Learning Task-Relevant Representations with Selective Contrast for Reinforcement Learning in a Real-World Application

Flemming Brieger [1] **Daniel A. Braun** [2] **Sascha Lange** [1]

Abstract

We use contrastive learning to obtain task-relevant state-representations from images for reinforcement learning in a real-world system. To test the quality of the representations, an agent is trained with reinforcement learning in the Neuro-Slot-Car environment (Kietzmann & Riedmiller, 2009; Lange et al., 2012). In our experiments, we restrict the distribution from which samples are drawn for comparison in the contrastive loss. Our results show, that the choice of sampling distribution for negative samples is essential to allow task-relevant features to be represented in the presence of more prevalent, but irrelevant features. This adds to recent research on feature suppression and feature invariance in contrastive representation learning. With the training of the reinforcement learning agent, we present to our knowledge a first approach of using contrastive learning of state-representations for control in a real-world environment, using only images from one static camera.

1. Introduction

We propose a system for real-world reinforcement learning (RL) with contrastive learning of task-relevant state-representations. Our approach builds upon the works of Kietzmann & Riedmiller (2009) and Lange et al. (2012), who introduced a slot-car racer as an environment for real-world reinforcement learning (RL) and demonstrated the viability of using an auto-encoder to learn low-dimensional state-representations from images of that environment. We make several updates and improvements to this approach on the basis of recent advancements in the field of self-supervised learning.

Learning based on contrasting representations of samples against each other has proven to be a powerful paradigm for self-supervised pre-training of artificial neural networks for downstream tasks such as classification (Oord et al., 2018; Chen et al., 2020a) or control (Srinivas et al., 2020; Mazoure et al., 2020). Unlike earlier self-supervised approaches, these methods avoid reconstruction of partial or complete inputs making it significantly more parameter-efficient. Moreover, the choice of which inputs are contrasted is highly influential on the learned representations and consequently their usefulness for a given downstream task (Chen & Li, 2020).

On this basis, we update the "Deep-Fitted-Q" system introduced by Lange (2010) and Lange et al. (2012): First, we use a contrastive representation learning paradigm - contrastive predictive coding (CPC) (Oord et al., 2018) - instead of an auto-encoder to learn state-representations for the visual RL-task, thereby eliminating the need for a parameter-expensive decoder for image reconstruction. Additionally, the CPC-model is trained to learn state-representations from image-sequences instead of single images, by which the dynamics of the environment are integrated at the representation learning stage. Second, using this contrastive learning paradigm, we demonstrate how contrastive learning can be used to focus on representation of task-relevant features in the presence of more prevalent, yet irrelevant features that can suppress the expression of the former. This is achieved independently from reinforcement learning by restricting the distribution from which negative samples are drawn for the contrastive objective to samples with similar lighting conditions.

2. Background

A range of self-supervised frameworks for learning representation from images and video have been proposed in recent years (Zhang et al., 2016; Vondrick et al., 2018; Qian et al., 2020; Misra et al., 2016; He et al., 2020; Le-Khac et al., 2020) with varying pretext tasks, that determine which features are represented in the resulting latent space (Doersch & Zisserman, 2017). In order to produce more general representations for various data-domains, contrastive predictive coding (Oord et al., 2018; Henaff, 2020) was developed, which is based on classifying related parts of one sample

[1]Psiori GmbH, Freiburg, Germany [2]Institute of Neural Information Processing, Ulm University, Ulm, Germany. Correspondence to: Flemming Brieger <flemming@psiori.com>.

The 2021 ICML Workshop on Unsupervised Reinforcement Learning.

(e.g. first and second part of a sequence) versus negative samples from one data-set using a loss based on noise contrastive estimation. This classification pretext task is entirely formulated in the latent space learned by the neural network and as such does not require any reconstruction of the input. Other approaches have used data augmentations to optimize agreement not between different parts of an input, but rather augmented versions of one input (Chen et al., 2020a;b) and noted, that the choice of what inputs are contrasted is a significant influence on the structure of the learned representation (Xiao et al., 2020). Chen & Li (2020) have since systematically demonstrated that the suppression of certain features by the presence or emphasis (e.g. through augmentations) of others is a demonstrably important influence on the learned representation and its usefulness for downstream tasks. Moreover, this is a distinctive property of contrastive compared to reconstruction based methods.

The viability of visual reinforcement learning for control of an exemplary real-world system was previously demonstrated in the works of Lange (2010) and Lange et al. (2012) for control of a slot-car racer. The authors proposed to tackle the problem with a two-stage system of an auto-encoder that learns to encode high-dimensional observations (images) of the environment into low-dimensional state-representations and using these representations to train an agent. The resulting system successfully solved the real-world RL-task and in doing so avoided common problems of an end-to-end RL-approach such as the need for vast amounts of experiences to learn the objective from high-dimensional inputs.
More recently, contrastive objectives have been applied as an auxiliary loss to support learning state-representations jointly with a visual RL-task (Oord et al., 2018; Srinivas et al., 2020; Mazoure et al., 2020). However, having an already trained feature extractor for state-representations, avoids having to collect training samples for learning representations during the potentially costly RL-phase. Training data for representation learning is generally easier to come by, e.g. from historic data, than the training experiences needed for RL, when applying RL for control in real-world systems. Moreover, while the aforementioned approaches have yielded state-of-the-art performance on several benchmarks, these benchmarks are based on control in a simulation, not a real-world system, which would potentially present additional challenges that are not necessarily accounted for in simulations (see Dulac-Arnold et al. (2019)).

3. Methods

3.1. The Slot-Car Environment

The slot-car environment follows the structure introduced by Kietzmann & Riedmiller (2009) and Lange et al. (2012): A downwards-facing camera recorded images of a car driving on a slot-car racing track from above. We record with 5 frames per second with an image-resolution of 202×76. Unlike the setup by Lange et al. (2012), the track is arranged in an 8-like shape with a bridge over the middle section, where the track would intersect itself (Figure 1), thereby occluding the car in a small section of the track. Only the right one of the two car slots was used.
The agent has three available actions corresponding to the voltage applied to the track and consequently the car's speed. The goal is for the agent to learn to drive as fast as possible without getting flung off the track by driving too fast in curves. To this end, costs are attributed to each state based on the voltage (the chosen action) applied. Lower costs are attributed to higher voltages, encouraging high speeds. However, the relation between voltage and speed is non-linear. The available actions and their corresponding costs are:

- 20 to drive at high speed (cost: 0).
- 15 to drive at moderate speed (cost: 0.005).
- 10 to drive at low speed (cost: 0.01).

A terminal state is defined for instances when the car has lost connection to the track and is punished with comparably high cost of 1. While it is not possible to drive steadily with 20 without crashing, the slower actions could be applied consistently without risk. This forces the agent to learn to choose between the available actions according to the car's position on the track. To further highlight this optimal behaviour, a guardrail was installed on the left, wider curve of the track to allow the agent to drive through this curve at high speed, while having to slow down for the right curve. There is no braking action included in our setup, since friction alone slows down the car considerably when switching to lower voltages. From a perspective of optimal control, this task is particularly interesting, because the optimal policy will always be close to a terminal state.

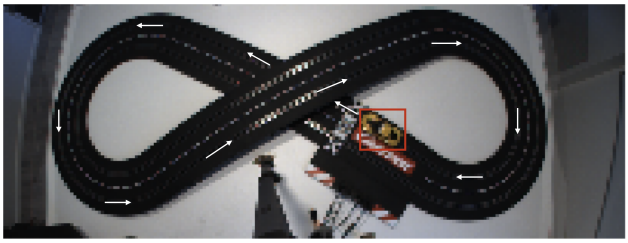


Figure 1. Example image from the camera above the slot-car track. The car (red square) is positioned at the starting line. White arrows indicate the car's driving direction and track for all experiments in this work. On this (right) track the car drives a wider bend on the left side than the right.

3.2. CPC-Model

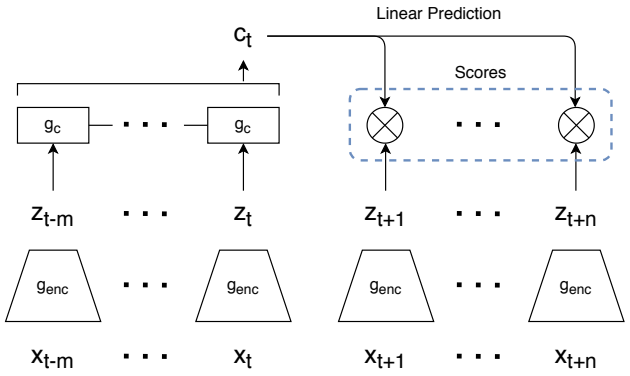


Figure 2. Illustration of the operations involved in the CPC-model, adapted from Oord et al. (2018). Each element of an input-sequence x is passed through the encoder network g_{enc} separately to produce a sequence of latent representations z. The first part of this sequence $z_{\leq t}$ is used as input to the recurrent context-network g_c to produce a context embedding c_t that "summarizes" the information from $z_{\leq t}$. Based on c_t a linear prediction is made for each z_t and scores are calculated via the dot-product between predictions and $z_{>t}$. The InfoNCE loss (Oord et al., 2018) is optimized to identify positive samples against a set of negative samples in a training batch.

In the original paper proposing CPC (Oord et al., 2018) the authors do not include an application for sequential images or video, but handle single images as a sequence of patches. Since the data for this project consists of sequential images, we instead pass single images to the encoder g_{enc}, such that there is one representation z_t per image x_t in the original sequence. Afterwards, the procedure is unaltered (see Figure 2).

CPC was originally applied to much larger and more complex problems w.r.t. the number of training samples as well as the diversity of the input. As the problem at hand is conceptually less complex, we defined a less powerful network topology to avoid overfitting.

The encoder is made up of ten convolutional layers followed by three dense layers. The convolutional layers can be divided into three blocks with two layers of 3×3 filters followed by one layer of 1×1-dimensional filters with one extra layer of 1×1 filters before the fully-connected layers. Every second 3×3 convolutional layer applies a strided convolution with stride 2. The final output of the convolutional part is a feature vector of 528 dimensions. The subsequent dense layers have 128, 64 and 3 units. The context network is a GRU (Cho et al., 2014) with 3 output units, and the prediction is done with one dense layer per target with 3 units and no activation or bias, i.e. a linear prediction. Unless otherwise noted, we use SELU activation functions (Klambauer et al., 2017) in all convolutional and dense layers except the final layer of the encoder, which has no activation. For the GRU, we omitted the $tanh$-activation as this produced better results in preliminary experiments. Both the encoder-representation z_t and the context-representation c_t are thereby three-dimensional, which is based on the assumption that there are three relevant features of the input-sequences to represent: Two dimensions for the position of the car on the track (e.g. x- and y-coordinates) and another for the speed of the car. The latter can however only be represented by c_t by considering sequences of images. The third dimension of z_t is instead intended as an "auxiliary" dimension to avoid local minima.

3.3. CPC Training

Selective Contrast A problem not considered in the auto-encoder based approach to the real-world slot-car environment of Lange et al. (2012), is that of changing environmental conditions (e.g. lighting, see (Lange, 2010), p. 196) that are apparent and even prevalent in the input-images but irrelevant to the RL-task. For instance, in our setup, the lighting of the track varies considerably with time-of-day due to the sunlight coming in through windows or different lamps being switched on or off. In order to account for this in the data used for representation learning, we record data from five different conditions (*ML*, *MN*, *NL*, *E1*, *E2*), for which the most prominent property between conditions is the lighting of the track (see Figure 3). This property is functionally irrelevant to the representation the model is expected to learn, which should be focused on the position and speed of the car. Moreover, invariance to the lighting in the image is even preferred, such that an RL-agent trained under one condition could be evaluated under another with little to no loss of performance.

When presenting the network with a sample from one condition as the target and contrasting it with negative samples from other lighting conditions, the lighting will be the major distinguishing feature between the target and the negative samples in the contrastive loss. During training, the model will therefore initially learn to distinguish samples by lighting condition not by the position of the car. To avert this, the complete training set could comprise only samples from the same condition, such that lighting differences are not present at all. However, diversity of the training data generally improves generalisation, as evident from the widespread use of data augmentation in deep learning (Shorten & Khoshgoftaar, 2019). We therefore hypothesize, that a network trained on data with only one lighting condition will perform poorly on data from other conditions. In order to use multiple conditions in a training set and still focus the representation on relevant features, we therefore sample training-batches in such a way that all samples in one batch stem from the same condition. Since the contrastive samples for a given sample in the CPC-paradigm are simply all other samples in the batch, this ensures that negatives do

not predominantly differ from the target w.r.t. the lighting condition. In the following, we will refer to this method as selective contrast, as it is selective in which data/feature expressions a target sample is contrasted with. This practice has previously been applied for speech recognition (Oord et al., 2018; Wang et al., 2020): A CPC-model trained with negative samples (audio-snippets) recorded by the same speaker as the target sample achieved higher performance for phone-classification than when trained with negative samples from mixed speakers.

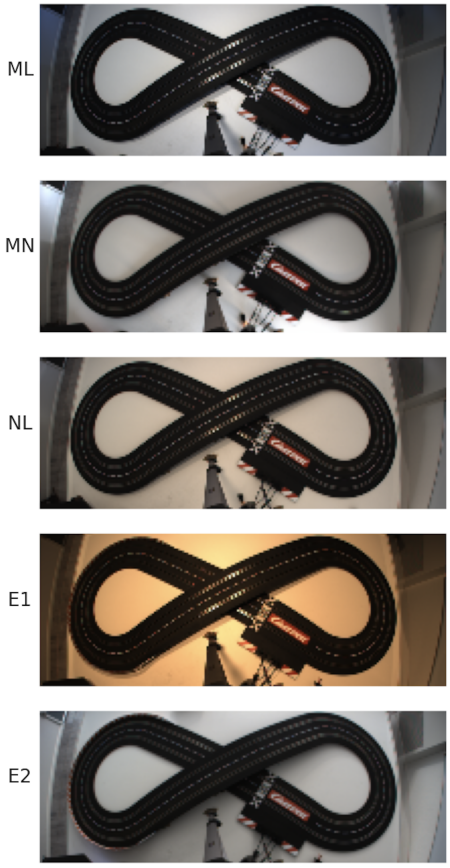


Figure 3. Visualization of the five recording conditions. Recorded at morning with overhead lights on (*ML*), morning without lights (*MN*), noon with lights (*NL*), evening with overhead lights (*E1*), evening with light from the side (*E2*). Conditions *E1* and *E2* have a guardrail installed in the left bend.

For training of the CPC-models we record a sequence of $4,000$ images per lighting condition of the car driving on the track, with step-wise increasing voltage in the range of safe actions. For each of the five training data-sets, we record a test-set of $1,000$ images under the same condition and with the same distribution of voltages. For training, we transform these data-sets into $3,985$ sub-sequences (985 for test-sets) of length 16 to use as input to the model. The image-sequence is split in the middle into 8 targets and 8 as input to the context-model g_c

Using this training-data, we train models with 3 different setups:

L: Training only on data from one lighting condition.

NSC: Training on data from four of the five conditions without selective contrast. Batches can include images from all conditions.

SC: Training on data from four of the five conditions with selective contrast. Batches only include images from the same condition.

By leaving out at least one of the five lighting conditions in each setup, we can use the remaining condition to assess how well the model is able to generalize to the unseen condition(s). In each setup, we use ~$4,000$, from which 25% of samples from the same recording(s) are used for validation after each epoch. Lighting conditions are evenly distributed in all *SC*- and *NSC*-sets. The images (pixel values in range $[0, 1]$) are passed to the encoder without further pre-processing, except applying random noise from a Gaussian distribution ($\mu = 0, \sigma = 0.02$) for regularization. Each model is trained with a batch-size of 16 (i.e. 15 contrastive samples) for a maximum of 300 epochs, stopping the training earlier if the loss on the validation data has not decreased for 20 epochs. We use Adam optimization (Kingma & Ba, 2014) with gradients clipped to a maximum norm of 0.01 as suggested by (Henaff, 2020) and an initial learning rate of 1×10^{-4} which is decreased by a factor of 0.9 if the validation loss has not decreased for five epochs to a minimum of 1×10^{-5}.

3.4. RL-Model

In order to validate the usefulness of the learned representation in a downstream task, we use neural fitted Q-Iteration (Riedmiller, 2005) to solve the previously established RL-task in the slot-car environment (Kietzmann & Riedmiller, 2009). The neural network of the NFQ-agent receives a 6-dimensional vector as input: the 3-dimensional output from the CPC-encoder of the current observation and its predecessor, in order to provide information on not only the car's position, but its velocity as well. This 6-dimensional input is passed through two fully-connected layers of 12 $tanh$-units and a third output layer of 3 $sigmoid$-units, one unit for each action. Since the problem is framed as minimizing costs instead of maximizing rewards, the output unit with the lowest activation indicates the chosen action.

We collect episodes of data for training the agent from interaction with the environment. One episode entails the car driving around the track for a maximum of 100 steps or less if the car loses connection to the track. For the first 20 episodes, actions are chosen uniformly at random in order to

build up an initial set of experience-tuples. Afterwards, we use an ϵ-greedy strategy, where the probability of choosing an action at random is initially set to $\epsilon = 0.8$ and decreased by 0.05 after each episode to a minimum of 0.1. After each episode, all experiences collected up to this point are used to train the neural network for 200 epochs with *RMSProp* optimization (Ruder, 2016) and a learning rate of 0.001 on batches of size 512. Inputs to the network are normalized to minimum and maximum over the complete set of experiences. We set the discount factor to $\gamma = 0.1$. Before starting the next episode we evaluate the performance of the agent with an additional episode of 100 steps without randomly chosen actions. Experiences from the evaluation episodes are not added to the training data. The complete process is repeated for a total of 70 training episodes.

4. Results

4.1. Representation Learning

In order to evaluate the performance of the three sampling setups previously discussed - without selective contrast (*NSC*), with selective contrast (*SC*) and training only on data from a single lighting condition (*L*) - we report the performance of one model per five possible combinations of lighting conditions and setup (always using 4 out of 5 conditions for *NSC*/SC). As a result, we have 5 models for each setup, a total of 15 models.

Model Performance An overview of the models' performance is depicted in Figure 4. The (contrastive) accuracy as reported here refers to the proportion of samples in one batch of 16 for which the model's mean score over $x_{>t}$ was highest for the correct sample against the other 15 samples in the batch. We use this metric over the actual loss for interpretability (Oord et al., 2018). In each sampling setup, the models' accuracy reaches an accuracy on the validation-set above 0.40, at least 6 times higher than the random accuracy of $\frac{1}{16} = 0.0625$. In all cases, accuracy on the test and generalisation sets is lower. However, for training with selective contrast, this discrepancy is notably smaller than for the other setups. For both *L* and *NSC*, some models fail to generalize on the generalisation set completely.

Quality of Representations To gain a deeper insight into the quality of the learned representations, we report the results of an analysis inspecting how the relevant features for the downstream RL-task are predictable from the embedding learned by the encoder. Our focus lies on the encoder-embedding z as only the encoder is used in the RL-task.
As exemplified in Figure 5, for the *L*- and *SC*-model, the embedding is arranged in an elliptical structure and the car's y-coordinate in the image can be clearly perceived, while differences between lighting conditions for the *SC*-model

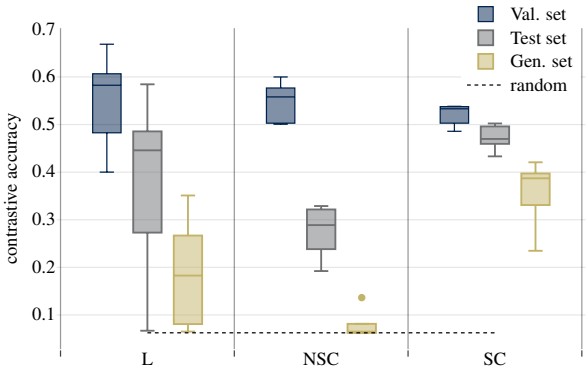


Figure 4. Contrastive accuracy for each setup on the respective sets: Validation set (same recording, different sequences), test set (different recording, same lighting), generalisation set (different recording and lighting). Each box is based on 5 models, whiskers extend to min/max values. The dotted line indicates accuracy for random guessing.

Table 1. Coefficient of determination (R^2) of regressing latent z on position of car in x- (upper) and y-position (lower) with support vector regression.

		μ	σ	min	max
x-coordinate	*L*	0.94	0.08	0.79	0.99
	NSC	0.00	0.00	0.00	0.00
	SC	**0.98**	0.01	0.96	0.99
y-coordinate	*L*	0.78	0.44	-0.10	1.00
	NSC	-0.01	0.02	-0.01	0.06
	SC	**1.00**	0.00	0.99	1.00

Table 2. Mean accuracy of support vector classifiers classifying the correct out of all 5 lighting conditions (random baseline: $\frac{1}{5} = 0.2$) from encoder outputs z.

	μ	σ	min	max
L	0.59	0.11	0.42	0.70
NSC	0.97	0.06	0.87	1.00
SC	0.85	0.13	0.64	0.97

are present, but do not vary substantially with the first two principal components. On the other hand, the *NSC*-model's embedding is clustered by the lighting condition (Figure 5 top center). The y-coordinate of the car in the image is not perceivable as a color-gradient like in the other setups (Figure 5 bottom center). Although the bridge of the track produces a section in which the car is occluded and its position therefore uncertain, this does not lead to gaps in the representation.
In order to quantify how well the x- and y-coordinate of the car and the lighting condition are represented in each

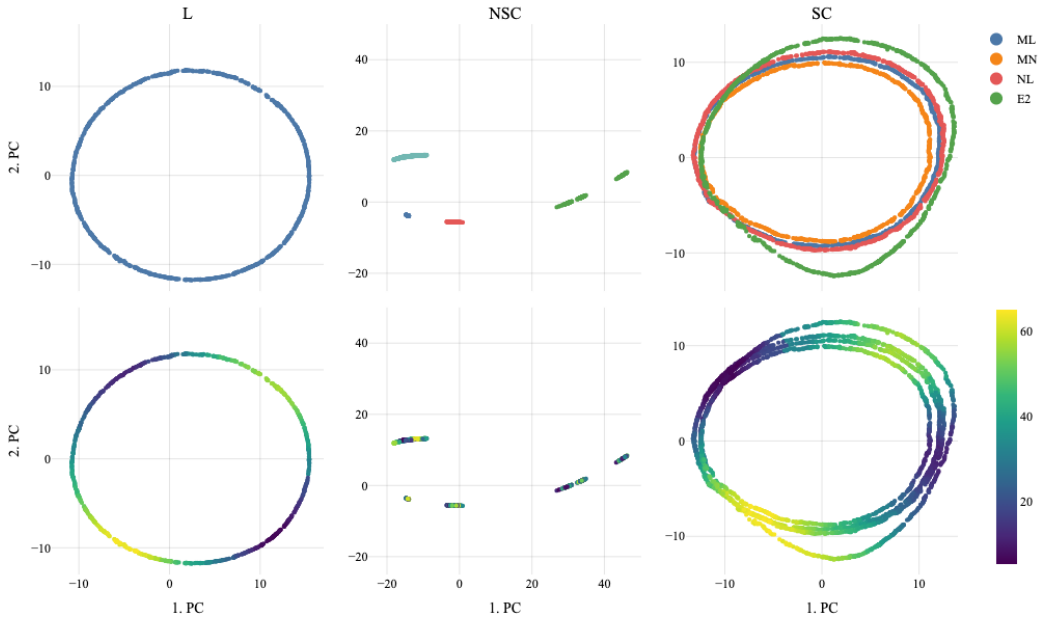


Figure 5. First two principal components (PC) for the embeddings of the models with the highest test-set accuracy for each training setup (columns). Percentage of explained variance of the two PCs is indicated above each column. Color indicates the individual lighting conditions included in the training-set (top) and the estimated y-coordinate of the car in the image (bottom). The high percentage of explained variance of the first two principal components as seen in Figure 5 is consistent over all trained models with a median of $\approx 99.9\%$ of explained variance with $max \approx 100\%$ and $min \approx 90.3\%$.

latent space, we fit support vector machines[1] mapping the encoder-representation to the respective property and assess the performance.

For both *L* and *SC*, the numbers in Table 1 show that the car's coordinates are overall predictable ($R^2 \approx 1$) from the latent vector, unlike for the *NSC*-models ($R^2 \approx 0$). For *L*, the results are however considerably less consistent than for *SC* as evident from the standard distribution. Predicting the lighting condition from the encoder-representations z is possible with higher accuracy than random for all training setups, with highest accuracy for the *NSC*-models, followed by *SC* and *L* (see Table 2).

Generalisation In order to assess how well the models generalize to previously unseen lighting conditions, we apply the presented evaluation methods to data from all lighting conditions that were not included in the training-sets of each model (referred to as the generalisation-set in the following). This again only includes the models with the highest test-set-accuracy per setup and lighting condition. For both *L* and *SC*, the car's position can be predicted from the encoder embedding fairly well with the SVM-fit for the test-set, but a discrepancy between the two approaches is apparent for the generalisation-set (Table 3). Like on the test-set, the car's position is not predictable from the embeddings of the *NSC*-models ($R^2 = 0$).

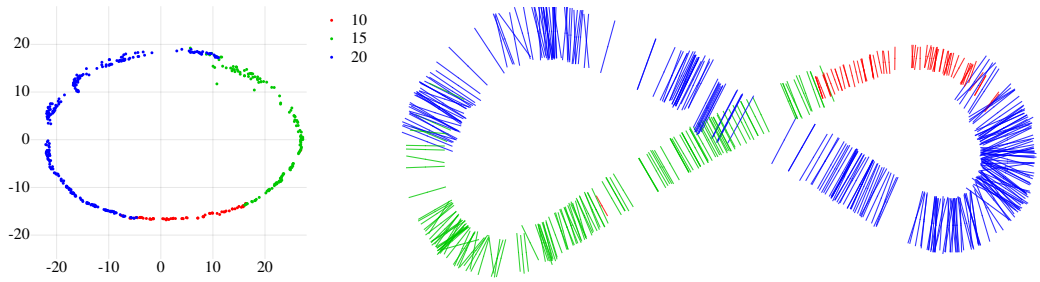
Table 3. Coefficient of determination (R^2) of regressing latent z on position of car in x- (upper) and y-position (lower) for images from the lighting conditions not included in the training-/test-sets. Summary statistics are calculated over the best models from each setup (L, *NSC*, *SC*).

		μ	σ	min	max
x-coordinate	*L*	0.84	0.21	0.10	0.98
	NSC	0.00	0.00	0.00	0.00
	SC	**0.96**	0.02	0.92	0.98
y-coordinate	*L*	0.74	0.40	-0.04	1.00
	NSC	-0.01	0.00	-0.02	-0.01
	SC	**0.99**	0.01	0.98	1.00

4.2. RL-Task

Figure 7 shows the cumulative costs divided by the number of steps for the evaluation episodes (100 steps) of one training run of 70 episodes after the initial 20 episodes of random interaction. While the agent achieves lower costs than constant driving with 15 already after two episodes of training, its performance only stabilizes after around 20 training episodes to a costs comparable to constant 15. From

[1] Support vector regressions for position, support vector classifiers for lighting condition. We used the default hyperparameters as specified in the *sklearn* machine learning library (Pedregosa et al., 2011).



(a) (b)

Figure 6. Visualisation of the fastest policy achieved by the agent. (a) The agent's policy in the encoder representation (first two principal components, 99.6% of variance explained). (b) the agent's policy for the car's position on the track. Colors indicate the chosen action.

Table 4. Costs per step and seconds per lap needed for each lighting condition compared to the baseline of constant 15.

	costs/step	seconds/lap
Baseline 15	0.0050	3.91
NL (Trained on)	0.0028	2.90
ML	0.0037	3.22
MN	0.0036	3.10
E1	0.0030	3.02
E2	0.0026	2.96

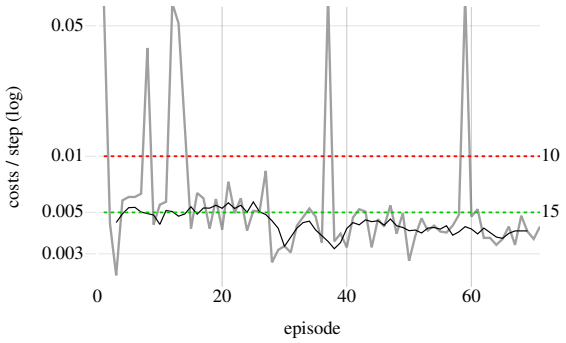


Figure 7. Costs per step over evaluation episodes for 70 episodes of training after 20 episodes of random interaction. The gray line shows actual costs per step. Costs > 0.01 are the result of reaching a terminal state. The thinner black line shows costs per step without terminal costs, smoothed by taking the mean of a rolling window over 5 episodes.

episode 25 on the agent attains on average lower costs than this baseline apart from two evaluation episodes (37 & 59), in which a terminal state was reached. Note, that because close to optimal policies are always close to terminal states for this task, occasional spikes in costs for the evaluation are expected.

As another metric of performance we report the seconds needed to complete one lap for the fastest iteration of the agent (episode 29) in table 4. With 2.9 seconds per lap under the lighting condition it was trained on, the agent is ~1 second faster than the baseline 15. Furthermore, the agent outperforms this baseline across conditions by a minimum of ~0.7 seconds, as the encoder's representation is mostly invariant to the lighting.

The policy employed by the fastest agent is visualised in Figure 6 for both the car's position on the track and the encoder's representation. For the two curves, the agent's behaviour noticeably differs: Approaching the right curve, the agent decelerates (10) in order to avoid a crash, but accelerates (20) already in the curve and all the way to the left side of the track as there is not risk of crashing there, due to the guardrail. Leaving the left curve, the agent decelerates again to 15. These three phases are also visible in the embedding, which further illustrates, how the encoder has learned an untangled representation of the track.

5. Discussion & Conclusion

Our results show, that CPC can be used to learn a sensible representation from raw image(-sequence)s for solving an RL-task of driving a slot-car around the track at high speeds without crashing. The CPC-objective could be optimized for all proposed training setups (*L*, *NSC*, *SC*) to a similar degree. However, the qualitative analysis of the learned representations reveals that the learned representations differ in which features are represented between setups.

By using selective contrast, we are able to train on data from multiple lighting conditions, while avoiding primarily learning to distinguish samples based on these conditions. In addition, the models benefit from the inclusion of multiple lighting conditions in the training-set for generalisation compared to training on only one condition. The car's position was most predictable from the embedding for the *SC*-models for both test and generalisation set, indicating that training with selective contrast produced representations which are sensible across multiple lighting conditions.

The circular structure of the learned representation shows that the CPC-model learned to represent the car's position in the context of the track, not only its coordinates in the input image.
As evident from the differences between the *NSC* and *SC* models, the expression of features in the contrasted data constitutes the structure of the learned embedding and thereby its applicability for the downstream task. Moreover, certain features can be suppressed by other, more prevalent features, if not accounted for in the pretext task. This is in line with the findings of Chen & Li (2020), who have studied contrastive learning with artificial data-sets, for which independent features can be specifically controlled, and summarize that "a few bits of easy-to-learn features could suppress, or even fully prevent, the learning of other sets of features" (Section 5, line 7-8). Although their work is based on augmentation-based contrastive learning methods, we believe that our findings are related regarding implications for contrastive learning.

Using the embedding from a CPC-encoder trained with selective contrast to build state-representations, we are able to train an agent with NFQ to drive around the track with higher speed than possible with the maximum possible constant baseline. Note, that with the applied cost-pattern this baseline is also the expected value for uniformly random actions, if leaving the track were impossible. Visualisation of the final policy reveals a generally sensible behaviour of the agent. By using CPC with selective contrast, we improve on the previous system of Lange et al. (2012) by achieving high performance independent of lighting conditions and integrating the environment's dynamics into the representation learning process.

Acknowledgements

This study was partly funded by the German Federal Ministry of Education and Research (BMBF) via the DeToL project (grant number 01IS18046F) and by the European Research Council (ERC-StG-2015—BRISC: 678082).

References

Chen, T. and Li, L. Intriguing properties of contrastive losses, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020a.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. Big self-supervised models are strong semi-supervised learners, 2020b.

Cho, K., van Merrienboer, B., Gülçehre, Ç., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

Doersch, C. and Zisserman, A. Multi-task self-supervised visual learning. *CoRR*, abs/1708.07860, 2017. URL http://arxiv.org/abs/1708.07860.

Dulac-Arnold, G., Mankowitz, D., and Hester, T. Challenges of real-world reinforcement learning, 2019.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Henaff, O. Data-efficient image recognition with contrastive predictive coding. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4182–4192. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/henaff20a.html.

Kietzmann, T. C. and Riedmiller, M. The neuro slot car racer: Reinforcement learning in a real world setting. In *2009 International Conference on Machine Learning and Applications*, pp. 311–316. IEEE, 2009.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. *Advances in neural information processing systems*, 30:971–980, 2017.

Lange, S. *Tiefes Reinforcement-Lernen auf Basis visueller Wahrnehmungen.* PhD thesis, University of Osnabrück, 2010.

Lange, S., Riedmiller, M., and Voigtländer, A. Autonomous reinforcement learning on raw visual input data in a real world application. In *The 2012 international joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2012.

Le-Khac, P. H., Healy, G., and Smeaton, A. F. Contrastive representation learning: A framework and review. *IEEE Access*, 2020.

Mazoure, B., des Combes, R. T., Doan, T., Bachman, P., and Hjelm, R. D. Deep reinforcement and infomax learning, 2020.

Misra, I., Zitnick, C. L., and Hebert, M. Unsupervised learning using sequential verification for action recognition. *CoRR*, abs/1603.08561, 2016. URL http://arxiv.org/abs/1603.08561.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Qian, R., Meng, T., Gong, B., Yang, M.-H., Wang, H., Belongie, S., and Cui, Y. Spatiotemporal contrastive video representation learning, 2020.

Riedmiller, M. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

Srinivas, A., Laskin, M., and Abbeel, P. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

Vondrick, C., Shrivastava, A., Fathi, A., Guadarrama, S., and Murphy, K. Tracking emerges by colorizing videos. *CoRR*, abs/1806.09594, 2018. URL http://arxiv.org/abs/1806.09594.

Wang, L., Kawakami, K., and van den Oord, A. Contrastive predictive coding of audio with an adversary. *Proc. Interspeech 2020*, pp. 826–830, 2020.

Xiao, T., Wang, X., Efros, A. A., and Darrell, T. What should not be contrastive in contrastive learning, 2020.

Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. *CoRR*, abs/1603.08511, 2016. URL http://arxiv.org/abs/1603.08511.