron scanning paradigm with Bayesian machine learning *Microsc. Microanal.* **22** 778–88
[20] Anderson H S, Ilic-Helms J, Rohrer B, Wheeler J and Larson K 2013 Sparse imaging for fast electron microscopy *Computational Imaging XI* vol 8657 (Burlingame, CA: International Society for Optics and Photonics) p 86570C
[21] Fang L *et al* 2019 Deep learning-based point-scanning super-resolution imaging *bioRxiv* 740548
[22] de Haan K, Ballard Z S, Rivenson Y, Wu Y and Ozcan A 2019 Resolution enhancement in scanning electron microscopy using deep learning *Sci. Rep.* **9** 1–7
[23] Ede J M 2019 Deep learning supersampled scanning transmission electron microscopy (arXiv:1910.10467)
[24] Mueller K 2011 Selection of optimal views for computed tomography reconstruction US Patent App. 12/842,274
[25] Wang Z and Arce G R 2009 Variable density compressed image sampling *IEEE Trans. Image Process.* **19** 264–70
[26] Ji S, Xue Y and Carin L 2008 Bayesian compressive sensing *IEEE Trans. Signal Process.* **56** 2346–56
[27] Seeger M W and Nickisch H 2008 Compressed sensing and Bayesian experimental design *Proc. 25th Int. Conf. on Machine Learning* pp 912–19
[28] Braun G, Pokutta S and Xie Y 2015 Info-greedy sequential adaptive compressed sensing *IEEE J. Sel. Top. Sig. Process.* **9** 601–11
[29] Carson W R, Chen M, Rodrigues M R, Calderbank R and Carin L 2012 Design with application to compressive sensing *SIAM J. Imaging Sci.* **5** 1185–212
[30] Godaliyadda G D P *et al* 2017 A framework for dynamic image sampling based on supervised learning *IEEE Trans. Comput. Imaging* **4** 1–16
[31] Ermeydan E S and Cankaya I 2018 Sparse fast Fourier transform for exactly sparse signals and signals with additive Gaussian noise *Sig. Image Video Process.* **12** 445–52
[32] Saldi N, Yüksel S and Linder T 2019 Asymptotic optimality of finite model approximations for partially observed Markov decision processes with discounted cost *IEEE Trans. Autom. Control* **65** 130–42
[33] Jaakkola T, Singh S P and Jordan M I 1995 Reinforcement learning algorithm for partially observable Markov decision problems *Advances in Neural Information Processing Systems* pp 345–52
[34] Sang X, Lupini A R, Unocic R R, Chi M, Borisevich A Y, Kalinin S V, Endeve E, Archibald R K and Jesse S 2016 Dynamic scan control in STEM: spiral scans *Adv. Struct. Chem. Imaging* **2** 6
[35] Sang X, Lupini A R, Ding J, Kalinin S V, Jesse S and Unocic R R 2017 Precision controlled atomic resolution scanning transmission electron microscopy using spiral scan pathways *Sci. Rep.* **7** 43585
[36] Hochreiter S and Schmidhuber J 1997 Long short-term memory *Neural Comput.* **9** 1735–80
[37] Olah O C 2015 Understanding LSTM Networks (available at: https://colah.github.io/posts/2015-08-Understanding-LSTMs)
[38] Cho K *et al* 2014 Learning phrase representations using RNN encoder-decoder for statistical machine translation (arXiv:1406.1078)
[39] Weiss G, Goldberg Y and Yahav E 2018 On the practical computational power of finite precision RNNs for language recognition (arXiv:1805.04908)
[40] Jozefowicz R, Zaremba W and Sutskever I 2015 An empirical exploration of recurrent network architectures *Int. Conf. on Machine Learning* pp 2342–50
[41] Bayer J, Wierstra D, Togelius J and Schmidhuber J 2009 Evolving memory cell structures for sequence learning *Int. Conf. on Artificial Neural Networks* pp 755–64
[42] Pascanu R, Mikolov T and Bengio Y 2013 On the difficulty of training recurrent neural networks *Int. Conf. on Machine Learning* pp 1310–8
[43] Graves A *et al* 2016 Hybrid computing using a neural network with dynamic external memory *Nature* **538** 471–6
[44] Werbos P J 1990 Backpropagation through time: what it does and how to do it *Proc. IEEE* **78** 1550–60
[45] An R S 2016 Overview of gradient descent optimization algorithms (arXiv:1609.04747)
[46] Mnih V, Heess N, Graves A and Kavukcuoglu K 2014 Recurrent models of visual attention *Advances in Neural Information Processing Systems* pp 2204–12
[47] Ba J, Mnih V and Kavukcuoglu K 2014 Multiple object recognition with visual attention (arXiv:1412.7755)
[48] Vinyals O *et al* 2019 AlphaStar: mastering the real-time strategy game StarCraft II (available at: https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/)
[49] Lillicrap T P *et al* 2015 Continuous control with deep reinforcement learning (arXiv:1509.02971)
[50] Heess N, Hunt J J, Lillicrap T P and Silver D 2015 Memory-based control with recurrent neural networks (arXiv:1512.04455)
[51] Grabocka J, Scholz R and Schmidt-Thieme L 2019 Learning surrogate losses (arXiv:1905.10108)
[52] Konda V R and Tsitsiklis J N 2000 Actor-critic algorithms *Advances in Neural Information Processing Systems* pp 1008–14
[53] Zhao T, Hachiya H, Niu G and Sugiyama S M 2011 Analysis and improvement of policy gradient estimation *Advances in Neural Information Processing Systems* pp 262–70

[54] Rere L R, Fanany M I and Arymurthy A M 2015 Simulated annealing algorithm for deep learning *Proc. Comput. Sci.* **72** 137–44
[55] Young S R, Rose D C, Karnowski T P, Lim S-H and Patton R M 2015 Optimizing deep learning hyper-parameters through an evolutionary algorithm *Proc. Workshop on Machine Learning in High-Performance Computing Environments* pp 1–5
[56] Such F P *et al* 2017 Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning (arXiv:1712.06567)
[57] Abadi M *et al* 2016 Tensorflow: a system for large-scale machine learning *12th USENIX Symp. on Operating Systems Design and Implementation (OSDI 16)* pp 265–83
[58] Ede J M 2020 Adaptive partial STEM repository (available at: https://github.com/Jeffrey-Ede/adaptive-scans)
[59] Ede J M and Beanland R 2020 Electron microscopy datasets (available at: https://github.com/Jeffrey-Ede/datasets/wiki)
[60] Uhlenbeck G E and Ornstein L S 1930 On the theory of the Brownian motion *Phys. Rev.* **36** 823–41
[61] Plappert M *et al* 2017 Parameter space noise for exploration (arXiv:1706.01905)
[62] Fujimoto S, Van Hoof H and Meger D 2018 Addressing function approximation error in actor-critic methods (arXiv:1802.09477)
[63] Barth-Maron G *et al* 2018 Distributed distributional deterministic policy gradients (arXiv:1804.08617)
[64] Ede J M and Beanland R 2020 Adaptive learning rate clipping stabilizes learning *Mach. Learn.: Sci. Technol.* **1** 015011
[65] Raschka S 2018 Model evaluation, model selection, and algorithm selection in machine learning (arXiv:1811.12808)
[66] Roh Y, Heo G and Whang S E A 2019 Survey on data collection for machine learning: a big data-AI integration perspective *IEEE Trans. Knowl. Data Eng.* **33** 1328–47
[67] Zaremba W, Sutskever I and Vinyals O 2014 Recurrent neural network regularization (arXiv:1409.2329)
[68] McCann M T, Jin K H and Unser M 2017 Convolutional neural networks for inverse problems in imaging: a review *IEEE Signal Process. Mag.* **34** 85–95
[69] Krizhevsky A, Sutskever I and Hinton G E 2012 Imagenet classification with deep convolutional neural networks *Advances in Neural Information Processing Systems.* pp 1097–105
[70] Zhang S and Sutton R S Deeper A 2017 Look at experience replay (arXiv:1712.01275)
[71] Mnih V *et al* 2015 Human-level control through deep reinforcement learning *Nature* **518** 529–33
[72] Kingma D P and Ba J 2014 ADAM: a method for stochastic optimization (arXiv:1412.6980)
[73] Smith L N 2017 Cyclical learning rates for training neural networks *2017 IEEE Conf. on Applications of Computer Vision (WACV)* (IEEE) pp 464–72
[74] Ede J M and Beanland R 2019 Improving electron micrograph signal-to-noise with an atrous convolutional encoder-decoder *Ultramicroscopy* **202** 18–25
[75] van Hasselt H P, Guez A, Hessel M, Mnih V and Silver D 2016 Learning values across many orders of magnitude *Advances in Neural Information Processing Systems* pp 4287–95
[76] Czarnecki W M *et al* 2019 Distilling policy distillation (arXiv:1902.02186)
[77] Lipton Z C *et al* 2016 Combating reinforcement learning's Sisyphean curse with intrinsic fear (arXiv:1611.01211)
[78] Wagner P A 2011 Reinterpretation of the policy oscillation phenomenon in approximate policy iteration *Advances in Neural Information Processing Systems* pp 2573–81
[79] Jia Y, Wu Z, Xu Y, Ke D and Su K 2017 Long short-term memory projection recurrent neural network architectures for piano's continuous note recognition *J. Robot.* **2017** 2061827
[80] Ng A Y, Harada D and Russell S 1999 Policy invariance under reward transformations: theory and application to reward shaping *International Conference on Machine Learning* vol 99 pp 278–87
[81] Weill C *et al* 2019 AdaNet: a scalable and flexible framework for automatically learning ensembles (arXiv:1905.00080)
[82] Molino P, Dudin Y and Miryala S S 2019 Ludwig: a type-based declarative deep learning toolbox (arXiv:1909.07930)
[83] He X, Zhao K and Chu X 2019 AutoML: a survey of the state-of-the-art (arXiv:1908.00709)
[84] Malekhosseini E, Hajabdollahi M, Karimi N and Samavi S 2019 Modeling neural architecture search methods for deep networks (arXiv:1912.13183)
[85] Jaafra Y, Laurent J L, Deruyver A and Naceur M S 2019 Reinforcement learning for neural architecture search: a review *Image Vis. Comput.* **89** 57–66
[86] Elsken T, Metzen J H and Hutter F 2018 Neural architecture search: a survey (arXiv:1808.05377)
[87] Waring J, Lindvall C and Umeton R 2020 Automated machine learning: review of the state-of-the-art and opportunities for healthcare *Artif. Intell. Med.* **104** 101822
[88] Hanussek M, Blohm M and Kintz M 2020 Can AutoML outperform humans? An evaluation on popular OpenML datasets using AutoML Benchmark (arXiv:2009.01564)
[89] Zoph B, Vasudevan V, Shlens J and Le Q V 2018 Learning transferable architectures for scalable image recognition *Proc. Conf. on Computer Vision and Pattern Recognition* pp 8697–8710
[90] Ophus C, Ciston J and Nelson C T 2016 Correcting nonlinear drift distortion of scanning probe and scanning transmission electron microscopies from image pairs with orthogonal scan directions *Ultramicroscopy* **162** 1–9
[91] Ning S *et al* 2018 Scanning distortion correction in STEM images *Ultramicroscopy* **184** 274–83
[92] Zhu J-Y, Park T, Isola P and Efros A A 2017 Unpaired image-to-image translation using cycle-consistent adversarial networks *Proc. IEEE Int. Conf. on Computer Vision* pp 2223–32
[93] Ede J M 2020 Supplementary information: adaptive partial scanning transmission electron microscopy with reinforcement learning (Zenodo) (available at: https://doi.org/10.5281/zenodo.4384708)
[94] Seki T, Ikuhara Y and Shibata N 2018 Theoretical framework of statistical noise in scanning transmission electron microscopy *Ultramicroscopy* **193** 118–25
[95] Laine S, Karras T, Lehtinen J and Aila T 2019 High-quality self-supervised deep image denoising *Advances in Neural Information Processing Systems* pp 6968–78
[96] Miller B and Mick S 2019 Real-time data processing using python in digital micrograph *Microsc. Microanal.* **25** 234–5
[97] Noronha D H, Salehpour B and Wilton S J 2018 LeFlow: enabling flexible FPGA high-level synthesis of tensorflow deep neural networks *FSP Workshop 2018 Fifth Int. Workshop on FPGAs for Software Programmers (VDE)* pp 1–8
[98] Ruan A, Shi A, Qin L, Xu S and Zhao Y 2019 Reinforcement learning based Markov-decision process (MDP) implementation for SRAM FPGAs *IEEE Trans. Circuits Syst. II: Express Briefs* **67** 2124–8