VRL3: A Data-Driven Framework for Visual Deep Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose VRL3, a powerful data-driven framework with a minimalist design for solving highly challenging visual deep reinforcement learning (DRL) tasks. We analyze a number of major obstacles in taking a data-driven approach, and present a suite of design principles, novel findings, and critical insights about data-driven visual DRL. Our framework has three stages: in stage 1, we leverage non-RL datasets (e.g. ImageNet) to learn task-agnostic visual representations; in stage 2, we use offline RL data (e.g. a limited number of expert demonstrations) to convert the task-agnostic representations into more powerful task-specific representations; in stage 3, we fine-tune the agent with online RL. On a set of highly challenging hand manipulation tasks with sparse reward and realistic visual inputs, compared to the previous SOTA, VRL3 achieves an average of 780% better sample efficiency. And on the hardest task, VRL3 is 1220% more sample efficient and solves the task with only 10% of the computation. These highly significant results clearly demonstrate the great potential of data-driven deep reinforcement learning.

1 Introduction

Over the past few years, the sample efficiency of Deep Reinforcement Learning (DRL) has significantly improved in popular benchmarks such as Gym [3, 34] and DeepMind Control Suite (DMC) [33]. However, the environments in these benchmarks often look very different from the real world. It remains unclear whether the sample-efficient methodologies developed for these benchmarks can be successfully extended to more practical tasks that rely on realistic visual inputs from cameras and sensors, such as in robotic control and autonomous driving.

A promising direction is to take a data-driven approach, that is, try to use all the data we can find that might contribute to the learning process. Indeed, in the past few decades, the most important advances in deep learning have been facilitated by the use of large amounts of data and computation [32, 20]. Such a data-driven approach is very natural and mature for supervised learning, but not as much so for RL, where learning is most commonly performed in an online fashion.

Although recent advances in representation learning for RL and offline RL reveal the potential of such an approach, most existing offline RL methods focus on tasks with proprioceptive ("raw-state") input and not the highly challenging visual control tasks. There is yet to be a method that can fully utilize data from multiple different sources to boost learning efficiency. In this work, we aim to develop a data-driven framework that fills this role. We want it to be very sample efficient, and we want to take a minimalist approach and make this framework as simple as possible. This is very important since in DRL, the performance of RL algorithms can be heavily affected by hidden details in the code, and as new algorithms become more complex, they can be challenging to analyze, reproduce and tune towards new tasks [10, 11, 6].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

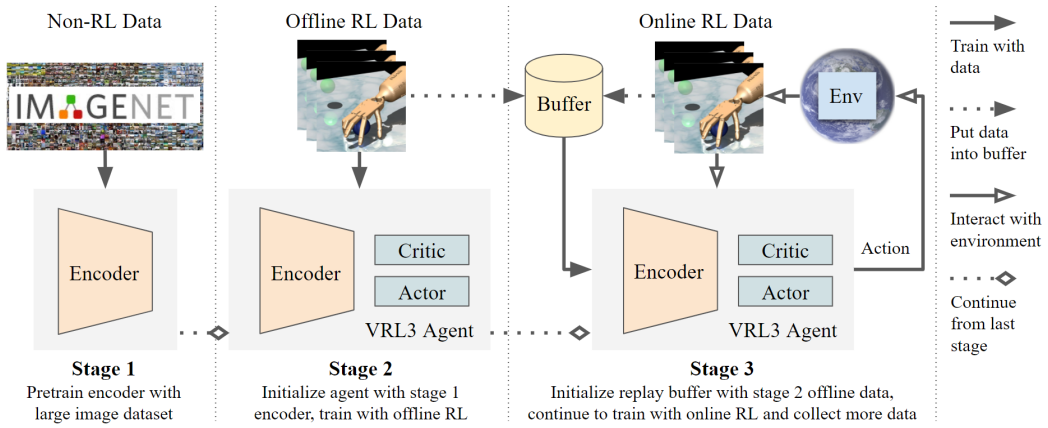


Figure 1: The design of VRL3. In stage 1, we pretrain an encoder using large image datasets to obtain task-agnostic visual representation. In stage 2, we initialize an actor-critic RL agent with the pretrained encoder, and then use offline RL techniques to train on offline RL data. Here we finetune the encoder and also learn the actor and critic. In stage 3, we initialize a buffer with the offline data from stage 2, and further train the entire agent with online RL. The idea is simple: to address highly challenging visual control tasks, we want to fully exploit all the available data.

In later sections, we show that our design of the framework is very simple, intuitive, and effective. We also make an effort to maximize robustness and reproducibility, by comparing to stronger baselines that we re-implemented and improved, presenting extensive ablations and a hyperparameter robustness study, listing full technical details, providing open source code, and more.

Figure 1 shows the proposed framework, Visual Deep Reinforcement Learning in 3 Stages (VRL3). VRL3 is designed with a data-driven mindset, and in a minimalist fashion, Each stage is designed to utilize a particular type of data source.

In stage 1, we learn from large, existing non-RL datasets such as the ImageNet dataset. These datasets might not be directly relevant to the RL task, but they provide task-agnostic visual representations that might be helpful, especially if the RL task also has realistic visuals. A major challenge here is that the non-RL data can come from a different domain with different input shapes. Special care is required to make sure the representations obtained here can be smoothly transferred to the RL task.

In stage 2, we learn from offline RL data, such as a small number of expert demonstrations, or data collected by a previous learning agent. This offline data is related to the RL task and contains important information about solving the task. We initialize an actor-critic RL agent with the pretrained encoder and then employ offline RL techniques which not only train the actor-critic RL agent but also fine-tune the encoder, thereby creating a task-specific representation.

In stage 3, we learn from online RL data. Our main goal is sample efficiency (achieve high performance with as little online data as possible). We, therefore, use off-policy training, which uses and re-uses all available data during training. Note that naively transitioning into standard online RL can lead to training instability and destroy what we learned from the previous stages. We avoid this instability by constraining the maximum Q target value during off-policy updates.

Although each stage brings a number of unique challenges, we show that VRL3 can address all of them and take full advantage of the different data sources. Our contributions are as follows:

1. Novel framework: we present a data-driven framework that can fully exploit non-RL, offline RL, and online RL data to solve challenging tasks with sparse reward and realistic visual input. Most existing visual methods do not utilize offline data, and most offline RL methods do not work with difficult visual tasks, making our framework a novel contribution. We provide source code (in supplementary materials) and a full set of technical details to maximize reproducibility.
2. Novel technical contributions: we propose convolutional channel expansion (CCE) to enable an ImageNet-pretrained encoder to work with RL tasks that take in multiple input images. We also propose the Safe Q target technique for minimal-effort offline to online transition.

3. Novel insights: we show that offline RL data can provide a benefit that cannot be replaced by other stages, and that conservative RL might be a superior option when exploiting offline data, compared to contrastive learning. We also show that a reduced encoder learning rate helps learning in a multi-stage setting. And very interestingly, representation pretrained with non-RL data, despite the large domain gap, can give better performance compared to representation learned from limited RL data, but is outperformed when RL data is ample. We also find stage 1 pretraining helps more in more difficult tasks.
4. Novel results: We present extensive ablations that reveal the effect of each component of VRL3, and conduct a large hyperparameter study on tasks with different difficulties, providing useful guidelines on how to deploy and tune our framework to new tasks.
5. Significant improvement over the previous SOTA: VRL3 achieves an entirely new level of SOTA sample efficiency, parameter efficiency, and computation efficiency on the highly challenging Adroit benchmark while also being very competitive on DMC.

2 Related Work

VRL3 follows a minimalist design and effectively combines representation learning, offline RL, and online RL. We now discuss related work and how our contribution is novel and different from prior works, with a focus on model-free methods. Additional related work is discussed in Appendix D.

2.1 Learning visual tasks with online RL

With the help of data augmentation, many visual RL tasks can be learned in an online fashion. Typically, agents trained with such methods use convolutional encoders to extract lower-dimensional features from the visual input, and train in an end-to-end fashion [14, 39, 17, 40]. Another class of methods uses contrastive representation learning to solve visual tasks [18, 31, 43]. Contrastive learning can be used to learn useful representations without gradients from the RL objectives. These methods work well in environments with simple visuals, such as in the DMC environments, but can fail entirely if naively applied to highly challenging control tasks with sparse rewards and more realistic visuals such as Adroit, due to the difficulty in both exploration [27] and learning representations for realistic visual inputs [30]. Different from these works, VRL3 can fully exploit offline RL data and even non-RL data and can achieve SOTA performance in both Adroit and DMC.

2.2 RL with ImageNet pretraining

A number of previous works have used a pretrained encoder, such as a ResNet trained on ImageNet to help convert visual input into low-dimensional features, the encoder can be further finetuned or kept frozen during training [30, 5, 12, 9, 21, 26, 29]. However, such pretraining alone might not be sufficient to achieve the best performance or computation efficiency. Naively initializing the agent with an ImageNet-pretrained encoder might not provide a significant advantage over training from scratch [12]. Most of these works do not incorporate offline RL data. RRL [30] uses offline RL data but does not exploit them to the full extent. Different from these works, we show how ImageNet pretraining can be best combined with offline and online RL and provide a series of novel analyses, interesting results, and important insights.

2.3 Offline RL and data-driven RL

In the past few decades, some of the most important advances in AI research have been based on data-driven methods, so it is natural to ask whether we can also utilize a similar mindset for DRL [20, 32]. Although off-policy algorithms do not naively work in offline settings [7, 22], many specialized algorithms are proposed to allow effective offline training [7, 16, 4, 25, 36, 15, 6, 35, 13, 38, 1, 8, 37]. Recent works also study the transition from offline to online [23, 19, 44, 19, 13, 22]. However, most of these works are tested on tasks with proprioceptive (positions & velocities) input, and they do not consider pretraining from non-RL data. Our work is very different and is focused on highly challenging control tasks with visual input, sparse reward, and limited demonstrations.

Note that Adroit has often been studied with proprioceptive and not the more challenging visual input. RRL [30] and FERM [42] are the most relevant to our work. RRL uses an ImageNet-pretrained

encoder, performs behavioral cloning (BC) on offline RL data, and then performs on-policy learning. Prior to our work, RRL is the SOTA on visual Adroit and is the only method that achieves non-trivial performance on the hardest Relocate task. FERM performs contrastive learning on offline RL data and then performs off-policy online RL with data augmentation. FERM can learn simple robotic tasks quickly but has brittle performance in Adroit [30].

In most prior works, the different training stages have been **studied separately**. What is missing is a framework that can **combine all 3 stages effectively and efficiently**. A major novel contribution of this work is such a framework and a comprehensive study on how its components interact and build up towards superior performance.

3 VRL3: Visual DRL in 3 stages

We now present the design of our framework VRL3. Algorithm 1 gives the pseudocode of VRL3 . We focus on the high-level ideas and provide additional technical details in Appendix A. Our agent has a convolutional encoder, two Q networks, two target Q networks and a policy network. Note the encoder can be trained separately from the policy and Q networks.

Algorithm 1 Visual DRL in 3 stages

Initialize parameterized encoder f_ξ, policy π_ϕ, Q functions $Q_{\theta_1}, Q_{\theta_2}$, Q target functions $Q_{\bar{\theta}_1}, Q_{\bar{\theta}_2}$, empty buffer $\mathcal{D}$. Set target parameters $\bar{\theta}_i \leftarrow \theta_i,$ for $i = 1, 2$. N_{S2} is the number of offline RL updates in stage 2; N_{S3} number of online RL updates for stage 3.
Stage 1:
Load pretrained encoder parameters into ξ and setup encoder for RL training.
Stage 2:
Load offline RL data into $\mathcal{D}$
for N_{S2} updates **do**
  Sample a mini-batch B from $\mathcal{D}$
  Update f_ξ and $Q_{\theta_1}, Q_{\theta_2}$ with $\mathcal{L}_Q + \mathcal{L}_{QC}$, update π_ϕ with $\mathcal{L}_\pi$. Update $Q_{\bar{\theta}_1}, Q_{\bar{\theta}_2}$ with polyak averaging.
Stage 3:
for N_{S3} updates **do**
  Sample an action from policy, take action and store the new data in $\mathcal{D}$
  Sample a mini-batch B from $\mathcal{D}$
  Update f_ξ and $Q_{\theta_1}, Q_{\theta_2}$ with $\mathcal{L}_Q$, update π_ϕ with $\mathcal{L}_\pi$. Update $Q_{\bar{\theta}_1}, Q_{\bar{\theta}_2}$ with polyak averaging.

Stage 1 We follow the standard supervised learning routine and train a convolutional encoder f_ξ on a 1000-class ImageNet classification task. We follow the standard training procedure of a typical ResNet model. During training, we shrink the images to the size of 84x84 so that they match the input dimension of our downstream RL task. In our main results, we also use a lightweight encoder with 5 convolutional layers, with batch norm after each layer.

Transition to multi-frame RL input After stage 1, we expand the first convolutional layer of the encoder so the input channel size matches the RL task which can take in multiple frames, and the weight values in this layer are divided accordingly to maintain features to later layers, we refer to this as **convolutional channel expansion (CCE)** (Further discussed in Section 5).

Stage 2 We first initialize an RL agent with the pretrained encoder and put the offline RL data in an empty replay buffer $\mathcal{D}$. We build our implementation on top of DrQv2 because it is a simple off-policy actor-critic algorithm with SOTA performance and a very clean and efficient codebase [39]. Here we also try to follow the notations in DrQv2 for better consistency.

Input and image augmentation Let aug denote random shift image augmentation, and cat for concatenation. Let $\boldsymbol{x}$ be the visual input from the camera (or cameras). In the Adroit environment, in addition to the visual observation, we are also given sensor values from the robotic hand, which is a vector of real numbers, denoted by $\boldsymbol{z}$. The augmented visual input (image frames) is put into the encoder, giving a lower-dimensional representation $\boldsymbol{h} = f_\xi(\mathrm{aug}(\boldsymbol{x}))$. We then concatenate it with the

sensor values (this is skipped in DMC). For simplicity, let $\boldsymbol{s} = \mathrm{cat}(\boldsymbol{h}, \boldsymbol{z})$ denote the concatenation; $\boldsymbol{s}$ is the input to the Q and policy networks, which are multi-layer perceptrons.

Offline RL update For each update, we sample a mini-batch of transitions $(\boldsymbol{x}_t, \boldsymbol{z}_t, \boldsymbol{a}_t, r_{t:t+n-1}, \boldsymbol{x}_{t+n}, \boldsymbol{z}_{t+n})$ from the replay buffer $\mathcal{D}$. Note we have $t + n$ here to compute n-step returns [39]. Let $\boldsymbol{s}_t = \mathrm{cat}(\boldsymbol{h}_t, \boldsymbol{z}_t)$. Let $\tilde{\boldsymbol{a}}_t \sim \pi_\phi(\boldsymbol{s}_t)$ be actions sampled from the current policy. The Q target value y is: $y = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \min_{k=1,2} Q_{\bar{\theta}_k}(\boldsymbol{s}_t, \tilde{\boldsymbol{a}}_{t+n})$.

The standard Q loss $\mathcal{L}_Q$ is (for $k = 1, 2$): $\mathcal{L}_Q(\mathcal{D}) = \mathbb{E}_{\tau\sim\mathcal{D}}\big[(Q_{\theta_k}(\boldsymbol{h}_t, \boldsymbol{a}_t) - y)^2\big]$.

Following [16], and build on top of our framework, we use an additional conservative Q loss $\mathcal{L}_{QC}$ term: $\mathcal{L}_{QC}(\mathcal{D}) = \mathbb{E}_{\tau\sim\mathcal{D}}\big[\log \sum_{\tilde{\boldsymbol{a}}_t} \exp(Q(\boldsymbol{s}_t, \tilde{\boldsymbol{a}}_t)) - Q(\boldsymbol{s}_t, \boldsymbol{a}_t)\big]$. On a high-level, this loss essentially reduces the Q value for actions proposed by the current policy, and increase the Q values for actions in the replay buffer [16].

The standard policy (actor) loss $\mathcal{L}_\pi$ is: $\mathcal{L}_\pi(\mathcal{D}) = -\mathbb{E}_{\tau\sim\mathcal{D}}\big[\min_{k=1,2} Q_{\theta_k}(\boldsymbol{s}_t, \boldsymbol{a}_t)\big]$.

Let α be the learning rate for the policy network and the Q networks. Let β_{enc} be the encoder learning rate scale, so that the encoder learning rate is $\alpha_{\text{enc}} = \beta_{\text{enc}}\alpha$. For each offline RL update in stage 2, we perform gradient descent on $\mathcal{L}_Q(\mathcal{D}) + \mathcal{L}_{QC}(\mathcal{D})$ to update the encoder f_ξ and the Q networks $Q_{\theta_1}, Q_{\theta_2}$, and gradient descent on $\mathcal{L}_\pi$ to update the policy π_ϕ. Note that we do not use the gradient from the policy to update the encoder, following previous work [40, 39]. We also use Polyak averaging hyperparameter τ to update target networks, as is typically done.

Safe Q target technique When computing the Q target in stages 2 and 3, we propose a simple yet effective solution to prevent potential Q divergence due to training instability and distribution shift. For all tasks, we set a maximum Q target value. To decide this threshold value, we reshape the maximum per-step reward $r_{\max}$ to be 1 and estimate what the maximum Q value should be for an optimal policy. To be concrete, we can compute the sum of this infinite geometric series, $Q_{\max} = \sum_{t=0}^{\infty} \gamma^t r_{\max} = 100$ when $\gamma = 0.99$ (γ is discount factor). To allow more flexibility in the Q network, we can also set the threshold $Q_{\max}$ to be higher than this value. Now if during any Q update, a Q target value y exceeds this threshold value, then we reduce it to be closer to the threshold value: if $y > Q_{max} + 1$, then $y \leftarrow Q_{\max} + (y - Q_{\max})^\eta$, where $0 \leq \eta \leq 1$ is the safe Q factor. Notice the $Q_{\max} + 1$ here is a technical detail to address edge cases.

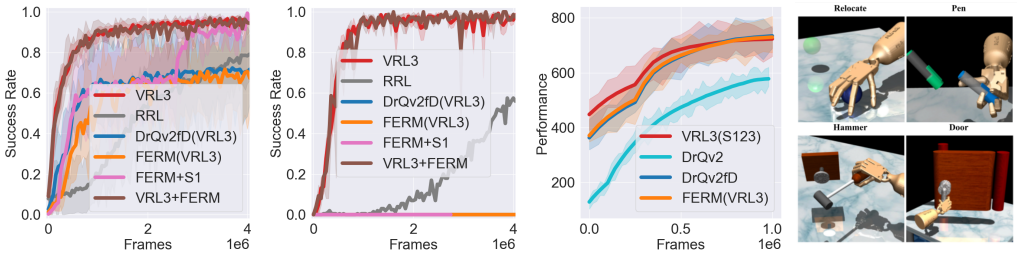
Reduced encoder learning rate Due to the domain gap between ImageNet and the RL task, we apply a reduced encoder learning rate to avoid the disturbance of pretrained features in stages 2 and 3.

Stage 3 We continue to use the replay buffer $\mathcal{D}$, which already contains the offline RL data. We now perform standard online RL, and newly collected data are added to $\mathcal{D}$. The update is the same as in stage 2, except we remove the conservative Q loss $\mathcal{L}_{QC}(D)$.

4 Results

Figure 2a shows success rate comparison averaged over all 4 Adroit tasks, Figure 2b shows results on the most challenging Relocate task, and Figure 2c shows average performance comparison on all 24 DMC tasks. We use the following methods as our baselines. DrQv2 is an off-policy online RL method that has the SOTA performance on DMC. RRL is an on-policy method that uses pretrained ImageNet features and utilizes demonstrations with behavioral cloning and has SOTA performance on Adroit. FERM is a 2-stage method that first trains an encoder with contrastive learning on offline RL data and then perform online RL. Note that currently there are not many successful multi-stage visual RL frameworks like VRL3, and as discussed in section 2, most existing offline-online methods are not designed to work with the more challenging visual inputs, so they are not included.

Getting stronger baselines Naively, DrQv2 does not use offline data and does not work well in Adroit due to the difficulty in exploration. And FERM has brittle performance on Adroit [30]. To create stronger baselines, we introduce variants of DrQv2 and FERM, which are both built on top of our framework. Specifically, DrQv2fD(VRL3) does not have stage 2 training but starts stage 3 training with demonstrations in the buffer. FERM(VRL3) is our re-implementation based on VRL3 and has much stronger performance compared to the results in [30] (more robust, is about 2 times faster, and reaches a 1.25x average success rate). For Adroit, we also compare with FERM+S1, which is FERM(VRL3) plus S1 pretraining, and VRL3+FERM, which is VRL3 with additional contrastive



(a) 4 Adroit tasks (b) Relocate (c) 24 DMC tasks (d) Adroit Suite [27]

Figure 2: (a) Average success rate over 4 visual Adroit tasks. The performance for VRL3 is averaged over 10 seeds (seeds 0-9). To ensure consistency with prior work, results for RRL (Adroit SOTA) are provided by the authors. VRL3 achieves the strongest performance with a minimalist design. (b) Success rate for the hardest Relocate task. (c) Average performance over all 24 DMC tasks. VRL3 is still the best, though the performance gap is smaller. Results for DrQv2 are provided by the authors.

encoder learning in stage 2. For these improved baselines, we use the hyperparameters we searched for VRL3 for maximum performance. Doing so helps to ensure our comparison actually reflects the differences in algorithmic design and not hidden implementation details [6, 10].

Figure 2a shows that interestingly, for Adroit, both FERM(VRL3) and DrQv2fD(VRL3) outperform RRL in the early stage, showing that with the right implementation, these off-policy agents can learn quickly even without stage 1 representations. Figure 2b shows that on the most challenging Relocate task, DrQv2fD and FERM variants fail to learn entirely because they do not exploit stage 2 data sufficiently. VRL3 greatly outperforms all other methods.

Significantly stronger SOTA sample efficiency on Adroit: Compared to RRL, VRL3 achieves an average of 780% better sample efficiency over all tasks and is 1220% faster to solve the most difficult Relocate task (in terms of the data required to reach 90% success rate). VRL3+FERM has the same performance as VRL3, showing that additional contrastive learning is not necessary. FERM+S1 is stronger than FERM, showing that stage 1 pretraining also works well with contrastive stage 2 learning. For parameter and computation efficiency, we also use a much smaller encoder compared to RRL. We are 50 times more parameter efficient in the encoder and 3 times more parameter efficient for all networks in the agent. VRL3 is also more computationally efficient, in fact, VRL3 can solve the hardest task with just 10% of the computation (Detailed comparison in Appendix B).

For DMC, we collect 25K data with fully trained DrQv2 agents to enable stage 2 training. FERM(VRL3) and DrQv2fD are both good, and VRL3 is slightly better, all three are better than DrQv2 (results from authors). Note that DMC is typically treated as an online RL benchmark, and DrQv2 is an online method, so we do not claim a new SOTA. These results are only to show that VRL3 can work in this popular benchmark and can utilize offline data to learn faster. Compared to DMC, Adroit is more challenging due to its sparse reward setting and more complex, more realistic visuals with more depth, lighting, and texture information. This might explain the larger performance gap in Adroit. This shows the advantage of VRL3 is more evident in more challenging visual tasks.

5 Challenges, Design Decisions, and Insights

In addition to the very strong performance on Adroit, another major advantage of VRL3 is its simplicity. The minimalist design of VRL3 not only makes it much faster and more lightweight than the previous SOTA but also allows us to easily ablate and study its core components. In this section, we discuss some technical challenges and provide a comprehensive analysis of how the different components of VRL3 work together towards superior performance. We focus on Adroit and all experiment results are averaged over 4 Adroit tasks (per-task plots in Appendix).

5.1 Stage 1: Pretraining with non-RL data

The first challenge we face in stage 1 is input shape mismatch. In stage 1, the input to the encoder is a single image with RGB channels. For the downstream RL task, however, it can be beneficial for the

agent to take in multiple consecutive frames of images as input, or frames from multiple different cameras (sometimes even a combination of both), so that the agent can learn to use temporal and spatial information in these frames for better performance.

A naive solution is to put each frame separately into the pretrained encoder, as is done in [30]. This naive approach has two downsides: 1) it requires more computation, and 2) even if we finetune the encoder, it cannot learn important temporal and spatial features from the interaction of frames.

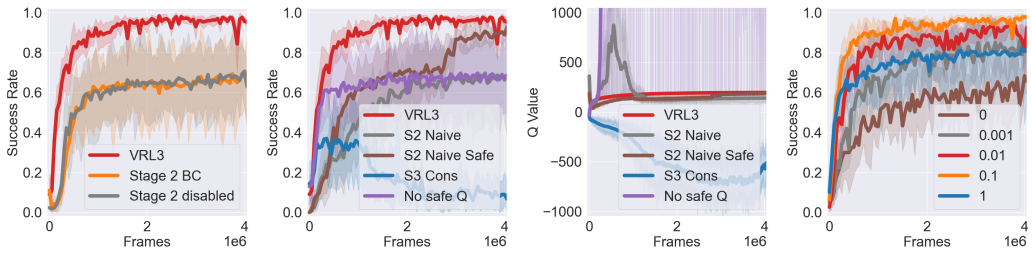
Advantage of convolutional channel expansion (CCE): We propose the CCE technique to address the above issue. Let m be the number of input frames for the RL task, With CCE, we expand the first convolutional layer of the stage 1 encoder so that it can now take in inputs of $3m$ channels instead of 3. We repeat the weight matrix m times and then scale all the weight values to $1/m$ of their original values. This allows us to modify the input shape without disrupting the learned representations. Such a very simple method can effectively solve the mismatch problem when transitioning a visual encoder from the non-RL pretraining stage to the RL stages.

To enable faster experimentation and avoid difficulties in training deep networks in DRL [2, 24], we use a lightweight encoder with 5 convolutional layers, which is 50 times smaller than the ResNet34 used in RRL. Quite surprisingly, VRL3 can achieve superior performance with such a tiny encoder.

5.2 Stage 2 and Stage 3

In stage 2, why use conservative RL updates instead of BC or contrastive representation learning? We use conservative RL for the following reasons: a) we can finetune the encoder to get task-specific representations with RL signal b) comparing with contrastive learning methods which only train the encoder, we can train the encoder as well as the policy and Q networks, giving a jump-start in stage 3. c) With the safe Q technique, this also brings us a minimal-effort design that allows us to transition effectively from stage 2 to stage 3. We can simply remove the conservative term without any elaborate modifications. d) this fits well with an off-policy backbone, allowing maximum data reuse. (BC such as in [27] only works with on-policy methods). Next, we discuss extensive empirical evidence that supports our design.

Figure 3a shows that for VRL3, only using BC on expert demonstrations in stage 2 does not give a good performance. In fact, using BC in stage 2 gives a similar performance to when stage 2 is entirely disabled. This problem is caused by a distribution mismatch of actor and critic at the beginning of stage 3 since BC only trains the actor and not the critic. For contrastive learning, we observe from Figure 2 that FERM has slightly worse performance in DMC and much worse performance in Adroit. This is because Adroit has a more challenging sparse reward setting and FERM fails to fully exploit offline RL data by only updating the encoder. These results show that when tackling highly challenging tasks, **learning only the representation in stage 2 can be useful, but is not enough to fully exploit offline RL data**. This is a novel and important finding that has never been carefully studied or emphasized in the literature.



(a) Stage 2 BC or disable (b) S2-S3 (c) S2-S3 Q value (d) Encoder lr

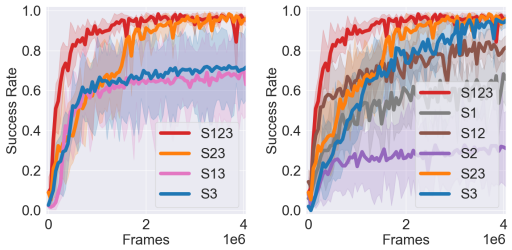
Figure 3: a) Effect of BC and offline RL updates in stage 2. Only applying BC or disabling stage 2 training entirely leads to poor performance. b) and c) Taking naive RL updates in stage 2 (S2 Naive) leads to severe overestimation, which can be mitigated by safe Q (S2 Naive Safe). Taking conservative updates in stage 3 (S3 Cons) leads to underestimation. VRL3 gives the best performance with conservative update only in stage 2 and safe Q. Note in our setting, the maximum reasonable Q value should be around 100. d) Effect of different encoder learning rates (e.g., 0.01 means the encoder's learning rate is 100 times slower than that for the policy and Q networks.)

Effect of safe Q target for stage 2-3 transition: Special care is required to effectively perform off-policy learning in stage 2. In VRL3, we perform conservative updates in stage 2 and remove the conservative term in stage 3. When we transition from stage 2 to stage 3, the new online data will cause a distribution shift that can lead to instability in Q values [23], The safe Q target technique is a very simple, efficient, and minimalist solution to this issue. Figure 3b and 3c show the effect of these design decisions: updating with naive RL in stage 2 (S2 Naive) or removing the safe Q technique (No safe Q) lead to severe overestimation in stage 3 and cause performance drop. Updating with conservative RL in stage 3 (S3 Cons) leads to severe underestimation and even worse performance. Naive RL in stage 2 with safe Q in stages 2 and 3 (S2 Naive Safe) can mitigate overestimation but is still weaker than VRL3, which takes conservative updates in stage 2 and use safe Q in stages 2 and 3.

Reduced encoder learning rate for better finetuning: When we fine-tune the encoder in stages 2 and 3, a reduced encoder learning rate can prevent pretrained representations from getting destroyed by noisy RL signals, especially in the early stage of RL training. This problem has also been observed in other environments [28]. Figure 3d shows the effect of encoder learning rates. These results show that even though there is a large domain gap between ImageNet and RL tasks, a frozen stage 1 encoder can still provide useful features, and finetuning the encoder can further improve performance. Although it is not new that transfer + finetuning can give the best result [41], it has not been adequately studied in the RL setting with large domain gap and noisy signals.

5.3 Contribution of each stage

VRL3 achieves very strong performance by fully exploiting data from 3 different stages. We now try to understand how much each stage contributes to the final performance. We perform 2 sets of comparisons, for each VRL3 variant in Figure 4a, we disable one or more stages of training entirely (e.g. S13 means stage 2 training is removed), and for Figure 4b, we disable the *encoder training* in one or more stages, but always maintain the updates to actor and critic networks (e.g. S12 means encoder is frozen in stage 3).



(a) Stage ablation (b) Encoder ablation

Figure 4: (a) Effect of each training stage. The numbers after "S" indicates the stages that are enabled. (b) Effect of enabling encoder training in each stage.

Here we summarize a number of insights we can obtain from these results. In Figure 4a: Stage 1 pretraining improves sample efficiency (S123 > S23); **stage 2 is critical and its contribution cannot be replaced by the other stages** (S123> S13, S23 > S13). Note that since Adroit is very challenging with limited offline data, S3 is always enabled to obtain non-trivial performance.

In Figure 4b: when the encoder is only trained in one stage, stage 3 encoder training gives a much stronger performance than the other two, showing the importance of task-specific representations (S3 > S1 and S2). Note that very interestingly, **although task-specific features are more useful, when RL data is scarce, non-RL pretraining provides better results despite the large domain gap** (S3> S1 > S2). Training the encoder in more stages always leads to improved performance (S123 > S12 > S1).

An important finding is that our results indicate **offline RL is more beneficial to just contrastive representation learning in stage 2**, note that S3 in Figure 4b (perform offline RL update but freeze encoder in stage 2, encoder trained in stage 3) has much stronger performance than S3 in Figure 4a (no stage 2 training at all). This is consistent with results in Figure 2, the performance gap between VRL3 and FERM+S1 is caused by VRL3 doing offline RL updates in stage 2, and thus learning a full agent, instead of just the encoder features.

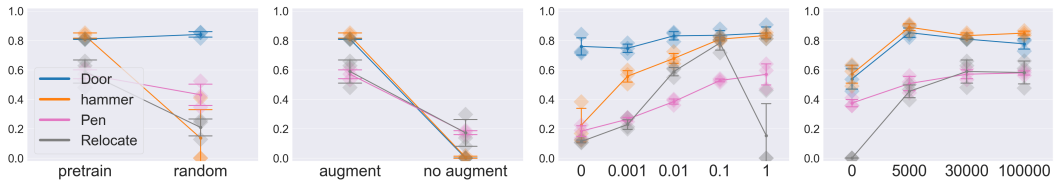
With the above insights, Table 1 summarizes fundamental differences between VRL3 and some other popular visual RL methods.

Table 1: Comparison of core design decisions. VRL3 fully utilizes non-RL, offline RL and online RL datasets. DA refers to data augmentation, Con for contrastive learning, Offline for offline RL. Note that offline RL data can be used to learn representation (encoder), or the task (actor, critic), or both.

Characteristics	DrQ/RAD	CURL/ATC	FERM	RRL	VRL3
Stage 1	None	None	None	Pretrain	Pretrain
Stage 2	None	None	Con	BC	Offline
Stage 3	DA	Con	DA	No DA	DA
Leverage large visual datasets	✗	✗	✗	✓	✓
Task-specific representations	✓	✓	✓	✗	✓
Offline RL data for representations	✗	✗	✓	✗	✓
Offline RL data for task learning	✗	✗	✗	✓	✓
High data reuse rate (off-policy)	✓	✓	✓	✗	✓
Prevent Q network divergence	✗	✗	✗	✗	✓

5.4 Additional Studies and Analysis

We also perform a very extensive study on hyperparameter robustness and sensitivity on each of the 4 Adroit tasks (with varying difficulty). Due to limited space, we only show a small portion of our study in Figure 5 and put the rest of the results in Appendix A. This provides additional insights and helps us understand what hyperparameters should be tuned with high priority when applied to a new task, for example, results show that S1 pretraining is useful for harder tasks but not for the simplest Door task; data augmentation is always critical; the encoder learning rate is important and should be tuned in difficult tasks; even a small number of S2 offline updates can greatly help learning.



(a) S1 pretrain (R) (b) S2,3 augmentation (R) (c) S2,3 encoder lr scale (S) (d) S2 num of update (R)

Figure 5: Part of the hyperparameter sensitivity study. For each figure, X-axis shows different hyperparameter values, Y-axis shows the average success rate in the first 1M training frames, over three seeds. Each dot is the average success rate of one seed, the error bar shows one standard deviation. (S) means critical and relatively sensitive. (R) means critical but is robust or easy to tune.

6 Discussion, Limitations and Future Work

In this paper, we propose a 3-stage data-driven framework for solving challenging visual control tasks. We perform extensive experimentation to ablate and analyze our framework to identify the most important technical components. We arrive at a minimalist design that is very simple, intuitive, and much more sample efficient and computationally efficient than the previous SOTA on Adroit. Although this is a highly significant result, we do have some limitations: for example, Adroit has more realistic visuals compared to DMC, but it is still a simulated benchmark. Experiments on robotics hardware are required to understand how VRL3 should be deployed in the real world. Another limitation is that it remains unclear whether stage 1 training is still beneficial if the domain gap between non-RL data and RL data is too great (RL task visuals very different from the real world). These can be exciting future research topics. Despite these limitations, VRL3 brings a number of novel and interesting insights, in addition to the very strong empirical results. We believe VRL3 is an important step toward practical data-driven DRL for real world visual control tasks.

References

[1] Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhihong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic bootstrapping for uncertainty-driven offline reinforcement learning. *arXiv preprint arXiv:2202.11566*, 2022.

[2] Johan Bjorck, Carla P Gomes, and Kilian Q Weinberger. Towards deeper deep reinforcement learning. *arXiv preprint arXiv:2106.01151*, 2021.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18353–18363, 2020.

[5] Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and Pieter Abbeel. Deep spatial autoencoders for visuomotor learning. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 512–519. IEEE, 2016.

[6] Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. *arXiv preprint arXiv:2106.06860*, 2021.

[7] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[8] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl. In *International Conference on Machine Learning*, pages 3682–3691. PMLR, 2021.

[9] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018.

[10] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[11] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *arXiv preprint arXiv:1708.04133*, 2017.

[12] Ryan Julian, Benjamin Swanson, Gaurav S Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning. *arXiv preprint arXiv:2004.10190*, 2020.

[13] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021.

[14] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[15] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

[16] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[17] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020.

[18] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[19] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, 2022.

[20] Sergey Levine. Understanding the world through action. In *Conference on Robot Learning*, pages 1752–1757. PMLR, 2022.

[21] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[22] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[23] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets. 2020.

[24] Kei Ota, Devesh K Jha, and Asako Kanezaki. Training larger networks for deep reinforcement learning. *arXiv preprint arXiv:2102.07920*, 2021.

[25] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[26] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[27] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[28] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, Devon Hjelm, Philip Bachman, and Aaron Courville. Pretraining representations for data-efficient reinforcement learning. *arXiv preprint arXiv:2106.04799*, 2021.

[29] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

[30] Rutav Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. In *Self-Supervision for Reinforcement Learning Workshop-ICLR 2021*, 2021.

[31] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. *arXiv preprint arXiv:2009.08319*, 2020.

[32] Richard S Sutton. The bitter lesson. `http://www.incompleteideas.net/IncIdeas/BitterLesson.html`, 2021. Accessed: 2022-01-24.

[33] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[34] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012.

[35] Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. In *Advances in Neural Information Processing Systems*, pages 6288–6297, 2018.

[36] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[37] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. *arXiv preprint arXiv:2105.08140*, 2021.

[38] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

[39] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[40] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021.

[41] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

[42] Albert Zhan, Philip Zhao, Lerrel Pinto, Pieter Abbeel, and Michael Laskin. A framework for efficient robotic manipulation. *arXiv preprint arXiv:2012.07975*, 2020.

[43] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

[44] Han Zheng, Xufang Luo, Xuan Song, Dongsheng Li, Jing Jiang, et al. Adaptive q-learning for interaction-limited reinforcement learning. 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] A discussion is provided in section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] A discussion on potential negative societal impacts is provided in Appendix D.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide source code in the supplementary materials.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We discuss major technical details in the main paper, with additional technical details in Appendix A and C.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] The plots are averaged over multiple runs (some are also over multiple environments), shaded area indicate the standard deviation across runs.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] They are discussed in detail in Appendix B.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cited the authors of the code (DrQv2) and testing environments (DMC, Adroit, MuJoCo) we use.
   (b) Did you mention the license of the assets? [Yes] A discussion is provided in Appendix C.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our source code is provided in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]