Scaling Offline Q-Learning with Vision Transformers

Jordi Orbay* **Yingjie Miao*** **Rishabh Agarwal** **Aviral Kumar** **George Tucker**
Aleksandra Faust
Google Deepmind
*Equal contribution

Abstract

It has been shown that offline RL methods, such as conservative Q-learning (CQL), scale favorably for training generalist agents with a ResNet backbone. Recent vision and natural language processing research shows that transformer-based models scale more favorably compared to domain specific models with strong inductive biases (such as convolutional neural networks and recurrent neural networks). In this paper, we investigate how well visual transformers (ViTs) serve as backbones for CQL for training single-game agents. In this work, we enhance the Vision Transformer (ViT) for image-based RL by introducing spatio-temporal attention layers. We further investigate the impact of various embedding sequence aggregation methods on ViT performance. Overall, our modified ViT outperforms the standard ViTs in the single-game Atari setting.

1 Introduction

Modern advances in foundation models clearly illustrate that training high-capacity neural networks on large datasets is crucial for obtaining generalist models that can solve a variety of tasks. In the domain of reinforcement learning and decision making, one would expect that offline reinforcement learning (RL) approaches – that seeks to learn decision-making policies directly from a static dataset, without any active environment interaction – would enable us to replicate a similar success story by combining the policy improvement benefits offered with RL algorithms with architectural and scaling benefits offered by foundation models. But is this really the case?

Empirical results from large-scale decision-making studies (Kumar et al., 2023; Lee et al., 2022; Chebotar et al., 2023) show that it is possible to devise scalable approaches for offline RL that can train on large databases of prior experience. For instance, (Kumar et al., 2023; Chebotar et al., 2023) develop methods for running offline Q-learning on multi-task data and (Lee et al., 2022) develops methods to train transformer policies via return-conditioned supervised learning (RCSL). While these initial results are very promising, utilizing the best neural network architecture designs from modern foundation models has been challenging: offline Q-learning approaches from the aforementioned prior works often fail with standard transformer-based architectures and require non-trivial amounts of architectural design to perform well (Li et al., 2023); whereas RCSL approaches that alleviate this issue with architectural design often fail to attain substantial amounts of policy improvement and lag behind Q-learning methods, especially with sub-optimal training data.

In this paper, we tackle an instance of the above challenge in the context of offline Q-learning. Concretely, in this paper, we attempt to develop an entirely transformer-based architecture for parameterizing value functions, which is then trained entirely via offline Q-learning. We specifically consider problems with visual observations, where an entire transformer based architecture utilizes a vision transformer (ViT) (Dosovitskiy et al., 2020). Our key insight is that while a vision transformer is hard to utilize in conjunction with offline Q-learning approaches, we can attain better performance by utilizing a variant of attention, that we call, spatio-temporal attention. This approach first applies temporal attention to aggregate information across observations appearing in multiple time-

Foundation Models for Decision Making Workshop at NeurIPS 2023

steps followed by spatial attention across patches in a given frame. Our results show that training this modified architecture approach outperforms standard vision transformers trained via offline Q-learning.

Discussion of the most related works. Previous work (Kumar et al., 2023) has scaled offline Q-learning using ResNets on multi-game Atari. This work aims to drive further scaling by improving the performance of transformer-based Q-learning architectures. Other works have trained transformers on offline reinforcement learning using return-conditioned behavioral cloning Lee et al. (2022). We aim to bring similar benefits to offline Q-learning. Kalantari et al. (2022) trained offline Q-learning agents with an architecture combining convolutional neural networks and transformers. We focus on only using transformers and feed-forward networks.

2 Background

We consider sequential-decision making problems where on each timestep, an agent observes a state s, produces an action a, and receives a reward r. The goal of a learning algorithm is to maximize the sum of discounted rewards. Our approach is based on conservative Q-learning (CQL) (Kumar et al., 2020), an offline Q-learning algorithm. CQL uses a sum of two loss functions: **(i)** standard TD-error that enforces Bellman consistency, and **(ii)** a regularizer that minimizes the Q-values for unseen actions at a given state, while maximizing the Q-value at the dataset action to counteract excessive value underestimation. Denoting $Q_\theta(s, a)$ as the learned Q-function, the training objective for CQL is given by:

$$\min_\theta \; \alpha \left(\mathbb{E}_{s \sim \mathcal{D}} \left[\log \left(\sum_{a'} \exp(Q_\theta(s, a')) \right) \right] - \mathbb{E}_{s,a \sim \mathcal{D}} \left[Q_\theta(s, a) \right] \right) + \mathsf{TDError}(\theta; \mathcal{D}), \qquad (1)$$

where α is the regularizer weight, which we fix to $\alpha = 0.05$ based on preliminary experiments unless noted otherwise. Kumar et al. (2020) utilized a distributional $\mathsf{TDError}(\theta; \mathcal{D})$ from C51 (Bellemare et al., 2017), whereas (Kumar et al., 2021b) showed that similar results could be attained with the standard mean-squared TD-error. Following Kumar et al. (2023), we use the distributional formulation of CQL.

Setup. Our goal is to train policies using CQL on a fixed offline dataset. We utilize the set of five Atari games from Agrawal et al. (2020). For each game, we utilize the DQN-Replay dataset (Agrawal et al. (2020)). Specifically, we uniformly sample 5% of the data from the DQN-Replay dataset for our offline RL experiments. Our hyperparameters for the single-game training setup are listed in Table 2.

Evaluation. We evaluate our models with sticky actions, meaning there is a 25% probability at every time step that the environment will execute the previous action again instead of the new action commanded. To ensure that our evaluations are reliable, for reporting performance, we follow the recommendations by Agarwal et al. (2021). Specifically, we report interquartile mean (IQM) normalized scores, which is the average scores across middle 50% of the runs, combined across all games, as well as performance profiles for qualitative summarization.

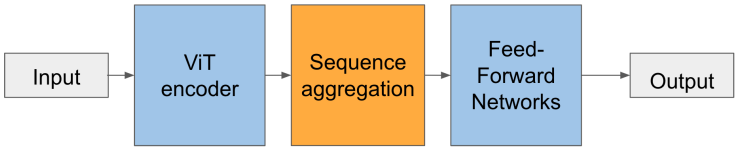
3 ViTs for Offline RL

We run our experiment across two different ViT architectures:

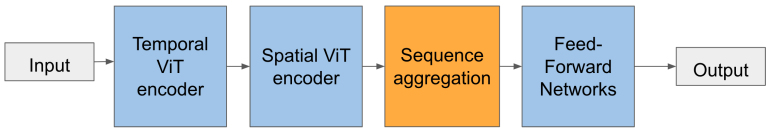
- Vision transformer (ViT) from Dosovitskiy et al. (2020).
- Vision transformer with spatio-temporal attention (ViT-ts).

Standard ViT. A ViT is an architecture that splits images into patches, adds positional embeddings to the patch embeddings, and feeds the sequences to a transformer encoder. Since vision transformers have achieved the state of the art in several vision tasks (Dosovitskiy et al. (2020) and Zhai et al. (2022)), we explored using ViT as a CQL policy's backbone. We chose our ViT to match the parameters of ViT-S in Steiner et al. (2022). In our setup illustrated in 1a, we first rearranged our four stacked frames into 144 patches (flattened from 4 frames, with 36 patches per frame) per step. The patches are passed through feedforward layers and the ViT-S encoder. The ViT output is aggregated. This is flattened into a vector that is fed into feed-forward networks.

ViT-ts architecture The Vit-ts architecture in Figure 1b is similar to the ViT architecture except that it uses two ViT encoders sequentially. Our use of spatio-temporal attention is adapted from Aksan et al. (2021), but we use the two temporal and spatial transformers sequentially, whereas Aksan et al. (2021) uses them in parallel. First, the patches are arranged so that the height and width axes of each patch are rearranged into the batch axis. We then pass these vectors into a temporal ViT encoder that does attention over the different frames per batch. After this, the embedding is then rearranged to put the height and width dimensions back into the token dimension. We pass the resultant embedding through a spatial ViT encoder, after which we proceed as the ViT backbone does.



(a) **ViT architecture** The ViT input is composed of four stacked frames that are rearranged into 144 patches. They pass through a single feed-forwrd layer before continuing to the encoder. The encoder attends over all 144 patch embeddings. The encoder output is then aggregated and passed through feed-forward networks.



(b) **ViT-ts architecture** The ViT-ts input is similar to the ViT input except that it is rearranged so that the Temporal ViT encoder attends across only the four temporal frame embeddings per patch. After the temporal ViT encoder, the embedding is then rearranged so that the Spatial ViT encoder attends over the height and width dimensions per embedding. Following that, the architecture continues similarly to the ViT.

Sequence aggregations. We run our ViT experiments with three methods for aggregating the embedding sequences after the ViT encoder.

- No aggregation: We only take the last dimension along the sequence dimension.
- Global average pooling: We take the mean along the sequence dimension.
- Learned spatial embeddings (Kumar et al., 2023): We use learned spatial embeddings that learn a matrix that point-wise multiplies the output of the ViT encoder.

In the ViT-ts architecture, the sequence aggregation is taken after the second ViT encoder for both "no aggregation" and learned spatial embedding. For global average pooling, the mean is taken after the first and second ViT encoder.

All models and sequence aggregation methods are listed with their number of parameters in Table 1. The learned spatial embeddings used by both models have about 49M parameters.

Table 1: **Number of parameters per model and sequence aggregation combination.** Number of parameters are listed in millions.

Model	Number of Parameters
ViT with no sequence aggregation	24.2M
ViT with mean pooling	24.2M
ViT with learned spatial embeddings	73.9M
ViT-ts with no sequence aggregation	45.5M
ViT-ts with mean pooling	45.5M
ViT-ts with learned spatial embeddings	95.3M
Convnet	63.5M

4 Results and Discussion

4.1 Results

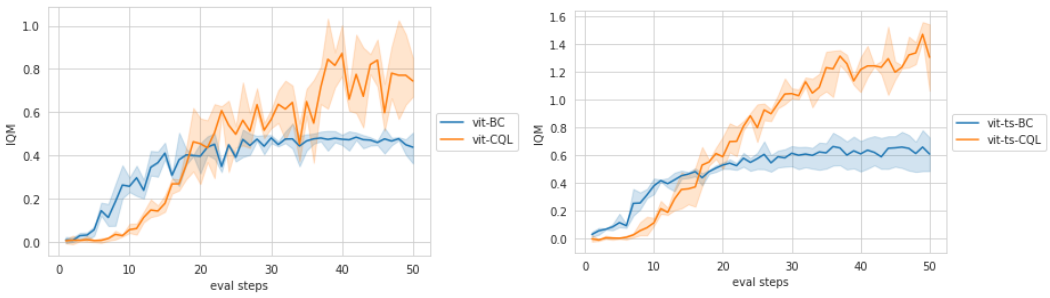
We measure results using average DQN-normalized interquartile-mean (IQM) (Agarwal et al., 2021). The DQN scores we normalize against are listed in Table 3.

Architecture Baseline. As a baseline, we report results using the convolutional neural network (convnet) from Agrawal et al. (2020) and Kumar et al. (2021a). The Convnet backbone we utilize has three convolutional layers and a hidden feedforward layer of 512 features.

Behavioral cloning vs CQL findings Our results are summarized in 2. CQL outperforms behavioral cloning when using the ViT backbone and when using a ViT-ts backbone. All models are using no sequence aggregation here. With ViT, the highest value achieved by CQL is 1.03 IQM, whereas behavioral cloning achieved a maximum of 0.55 IQM. With Vit-ts, the highest value achieved by CQL is 1.56 IQM, whereas behavioral cloning achieved a maximum of 0.78 IQM.

Architecture findings Our results are summarized in Figure 3. ViT-ts reached the highest performance within the observed steps. The maximum IQM achieved by an individual ViT-ts run is 1.68 IQM. ViT's maximum score is 1.21 IQM and convnet's maximum score is 1.35 IQM. The median ViT-ts also surpassed both of these opposing maximum scores before ViT and convnet reached them, showing that it was also the more sample-efficient algorithm. Vit-ts, ViT, and convnet ran at approximately 12,24, and 48 training steps per second, respectively.

Sequence aggregation findings Our results are summarized in Figure 4. When testing the three different sequence aggregation methods, we find that the mean performance of using learned sequence embeddings outperforms the other two methods in ViT. However, the variance of using both other methods are large enough to intersect with the performance of mean learned sequence aggregation. In ViT-ts, the mean performance of using global average pooling outperforms the other two methods. ViT-learned reached a maximum score of 1.35 IQM and vit-ts-mean reached a maximum score of 1.93 IQM. ViT-ts also consistently outperforms ViT with all three sequence aggregation methods.



(a) ViT. (b) ViT-ts.

Figure 2: **Behavioral cloning vs CQL with no sequence aggregation.** CQL outperforms behavioral cloning when using a ViT or a ViT-ts backbone, as expected. Each eval step in the x-axis here and in the other figures occurs every 62,500 training environment steps.

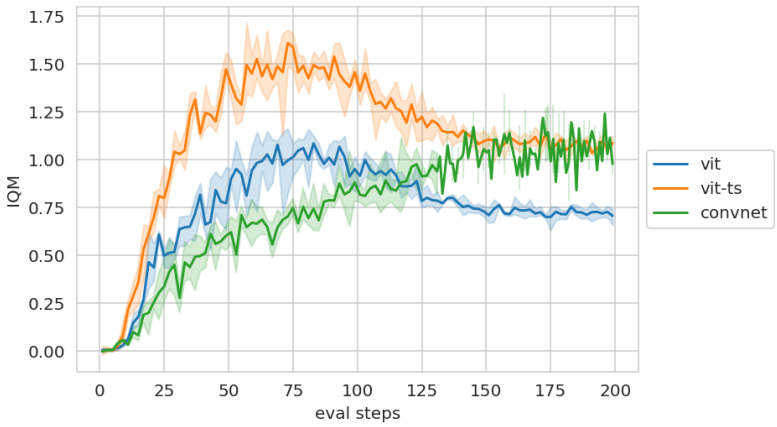
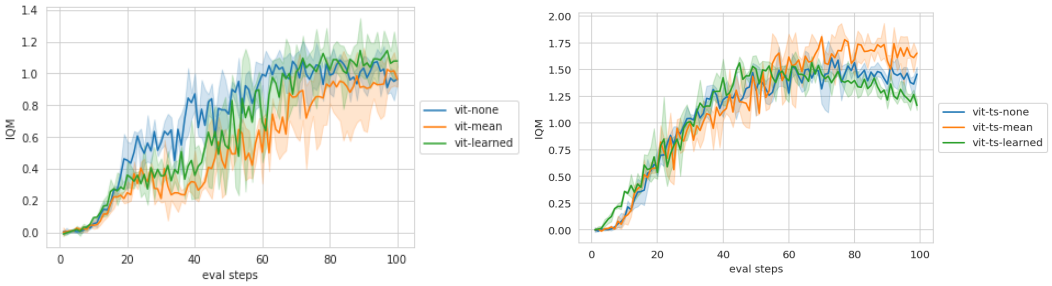


Figure 3: **CQL using ViT, ViT-ts, and convnet as backbones.** The ViT-ts outperformed the other models, attaining an IQM score of 1.68. ViT-ts reliably outperformed convnet in all seeds.



(a) ViT.

(b) ViT-ts.

Figure 4: **CQL runs using either no sequence aggregation (none), global average pooling (mean), or learned spacial embeddings (learned).** In ViT, vit-learned attained the highest IQM score of 1.35. In ViT-ts, vit-ts-mean attained the highest IQM of 1.93.

4.2 Discussion

This work shows that CQL outperforms behavioral cloning when using a ViT backbone. It shows that adding spatio-temporal attention to ViT aids the policy in using information from all four frames for decision making. As we hoped, we recorded similar trends as those observed in Aksan et al. (2021). It shows that when running CQL with a ViT backbone without spatio-temporal attention, the mean score for the learned spatial aggregation models surpasses the other two models within the observed steps, but no sequence aggregation method reliably surpasses the others. When running a ViT backbone with spatio-temporal attention, the mean pooling models surpass the other two models within the observed steps.

Although we improved on ViT's performance, our setup is not optimal. The scale of the experiments limited the number of ablations we could explore. We expect future experiments to improve on our results. For example, unlike decision transformer Chen et al. (2021), we did not incorporate data augmentation into our experiments, which we expect should improve results. We also did not vary the number of layers in either the spatial or temporal attention layers, which can likely be optimized. Optimizing will be needed to confirm that ViT-ts models surpass ViT models when they are constrained to the same number of parameters. Addressing these improvements is left to future work.

5 Acknowledgements

We thank Igor Mordatch and Kamyar Ghasemipour for informative discussions.

Table 2: **Hyperparameters used by single-game training.** Here we report the key hyperparameters used by the single-game training. These are typical for single-game training following Agrawal et al. (2020).

Hyperparameter	Setting (for both variations)
Eval Sticky actions	Yes
Grey-scaling	True
Observation down-sampling	(84, 84)
Frames stacked	4
Frame skip (Action repetitions)	4
Reward clipping	[-1, 1]
Terminal condition	Game Over
Max frames per episode	108K
Discount factor	0.99
Mini-batch size	32
Target network update period	every 2000 updates
Training environment steps per iteration	62.5k
Update period every	1 environment steps
Evaluation ϵ	0.001
Evaluation steps per iteration	125K
Learning rate	.00001
n-step returns (n)	3
CQL regularizer weight α	0.05

Table 3: **Average returns across 5 runs for the random agent and the average performance of the trajectories in the DQN (Nature) dataset.** For Atari normalized scores reported in the paper, the random agent is assigned a score of 0 while the average DQN replay is assigned a score of 100. Note that the random agent scores are also evaluated on Atari 2600 games with sticky actions. These values are from Kumar et al. (2021b).

Game	Random	Average DQN-Replay
Asterix	279.1	3185.2
Breakout	1.3	104.9
Pong	-20.3	14.5
Qbert	155.0	8249.7
Seaquest	81.8	1597.4

References

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A. C., and Bellemare, M. (2021). Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320.

Agrawal, R., Schuurmans, D., and Norouzi, M. (2020). An optimistic perspective on offline reinforcement learning. In *ICML*.

Aksan, E., Kaufmann, M., Cao, P., and Hilliges, O. (2021). A spatio-temporal transformer for 3d human motion prediction. In *3DV*, pages 565–574. IEEE.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, pages 449–458. PMLR.

Chebotar, Y., Vuong, Q., Irpan, A., Hausman, K., Xia, F., Lu, Y., Kumar, A., Yu, T., Herzog, A., Pertsch, K., Gopalakrishnan, K., Ibarz, J., Nachum, O., Sontakke, S., Salazar, G., Tran, H. T., Peralta, J., Tan, C., Manjunath, D., Singht, J., Zitkovich, B., Jackson, T., Rao, K., Finn, C., and Levine, S. (2023). Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. In *NeurIPS*, pages 15084–15097.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929.

Kalantari, A. A., Amini, M., Chandar, S., and Precup, D. (2022). Improving sample efficiency of value based models using attention and vision transformers.

Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. (2023). Offline q-learning on diverse multi-task data both scales and generalizes. In *ICLR*. OpenReview.net.

Kumar, A., Agarwal, R., Ghosh, D., and Levine, S. (2021a). Implicit under-parameterization inhibits data-efficient deep reinforcement learning.

Kumar, A., Agarwal, R., Ma, T., Courville, A., Tucker, G., and Levine, S. (2021b). Dr3: Value-based deep reinforcement learning requires explicit regularization.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative q-learning for offline reinforcement learning.

Lee, K., Nachum, O., Yang, M., Lee, L., Freeman, D., Guadarrama, S., Fischer, I., Xu, W., Jang, E., Michalewski, H., and Mordatch, I. (2022). Multi-game decision transformers. In *NeurIPS*.

Li, W., Luo, H., Lin, Z., Zhang, C., Lu, Z., and Ye, D. (2023). A survey on transformers in reinforcement learning.

Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., and Beyer, L. (2022). How to train your vit? data, augmentation, and regularization in vision transformers.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2022). Scaling vision transformers. In *CVPR*, pages 1204–1213. IEEE.