# Boosting Reinforcement Learning with Unsupervised Feature Extraction

Simon Hakenes[0000-0002-4623-6364]* and Tobias Glasmachers[0000-0003-1886-1696]

Institute for Neural Computation, Ruhr University Bochum, Germany
{simon.hakenes,tobias.glasmachers}@ini.rub.de

**Abstract.** Learning to process visual input for Deep Reinforcement Learning is challenging and training a neural network with nothing else but a sparse and delayed reward signal seems rather inappropriate. In this work, Deep Q-Networks are leveraged by several unsupervised machine learning methods that provide additional information for the training of the feature extraction stage to find a well suited representation of the input data. The influence of convolutional filters that were pretrained on a supervised classification task, a Convolutional Autoencoder and Slow Feature Analysis are investigated in an end-to-end architecture. Experiments are performed on five ViZDoom environments. We found that the unsupervised methods boost Deep Q-Networks significantly depending on the underlying task the agent has to fulfill. While pretrained filters improve object detection tasks, we find that Convolutional Autoencoders leverage navigation and orientation tasks. Combining these two approaches leads to an agent that performs well on all tested environments.

**Keywords:** Deep Reinforcement Learning · Unsupervised Learning

## 1 Introduction

When thinking about an autonomous agent acting in the real world, deep Reinforcement Learning (RL) algorithms, that are able to efficiently process visual input, are of central importance. In order to learn an intelligent behaviour, high level information about the environment has to be extracted from the input data. Learning both, feature extraction and policy, from nothing else than a scalar, sparse and delayed reward signal seems to be inefficient. Especially in three dimensional environments with an agent that is able to move around, the state space is huge and impossible to discover completely. Similar to humans and animals, which learn mostly unsupervised [11], combining unsupervised machine learning methods with reinforcement learning algorithms into an end-to-end solution is promising in terms of accelerated training while keeping the training procedure simple.

We therefore apply several unsupervised machine learning techniques with the goal to learn meaningful visual filters quickly, so that reinforcement learning can focus on learning a policy expressed by the higher (dense) layers of the network.

We compare several unsupervised machine learning techniques in combination with reinforcement learning in an end-to-end framework for their suitability towards this goal. As a baseline, a Deep Q-Network (DQN) is trained from scratch, i.e., with randomly initialized weights and without any auxiliary method, as in the work of Mnih et al. [19]. As a first step, the opportunities of transfer learning are explored by pretraining the convolutional layers of the DQN with real world images in a supervised classification task. Furthermore, we add two different unsupervised auxiliary objective functions to the DQN: a Convolutional Autoencoder (CAE) [18] and a gradient-based variant of Slow Feature Analysis (SFA) [24]. CAEs operate on single images and ignore the rich temporal structure of visual input. SFA is based on the idea that the most informative signals are slowly changing over time compared to the raw sensor signals, and hence takes the temporal structure into account. All three methods affect only the convolution filters. After initial experimentation, the pretrained filters and the CAE were found to be most promising, however, for different tasks. Hence, we also explored the combination of both with a joint feature set.

For reproducibility, the code is publicly available at https://github.com/shakenes/unsupervised-drl.

The remainder of this work is organized as follows. After discussing related work we introduce the background and the environments in Sections 3 and 4. We describe our experiments, present and discuss the results in sections 5, 6, and 7, and finally draw conclusions.

## 2 Related Work

The combination of unsupervised learning with reinforcement learning was explored by several authors.

Unsupervised auxiliary tasks for RL that each yield a reward by themselves are proposed in [12,21]. However, the visual input is only exploited indirectly with an additional reward function, in contrast to using established unsupervised learning algorithms.

The authors of [3] decoupled feature extraction from policy learning. They proposed two new methods based on vector quantization and sparse coding to learn the features separately but simultaneously to the policy. Some interesting results on Atari games were shown as a neural network consisting of only 6-18 neurons was necessary to learn the policy. One may conclude that the actual RL part is nearly trivial when fed with high level and low dimensional features. However, the vector quantization methods are unsuitable for quickly changing scenes of 3D environments like ViZDoom.

In [1,16] an autoencoder is used for dimensionality reduction of visual input data and learning a policy based on the encoded data. However, the study aimed for a bottleneck as small as possible, while we only aim for efficient training of the convolutional layers. Similarly, in [15] the CAE features are used to predict the immediate rewards and to compute the Successor Representation for each possible action.

## 3 Theoretical Background

### 3.1 Reinforcement Learning

Many sequential decision making processes can be formulated as a Markov Decision Process (MDP) defined by the state space $\mathcal{S}$, the set of possible actions $\mathcal{A}$, the state transition probability $\mathcal{P}(s_{t+1}|s_t, a_t)$, the reward function $\mathcal{R}(s, a)$ and the discount factor $\gamma \in (0, 1]$, which represents the preference of immediate rewards over future rewards. An agent receiving ego-perspective visual input in a three dimensional environment can only observe the state partially. The future discounted reward at time $t$ is $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t$, where $T$ is the time step at which the MDP terminates and $r_t$ is the reward received in time step $t$.

The most prominent class of deep RL algorithms are temporal difference methods, in particular Q-Learning [25]. Key to Q-Learning is approximating the optimal action-value function, that returns the expected $R_t$ after seeing some state $s$, taking an action $a$ and following a policy $\pi$ thereafter:

$$Q^*(s, a) = \max_{\pi} \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right] \tag{1}$$

For high-dimensional visual inputs, this function can be approximated with a CNN, which makes it dependent on the network weights $\theta$ [19]: $Q^*(s, a; \theta)$. The Q-Learning loss is then:

$$L_i(\theta_i) = \mathbb{E}\left[(y_i - Q(s, a; \theta_i))^2\right] \tag{2}$$

with the target

$$y_i = \mathbb{E}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1} | s, a)\right] \tag{3}$$

In this work, double Q-Learning [9] is applied to avoid overestimation of the Q-values.

### 3.2 Transfer Learning

The general idea of transfer learning is to gain knowledge about one task and exploit it in a different but related task. In the case of a CNN, instead of training it from scratch with randomly initialized weights, it can be pretrained on an arbitrary visual task. The weights, especially the last layers, are then fine-tuned with respect to the new task. This is more data efficient and often leads to better performance and faster convergence than training from scratch [8,20,29,28]. This fact suggests that the features extracted by a CNN are generic and not specific for any task.

### 3.3 Convolutional Autoencoders

An autoencoder is a neural network with a small central layer, which is used to learn a low-dimensional representation of high-dimensional input data. To achieve this, the autoencoder is trained to approximate the identity function in a least squares sense. It can hence be thought of as a non-linear extension of principal component analysis for dimensionality reduction. Since the central layer is smaller than the input and output layers, it acts as an information bottleneck, which forces the encoder to transform the input into a low dimensional representation. The decoder needs to back-transform the data such that the output data is as similar as possible to the input data [10]. For visual data, it makes sense to exploit the local correlation between the pixels by using (stacked) convolutional layers [18].

### 3.4 Slow Feature Analysis

Slow Feature Analysis follows the intuition that the most informative features are slowly changing over time compared to raw input/sensor signals [26,27]. It was shown that SFA is able to encode and disentangle object identity, rotation and position in visual tasks [5], as well as position and orientation from visual first person recordings [6].

SFA can be formally described as a sequence of optimization problems. Considering a time series $\{x_t\} \in \mathbb{R}^d$ with $t = 0, \ldots, \tau$, SFA aims at finding input-output functions $g_i : \mathbb{R}^d \to \mathbb{R}$ that minimize $\left\langle (g_i(x_{t+1}) - g_i(x_t))^2 \right\rangle_t$ where $\langle \cdot \rangle_t$ being the average over time. To avoid the trivial constant solution, to ensure normalization and to avoid redundant features, zero mean $\langle g_i(x_t) \rangle_t = 0$, unit variance $\left\langle g_i(x_t)^2 \right\rangle_t = 1$ and decorrelation, $\langle g_i(x_t) g_j(x_t) \rangle_t = 0$ are enforced.

Schüler et al. [24] proposed a method for gradient-based end-to-end training of SFA. In order to do that, a differentiable loss as well as a differentiable whitening procedure enforcing the constraints (zero mean, unit variance and decorrelation) are designed.

## 4 Environments

Our experiments were carried out on five ViZDoom [13] environments which are described in this section. Screenshots are given in Fig. 1. The simplest environment is *Basic* where the agent is supposed to hit an immobile enemy with a gun while moving only left and right. If hit, the episode terminates and a reward is given. Missing the enemy, timeout and time passing are punished. To solve the task, the agent only needs to detect the target object's location on a static background, which is quite easy.

In *Defend the Center*, the agent is spawned in the center of a circular room together with several enemies. The agent cannot move around, but only turn left and right. The goal is to kill the monsters before they reach the agent. Each kill is rewarded while dying is punished. Here, the agent has to detect the locations of multiple objects at once on a static background. Additionally, since enemies might be behind the agent, it has to keep track of its complete surrounding area.

*Health Gathering* consists of a rectangular room with an acid floor that damages the agent over time. To survive, the agent has to gather health packs that are randomly distributed on the floor. Increasing the health by collecting such a health pack is rewarded. In contrast to the preceding environments, the agent has three degrees of freedom instead of one. Therefore, the agent needs to actually navigate in the room and detect objects on a dynamically changing background.

*Health Gathering Supreme* is similar to *Health Gathering* except for maze-like walls in the room. That means, simple (input ignoring) policies like running in a circle become useless. Additionally, there are poisonous potions that lower the agent's health when collected. This constellation makes navigation much more challenging and demands the ability to distinguish between different kinds of objects.

The environment *My Way Home* places the agent in a maze with several connected rooms. A green vest is placed as a target in one of the rooms. The agent is spawned in a random room facing in a random direction. When the agent finds the vest, it gets a reward. Navigation and orientation by recognizing the rooms are the key challenges in this environment.

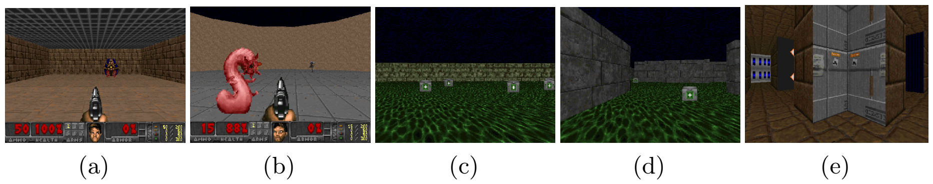


**Fig. 1.** Screenshots of the environments: (a) Basic, (b) Defend the Center, (c) Health Gathering, (d) Health Gathering Supreme, (e) My Way Home.

## 5 Experiments

We carried out experiments aiming to answer the following research question: can unsupervised methods exploit the rich visual data to quickly evolve useful visual filters and speed up reinforcement learning? And if so, which method works best?

To this end, five experiments were carried out in each environment. As a baseline, the DQN was trained from scratch, i.e. with randomly initialized weights without any auxiliary method. The feed-forward CNN contains two convolutional layers with 32 $7 \times 7$ filters and $4 \times 4$ strides in the first, and 32 $5 \times 5$ filters with $2 \times 2$ strides in the second layer. They are followed by a dense layer with 1024 units. Each layer has a ReLU activation, $f(x) = \max(0, x)$. The last layer is dense (fully connected) with as many units as there are possible actions in the environment with linear activations representing the Q-values. The DQN is fed with $128 \times 128$ px grayscale images.

In order to exploit the capabilities of transfer learning, the convolutional layers of the DQN are pretrained on a subset of the ImageNet data set [22] in a supervised classification task. The data set contains real world images of animals and objects. The evolved convolutional filters form the initialization of the DQN. The fully connected layers are initialized at random.

To combine a CAE with the DQN, the decoder branch is placed on top of the last convolutional layer (Fig. 2a). Two dense layers with 1024 and $128^2$ units, respectively, reconstruct the input image and propagate their gradients into the encoding convolutional layers. The DQN branch stays unaltered. Both branches are trained alternately. The training of the CAE is stopped after some time to prevent overfitting.

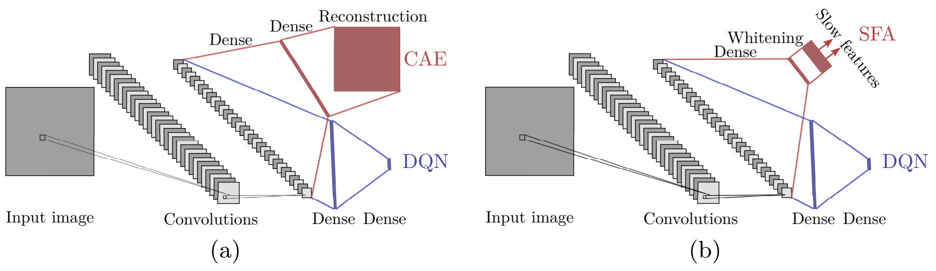


**Fig. 2.** Visualization of the CAE (a) and SFA (b) architecture.

Similar to the CAE architecture, a dense layer with 16 units and the power whitening layer [24] are placed on top of the convolutional layers (Fig. 2b). In contrast to previous work [17] the slow features are not fed into the value function. Their slowness just serves as an objective function for training more powerful filters. Note that the batches for DQN and SFA differ, because the SFA needs data from consecutive time steps.

Initial experiments showed that the pretrained filters and the CAE show the most promising results, however, on different environments. Therefore, we implemented a combination of both methods. Our architecture contains two parallel filter banks, one with pretrained filters and one trained by the CAE (which does not impact the pretrained filters). The dense layers connect to both filter banks, allowing the DQN to decide which set of features to use for which task – or to use both simultaneously. This architecture is referred to as the *Combination* in the following.

## 6 Results

### 6.1 Reward

All proposed methods improve the performance of the agent in terms of reward in the *Basic* environment as shown in Fig. 3a. They all manage to learn a sensible behaviour after less episodes than the baseline method which stagnates in the beginning. This scenario is mostly an object detection task as the agent can only move left and right and needs to align its gun with the still standing enemy to shoot it.

In *Defend the Center*, the Combination and the pretrained filters alone improve the performance greatly (Fig. 3b). The CAE and the SFA are not able to bring any benefit. Here, the agent solves an object detection task with multiple enemies that need to be shot in order to survive. However, the agent stands in a circular room and can only turn itself left and right, so there is no navigation involved.

In contrast to *Basic* and *Defend the Center*, the agent faces the challenges of an ego perspective 3D environment in *Health Gathering*, where it can move freely through the room. Here, training from scratch, the CAE, and the Combination work well. Surprisingly, SFA and the pretrained filters harm the performance significantly. The DQN apparently learns to choose the features from the CAE branch to determine the policy, as the pretrained filters have a far worse performance (Fig. 3c).

In *Health Gathering Supreme* the same task has to be solved in a maze like environment. The agent also has to distinguish between health kits and poisonous potions. The CAE improves the agent's performance, while the Combination shows the same performance as the baseline. Again, the pretrained filters and the SFA lead to a worse result (Fig. 3d).

In *My Way Home*, the agent needs to navigate through a labyrinth. The Combination of CAE and pretrained filters is the only method that really improves the agent's performance compared to the baseline (Fig. 3e).

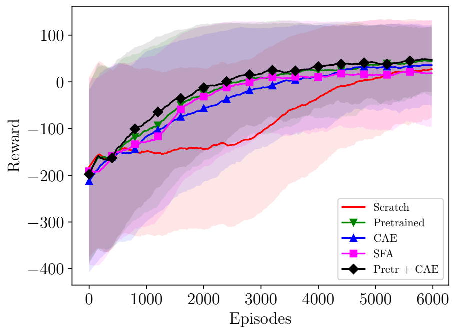
## 6.2 Convolutional Filters

Since our central aim was to evolve filters quickly so a policy can be learned on top, we investigate the filters of the convolution layers in more details. Big differences between methods and environments can be observed. When trained from scratch without any auxiliary method, the DQN did not evolve a meaningful structure in the convolutional layers, see Fig. 4a. A more detailed investigation showed that they only changed negligibly compared to their initialization.

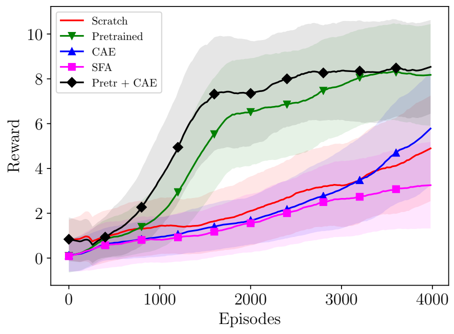
The pretrained filters behave similarly. Although minor changes are visible, their overall structure remains the same during the training procedure (Fig. 4b).

In the challenging environments *Health Gathering* and *Health Gathering Supreme*, the CAE encodes the features using well structured filters (Fig. 4c). One can observe edge detectors, centre-surround filters and cross-shaped structures, suitable for the detection of health packs. In contrast, the filters after training on *Basic* and *Defend the Center* are quite unstructured. The second convolutional layer still lacks meaningful structures.

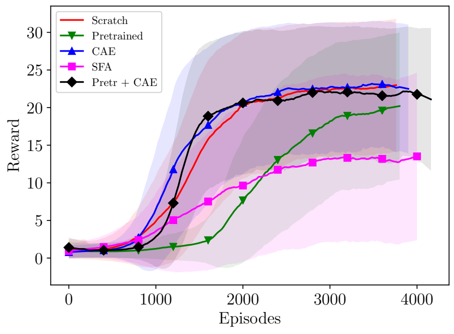
As the only method, the SFA develops filters in both convolutional layers (Fig. 4d). However, their functional "meaning" is not obvious from visual inspection. The slowness loss drops significantly throughout the training, showing that the CNN successfully evolves slow features.



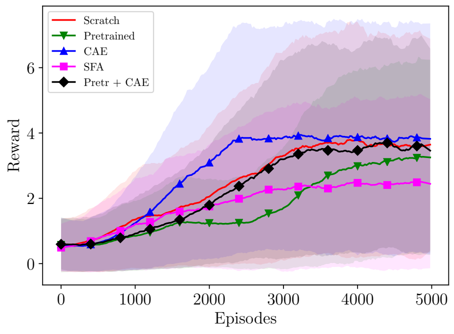
(a) *Basic*



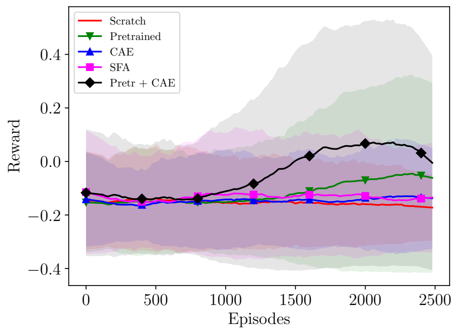
(b) *Defend the Center*



(c) *Health Gathering*



(d) *Health Gathering Supreme*



(e) *My Way Home*

**Fig. 3.** The mean reward per episode with standard deviation over 15 runs.

As expected, the combination of pretrained filters and CAE develop similar filters compared to the individual approaches, because they cannot interfere with each other.

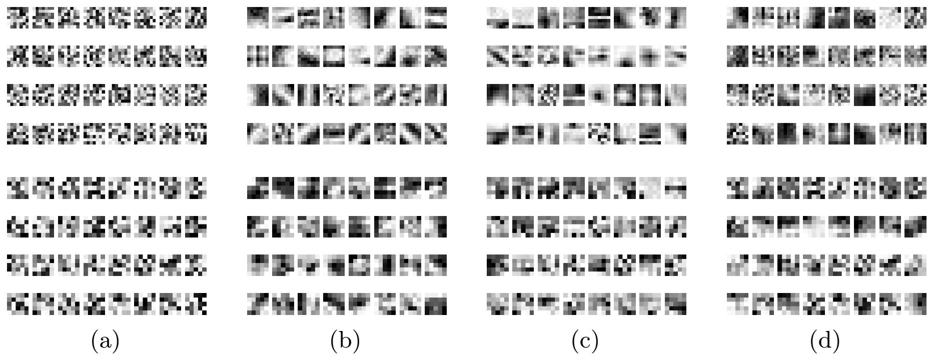


**Fig. 4.** Examples of evolved filters after training. The upper mosaics show the filters of the first convolutional layer while the lower mosaics belong to the second layer.

## 7 Discussion

The results show that unsupervised auxiliary methods can accelerate the training of a DQN significantly. The impact highly depends on the method and the environment. None of the methods works well in isolation across all environments. However, the proposed Combination of pretrained convolutional filters and a CAE seems to be a good and universally applicable choice. This can be explained by the joint set of features that specialize in different tasks: the pretrained filters perform better in object detection tasks, which is unsurprising because that is the task they were initially trained on, while the CAE focuses on global properties of the scene, which helps the agent to navigate in three dimensional environments.

The fact that the filters did not change during training and stayed as initialized implies that the DQN on its own is not able to generate useful gradients to develop actually meaningful filters. This suggests that it is important to somehow solve the problem of feature extraction when attacking complex reinforcement learning tasks. This is underlined by the findings of [3]. Their work implies that the reinforcement learning part itself is rather easy when there is a well suited representation of the input data.

The SFA shows disappointing results. Although it seems beneficial to exploit the temporal characteristics of the input signals, the used method was not able to develop a well suited representation for reinforcement learning. It is worth noting that the authors of [24] use far deeper architectures than we did.

Taking a closer look at the evolved filter kernels in the convolutional layers, one can conclude that unsupervised methods do help to develop more powerful filters, with the CAE showing the most impressive results. At this point two questions have to be answered: First, how is the quality of a filter kernel determined? And second, how should one interpret the still random looking filters in the second layer, and in the first layer when training from scratch? To answer the first question, filters from the literature are investigated [30,14]. Well evolved filters in CNNs often show some kind of structure with the weights being locally correlated. They usually can easily be distinguished from the randomly initialized ones. Similar to filters used in computer vision, it is often possible to assign functions like edge detection or templates to good filters. However, the results of this work show that even random looking filters can lead to a good performance in visual RL. As [23] stated, CNNs can perform surprisingly well with random filters, nevertheless outperformed by a pretrained and fine-tuned one. They showed that random filters are frequency selective, which might already be a decent enough feature in our case. Another explanation can be the findings of [2]. They investigated Random Projections of high dimensional data onto a lower-dimensional subspace using a random matrix. This does usually not distort the data in a significant way. Possibly, the filters stayed random looking due to a too high number of parameters of the model, which would suggest to lower the number of hidden units in the network, for example by decreasing the number of filters in the second layer. A short, qualitative study on this hypothesis shows that scaling the number of filters in the second convolutional layer in our DQN down by a factor of two leads to results comparable to the full network. Training time and maximum performance only suffer slightly, but the method based on SFA becomes quite unstable. However, it is worth mentioning here that [19] used even three convolutional layers with 32, 64 and 64 filters in their DQN, while testing their algorithm on Atari 2600 games, which are less challenging in terms of exploiting visual input than the ones explored in this work. We therefore hypothesize that their results could possibly also be achieved with a DQN with fewer parameters.

## 8 Conclusion

Returning to our initial question, it is now possible to state that unsupervised auxiliary methods can actually improve deep RL from visual input. While pre-trained convolutional filters accelerate the training in environments that mostly require object detection and localization, CAEs greatly improve the performance in three dimensional worlds where it is crucial for the agent to fulfill navigation tasks. The Combination of both shows good results on all tested environments. Although these methods are completely unsupervised and can therefore not know which features are useful for the task, they still help developing meaningful filters to extract features that are well suited to learn sensible behavior.

An explanation for the poor performance of the DQN is given by pointing out that it was not able to develop meaningful filters on its own. Therefore, the importance of a powerful feature extraction stage for RL on complex problems is emphasized.

In future work we will develop our approach further by considering additional feature sets that are generated by objective functions aiming for monocular depth or optical flow estimation, since these take the spatial properties of the three dimensional environments into account [7,4].

## References

1. Alvernaz, S., Togelius, J.: Autoencoder-augmented neuroevolution for visual Doom playing. CoRR **abs/1707.03902** (2017), http://arxiv.org/abs/1707.03902
2. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: Applications to image and text data. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 245–250. ACM, New York, NY, USA (2001). https://doi.org/10.1145/502512.502546
3. Cuccu, G., Togelius, J., Cudré-Mauroux, P.: Playing atari with six neurons. In: International Conference on Autonomous Agents and MultiAgent Systems. pp. 998–1006. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2019)
4. Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazirbas, C., Golkov, V., v. d. Smagt, P., Cremers, D., Brox, T.: FlowNet: Learning optical flow with convolutional networks. In: IEEE International Conference on Computer Vision (ICCV). pp. 2758–2766 (Dec 2015). https://doi.org/10.1109/ICCV.2015.316
5. Franzius, M., Sprekeler, H., Wiskott, L.: Slowness and sparseness lead to place, head-direction, and spatial-view cells. PLOS Computational Biology **3**(8), 1–18 (08 2007)
6. Franzius, M., Wilbert, N., Wiskott, L.: Invariant object recognition and pose estimation with slow feature analysis. Neural computation **23**, 2289–323 (06 2011)
7. Godard, C., Aodha, O.M., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6602–6611 (July 2017). https://doi.org/10.1109/CVPR.2017.699
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
9. Hasselt, H.v., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: AAAI Conference on Artificial Intelligence. pp. 2094–2100. AAAI Press (2016)
10. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006). https://doi.org/10.1126/science.1127647
11. Hinton, G., Sejnowski, T., Poggio, T.: Unsupervised Learning: Foundations of Neural Computation. A Bradford Book, MCGRAW HILL BOOK Company (1999)
12. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. CoRR **abs/1611.05397** (2016)
13. Kempka, M., Wydmuch, M., Runc, G., Toczek, J., Jaśkowski, W.: ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In: IEEE Conference on Computational Intelligence and Games. pp. 341–348. IEEE, Santorini, Greece (Sep 2016)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)

15. Kulkarni, T.D., Saeedi, A., Gautam, S., Gershman, S.J.: Deep successor reinforcement learning. CoRR **abs/1606.02396** (2016), http://arxiv.org/abs/1606.02396
16. Lange, S., Riedmiller, M.: Deep auto-encoder neural networks in reinforcement learning. In: International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (July 2010). https://doi.org/10.1109/IJCNN.2010.5596468
17. Legenstein, R., Wilbert, N., Wiskott, L.: Reinforcement learning on slow features of high-dimensional input streams. PLoS Computational Biology **6**(8), e1000894 (2010)
18. Masci, J., Meier, U., Cireşan, D., Schmidhuber, J.: Stacked convolutional autoencoders for hierarchical feature extraction. In: Honkela, T., Duch, W., Girolami, M., Kaski, S. (eds.) Artificial Neural Networks and Machine Learning – ICANN 2011. pp. 52–59. Springer Berlin Heidelberg (2011)
19. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (02 2015)
20. Oquab, M., Bottou, L., Laptev, I., Sivic, J.: Learning and transferring mid-level image representations using convolutional neural networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1717–1724 (June 2014)
21. Papoudakis, G., Chatzidimitriou, K.C., Mitkas, P.A.: Deep reinforcement learning for Doom using unsupervised auxiliary tasks. CoRR **abs/1807.01960** (2018), http://arxiv.org/abs/1807.01960
22. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet large scale visual recognition challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015)
23. Saxe, A.M., Koh, P.W., Chen, Z., Bhand, M., Suresh, B., Ng, A.Y.: On random weights and unsupervised feature learning. In: International Conference on Machine Learning. pp. 1089–1096. Omnipress, USA (2011)
24. Schüler, M., Hlynsson, H.D., Wiskott, L.: Gradient-based training of slow feature analysis by differentiable approximate whitening. CoRR **abs/1808.08833** (2018), http://arxiv.org/abs/1808.08833
25. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. thesis, King's College, Cambridge (1989)
26. Wiskott, L.: Learning invariance manifolds. In: International Conference on Artificial Neural Networks (ICANN). pp. 555–560. Springer (1998)
27. Wiskott, L., Sejnowski, T.: Slow feature analysis: unsupervised learning of invariances. Neural Computation **14**(4), 715–770 (2002)
28. Wohlfarth, K., Schröer, C., Klaß, M., Hakenes, S., Venhaus, M., Kauffmann, S., Wilhelm, T., Wöhler, C.: Dense cloud classification on multispectral satellite imagery. In: IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS) (Aug 2018). https://doi.org/10.1109/PRRS.2018.8486379
29. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, pp. 3320–3328. Curran Associates, Inc. (2014)
30. Zeiler, M., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision (ECCV). Lecture Notes in Computer Science, vol. 8689, pp. 818–833. Springer Verlag (2014). https://doi.org/10.1007/978-3-319-10590-1_53