Extremely Lossy Compression through Reinforcement Learning

Jessica Bader, 5624582; Philipp Noel von Bachmann, 4116220

August 31, 2021

1 Introduction

Image compression exists in multiple forms. Lossless compression mandates original data be perfectly reconstructed, but is limited by data entropy. To compress further, information content must be sacrificed. Therefore, lossy compression minimizes the loss for a specific bitrate. But which content should be discarded? Early image compression loss-functions, like L_2-loss, give each pixel equal value. However, some parts retain more valuable information than others. Research has hand-crafted improved metrics, like MS-SSIM [9], but defining these is task-specific. Ultimately, achieving the most extreme forms of compression requires task-dependent methods to quantify information value, preserving only the minimum required to complete the task. However, general methods for this process would be useful. This facilitates a natural transition into the reinforcement learning (RL) space, where agents already complete tasks by acting on an environment to maximize rewards. The goal of this research is to define a method through which data is compressed such that the agent can maximize the reward in the reconstructed environment.

2 Related Works

Balle et. al's work in modifying neural networks (NNs) (variational autoencoders) for data compression provided the inspiration for modifications to the RL agent for data compression [3].
Furthermore, a variety of work explores using RL agents in data compression, including pruning for image compression [1] [6] [11], optimal codebook mapping [10], NN architecture compression [2], sentence compression [7], and state compression for RL agents [5]. However, to the best of our knowledge, no previous research exists in the task-dependent image compression domain.

3 Methods

3.1 Architecture

In Deep-Learning based compression, an encoder NN embeds data in a latent space, where it is losslessly encoded and transmitted. The decoder NN decompresses the latent space and returns to the input domain.

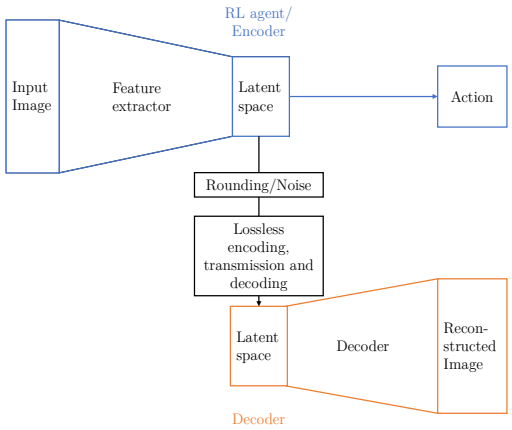

Figure 1: Overall architecture

In our approach, a RL agent acts as the first NN. The agent contains a feature extractor, which extracts information to determine the action. Ideally, this content should also be important for reconstruction. Then the latent space is losslessly encoded and transmitted with an algorithm of choice, such as ANS (for training this step is unnecessary as the latents will be losslessly decompressed). After decoding, the latents are passed through a sim-

ple upsampling NN. Figure 1 shows an overview of the architecture.

3.2 Theory

Lossy compression balances a low bitrate and a low distortion. This section describes how these objectives apply to our project.

3.2.1 Bitrate

Determining algorithm bitrate requires observing the encoding process, where the input image is mapped to the latent space (see 3.1). This decreases the image bitrate if the latent entropy is lower than the image entropy and a sufficient encoding distribution for latent transmission is chosen. If the latent space distribution of z is given by $Q_\theta(Z = z|x)$ where θ parametrizes the distribution and $P(Z = z)$ is used to encode the latents for transmission, the bitrate is given by the cross entropy

$$\mathbb{E}_{z\sim Q_\theta(Z=z|x)}[log(P(Z = z))] \tag{1}$$

In addition, replacing P by Q, achieves the perfect encoding distribution. Hence, the entropy is a lower bound on the cross-entropy.
Because NNs have real valued outputs, $Q_\theta(Z = z|x)$ is a continuous PDF, resulting in a high bit rate. Therefore, one approach is to round the latents [3]. This reduces Q to a discrete probability distribution, which reduces the cross entropy. However, rounding is undesirable during training, since its gradient is 0 almost everywhere. Therefore, we replace rounding during training with adding uniform noise u (see also figure 1). This shifts the values similarly to rounding, but behaves better.
Another effect of adding uniform noise is that we can reparametrize equation 1 and take the expectation over the uniform noise:

$$\mathbb{E}_{u\sim U[-\epsilon,\epsilon]}[log(P(Z = \hat{z} + u))] \tag{2}$$

where $\hat{z}$ is the mean of $Q_\theta(Z = z|x)$.

The encoding distribution P is naively chosen as Normal. The mean can be chosen arbitrarily given a powerful enough transformation, so it is set to 0 for simplicity. The variance is learned during the training process.

3.2.2 Distortion

Bitrate decrease is often traded for distortion increase. Therefore, we need a performance measure to evaluate reconstructed image distortion $\hat{x}$. A traditional metric is the Mean Squared Error (MSE),

$$\mathbb{E}_{x,\hat{x}}[\sum_{i,j}(x_{ij} - \hat{x}_{ij})^2] \tag{3}$$

Early experimentation (see section 4.1) found that some task-important details, specifically ball location, were omitted as they accumulated little MSE penalty; attempting to recover these aspects, we devised a second loss scheme, referred to as latent loss. The motivation was to reward the decoder for reconstructing the image such that the same features would be extracted, resulting in important aspect preservation. For this loss, we passed the reconstructed image through the feature extractor and evaluated the MSE between the original and reconstructed latent space.

3.3 Training

3.3.1 Process

RL-agents normally train with a specific loss function, which will be abstracted by $Loss_{RL}$. As the RL-agent acts as the encoder, it is responsible for the bitrate. Therefore we add the loss from 1 to the standard RL loss:

$$\mathbb{E}_x[Loss_{RL}(x)+\alpha\cdot\mathbb{E}_{u\sim U[-\epsilon,\epsilon]}[log(P(Z = \hat{z}+u))]] \tag{4}$$

where $\alpha \in \mathbb{R}$ is chosen to balance the two terms. Both expectations will be approximated by empirical means over the training data. As the RL-agent is independent of the decoder, we can also train it separately first.
After training the RL-agent, we fix its values. Therefore, at this point the encoding to the latents is fixed. Now the decoder is trained. For the decoder, we use the Loss schemes given in section 3.2.2.

3.3.2 Adaptive alpha (omitted)

During training, an issue arose in choosing α (see 4.5) possibly caused by the RL reward: the agent behaves randomly in early iterations and earns little reward, but improves drastically once learning starts. Therefore, choosing α too large prevents the agent from ever learning to complete the task, while choosing α too small allows the agent to ignore the cross entropy loss during action fine-tuning. We tried to resolve this with a non-static α: as the reward grows, the relative importance of the two terms should stay approximately the same. We designed several adaptation schemes (others can be found in Future Work), but the general chosen algorithm was to update α each time the ratio threshold was violated. We implemented this by

$$\alpha = \alpha \cdot 10 \cdot e^{-i/1000} \tag{5}$$

where i is the number of iterations. Multiplying by the negative iterations ensures that α changes drastically at the beginning, but converges over time.

4 Experiments and Evaluation

4.1 Environment

We used the game "Breakout" from the Gym toolkit [4] as an environment. Breakout is a game in which the player controls a paddle to bounce the ball and destroy blocks. The player wins if no block is left, but loses if the ball is dropped.
The RL agent learning and playing the game was Proximal Policy Optimization (PPO) [8]. PPO is an on-policy algorithm, that uses an actor critic approach. We started with the stablebaselines3 implementation [8], and modified it to our needs.

4.2 Dimension reduction

Our first experiments featured the basic dimension reduction method, without the modifications for compression. The goal was to verify the required reconstruction information was present within the feature extractor's lower dimensional representation. The qualitative results can be seen in Figure 2 (Note: images came from training set).

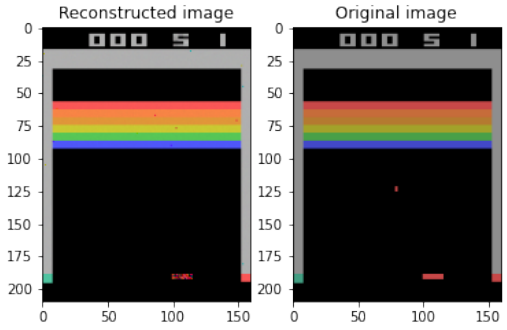

Figure 2: Baseline method (no compression) with decoder trained on MSE for 10,000 iterations

4.3 Compression

Now we added the compression loss to the RL agent training, equation 4 and trained the agent with this loss function. We modified the α value, looking at the performance of the decoder after some epochs. An α of $1e-4$ worked best, therefore we chose this value for an extended run over $1e5$ epochs. Tested on an independently created test dataset, this resulted in an entropy of 1.7 bits per latent dimension, when the latent values were rounded to 3 digits. The actual bitrate given by the cross-entropy 1 is higher however, as we must choose a fixed latent encoding and transmission distribution. For training we chose a Normal distribution with mean 0 and learned variance, also used for testing. The expected variance per dimension was estimated by the mean per dimension over the whole test dataset. However, the test cross-entropy was significantly higher, mostly in the order of at least times 1000. A visual investigation of the latent values showed that while the entropy of the latent values was low, some values deviated from 0 significantly in the order of 100, leading to a probability of nearly 0.

After training the RL Agent, the decoder was trained for $2e5$ epochs. This resulted in an MSE Loss of 1895. As discussed in 1, MSE is an unideal metric for performance, since it gives each pixel the same value. Therefore we investigated some images qualitatively. An example can be seen in figure 3.

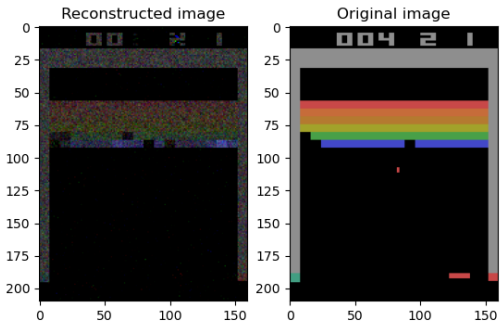

Figure 3: Final agent

4.4 Latent Loss Scheme

As discussed in section 1, L_2 Loss may be suboptimal and we introduced a latent loss scheme 3.2.2 as an alternative. Training a decoder with the latent loss scheme showed an image quality decrease compared to L_2 Loss, (see figure 4) (also from the training set).

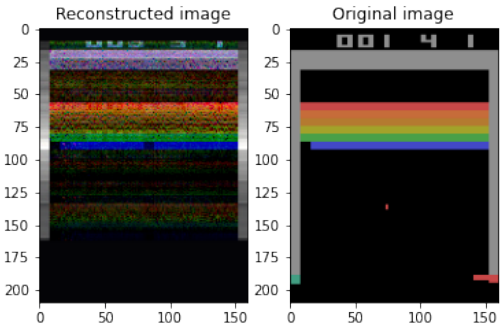

Figure 4: Baseline method (no compression) with decoder trained on MSE for 10,000 iterations, then latent MSE for 10,000 iterations

4.5 Adaptive Alpha

When implementing our custom loss function with static α, several rudimentary tests were performed to verify the model behaved as expected. One such test involved varying α: we expected that a lower value would prioritize task performance, while a higher value would prefer a lower bitrate. However, we found this was not the case: there seemed to be just as much variation between independent tests of the same α as changing α. Given our hypothesis for where the problem lay (see 3.3.2), we developed the adaptive α scheme. However, initial results showed no improvement and this was moved to Future Work.

4.6 Pretrained agents

Another idea was that adding the compression loss to the RL agent from the beginning on was too large of a constraint for the agent to learn anything. Therefore, we tried to fix this by pretraining the agent before adding the compression loss to the agent. On the one hand, the additional compression loss reduced the entropy but on the other hand, the decoder did not learn after the training, so this procedure showed no improvements.

5 Discussion/Future Work

As can be seen in Figure 1, our baseline with the image MSE produced encouraging results: the location of the paddle was consistently preserved. However, areas of improvement include the artifacts that exist in this paddle along with the loss of the ball.

5.1 Bitrate inconsistency

When adding the custom loss function to the RL-agent, we saw that we could not verify the desired influence. Increasing α did not necessarily decrease entropy, but entropy fluctuated strongly between runs. One reason could be that the RL-agent loss function is quite unstable in itself. While in some epochs, the agent loss was relatively high, it was fairly small in other epochs. Therefore, we adjusted α dynamically to achieve to a desired bitrate instead of using a fixed α. However, our adaptive α scheme was largely ineffective in verifying the expected behavior. Therefore, modifications should be made so both task accuracy and bitrate increase inversely with α, or a theoretical foundation should found why they do not. This could be possible through other adaptive schemes (thresholds, static ratios, adaptive ratios, etc.). This may also require reflection to analyze if this divergence from expected behavior is driven by mechanisms other than RL loss.

5.2 Bitrate evaluation

On the test set, the entropy of the final agent 4.3 is reasonably small and the image is compressed. However, if in testing we encode the latents with a Normal distribution as during the training process, the bitrate would rise. This means that while the agents is able to compress the data, it is not able to fit the distribution. Therefore future work needs to either fit the agent better to the data or find a new method to chose/estimate the latent encoding distribution.

5.3 Decoder generalization

On the decoder side, we saw that the image quality decreased in comparison to the baseline, especially on the test dataset. Although a decrease can be expected due to compression, the decoder failed to reconstruct important properties like the paddle position, and just reproduced static properties like the game frame. While we cannot yet rule out insufficient training time (due to lack of resources), we believe this comes from lack of information in the encoding. As the encoding is designed to allow the agent to act, it may not be injective; hence, many different game states could have the same representation. A new architecture is required, for example: using an earlier layer of the feature extractor as the encoding; training the encoder and decoder together; using a different encoder altogether.

5.4 Latent loss scheme

Changing the decoder loss function to our latent loss scheme showed no improvement. The image becomes noisy and information is distorted. Even in the lower part of the image where normally the screen is black, the decoder introduces noise. One reason for this behaviour could be similar to adversarial attacks: the decoder tricks the encoder into generating the desired latents from totally different input images. One improvement could be to combine the latent loss scheme with L_2-Loss such that L_2-Loss preserves the general image quality and penalizes distortion, and latent loss preserves semantic information.

5.5 Environment

Furthermore, we would like to explore other environments. Because this environment included a static start position and relatively slow image change, we found that many of the images looked similar (full or almost full block pattern) which lead to overfitting and consqeuently poor generalization. Using the agent long enough to train and collect diverse images proved infeasible under time/resource constraints.

5.6 Train test separatation

In addition, a unique issue with train/test separation resulted from the cross-domain approach. Clear separation is an industry standard for data compression. However, RL agents train freely, and therefore it is not possible to make guarantees in advance about training images. Our solution was to generate the test dataset independently from our agent, so that the probability of the same image appearing in both testing and training was low. Evaluation of this probability would need to be done before this could be a solution, but is likely intractable as state probabilities are unequal. A better solution would be to generate a large test set in advance and remove images if they appear during training. It can also be noted that full, final results should be reported on a test set rather than a training set (as done in the section 4.2 of experiments). These were done as rudimentary sanity checks, and were included due to lack of better results.

6 Contact information

For further questions about this work (including code, omitted results, etc), please contact the authors at jessica.bader@student.uni-tuebingen.de.

References

[1] M. Alwani, V. Madhavan, and Y. Wang. DECORE: deep compression with reinforcement learning. *CoRR*, abs/2106.06091, 2021.

[2] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani. N2N learning: Network to network compression via policy gradient reinforcement learning. *CoRR*, abs/1709.06030, 2017.

[3] J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimized image compression. *CoRR*, abs/1611.01704, 2016.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] S. Guo, Q. Yan, X. Su, X. Hu, and F. Chen. State-temporal compression in reinforcement learning with the reward-restricted geodesic metric. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[6] R. Khatri. "kernel-controlled dqn based cnn pruning for model compression and acceleration". *Electronic Theses and Dissertations*, 4105, 2021.

[7] W. Liangguo, J. Jiang, and L. Liao. *Sentence Compression with Reinforcement Learning: 11th International Conference, KSEM 2018, Changchun, China, August 17–19, 2018, Proceedings, Part I*, pages 3–15. 08 2018.

[8] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann. Stable baselines3. *GitHub repository*, 2019.

[9] Z. Wang, E. Simoncelli, and A. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003.

[10] X. Yuan, L. Ren, J. Lu, and J. Zhou. Enhanced bayesian compression via deep reinforcement learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6939–6948, 2019.

[11] H. Zhan and Y. Cao. Deep model compression via deep reinforcement learning. *CoRR*, abs/1912.02254, 2019.