Reinforcement learning vs supervised learning

A comparison on DonkeyCar autonomous driving

Master's Thesis submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Master of Science in Informatics
Dependable Artificial intelligence

presented by
Giorgio Macauda

under the supervision of
Prof. Paolo Tonella
co-supervised by
PhD Matteo Biagiola

September 2022

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Giorgio Macauda
Lugano, 12 September 2022

To my beloved

Someone said . . .

Someone

Abstract

This is a very abstract abstract.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et

vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Acknowledgements

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras

ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Contents

Contents **xi**

List of Figures **xiii**

List of Tables **xv**

1 Background **1**
1.1 Reinforcement Learning 1
1.2 OpenAI Gym interface 3
1.3 Soft Actor Critic - SAC 5
1.4 Generative Adversarial Networks - GAN 6
1.5 CycleGAN 7
1.6 AutoEncoder and Variational AutoEncoder 7
1.6.1 AutoEncoder 7
1.6.2 Variational AutoEncoder 9
1.7 DonkeyCar 10

2 Related works **13**
2.1 Describe here the related works 13

A Some retarded material **15**
A.1 It's over... 15

Glossary **17**

Bibliography **19**

Figures

1.1 Basic reinforcement learning 1
1.2 Atari games in Gym 4
1.3 Soft actor critic 5
1.4 GAN diagram 6
1.5 Image-to-image translation example 7
1.6 AE diagram 8
1.7 Example of an AE latent space Z on MNIST dataset 8
1.8 VAE diagram 9
1.9 Example of a VAE latent space Z on MNIST dataset 10
1.10 Assembled donkeycar 10

Tables

Chapter 1

Background

1.1 Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning, alongside supervised learning and unsupervised learning, that defines a set of algorithms meant to learn how to act in a specific environment without the need of labelled data to learn from.

The algorithm defines the agent that learns a given task, for example, walking, driving and playing a game, by trial and error, while interacting with an environment which can be real or simulated. Whenever the agent makes a set of good actions it receives a positive reward, which makes such actions more likely in the future. State, action and reward are the most important concepts in RL. The state represents the current situation of the environment. If the agent is a humanoid robot and the task is walking, one possible state representation is the positions of its actuated joints. The action set or space in case of continouos domain, describes what the agent can do in a particular state. In the humanoid robot example above, the action space is a n-dimensional vector where each dimension represents the torque command to each of the n joint motors. Finally the reward is a measure of good are the actions carried out by the agent.

The reward function, usually human-designed, assigns a score to the action taken by the agent. Every action that leads to a *good* state increases the score and viceversa every *bad* action decreases it. As described in Figure 1.1 the agent interacts with the environment in discrete time steps. At time t it gets the current state s_t and the associated reward r_t then the action a_t is chosen from the set of available actions. After receiving the chosen action, the environment moves to a new state s_{t+1} and the reward r_{t+1} is given back to the agent. The total discounted reward to be maximized is:

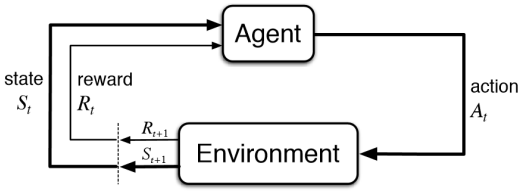

Figure 1.1. Basic reinforcement learning

$$R = \sum_{t=0}^{T} \gamma^t r_t \tag{1.1}$$

where T is the time horizon (enventually ∞), $\gamma \in [0,1)$ is the discount factor which allows R to be a finite value in case of $T = \infty$ and makes future rewards worth less than immediate reward. The total discounted reward function is fundamental to the agent in order to learn and optimize a policy function π:

$$\pi : A \times S \rightarrow [0,1] \qquad \pi(a,s) = \Pr(a_t = a \mid s_t = s) \tag{1.2}$$

The policy is a mapping that gives the probability of taking action a in state s. By following the policy the agent takes the action that maximizes the reward. However, the policy, especially during training, is not deterministic. This is due to one of the fundamental challenges in RL, i.e. the exploration-exploitation dilemma [Sutton and Barto, 2018]. Indeed, the agent needs to repeat the actions it already know to be rewarding but, at the same time, it needs to explore the environment to discover actions that can lead to an even higher reward. The final goal of the algorithm is to learn a policy that maximizes the expected cumulative reward.

$$J(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)] \tag{1.3}$$

There are multiple ways to learn the optimal policy $\pi^*(s)$. The first one is called *Value iteration*, which exploits the state value function $V^{\pi}(s)$ and the action value function $Q^{\pi}(s,a)$. The state value function V is the expected return starting from the state s and following the policy π:

$$V_{\pi}(s) = E_{\pi}[\sum_{t=0}^{T-1} \gamma^t r_t \mid s_t = s] \tag{1.4}$$

while the action value function Q is the expected return starting from the state s, following the policy π, taking action a:

$$Q_{\pi}(s,a) = E_{\pi}[\sum_{t=0}^{T-1} \gamma^t r_t \mid s_t = s, a_t = a] \tag{1.5}$$

There is an important reletionship between function 1.4 and 1.5, in fact they can be written in terms of each other:

$$V_{\pi}(s) = \sum_{a \in A} \pi(a \mid s) * Q^{\pi}(s,a) \tag{1.6}$$

$$Q_{\pi}(s,a) = \sum_{s' \in S} P(s' \mid s,a)[r(s,a,s') + \gamma V^{\pi}(s')]. \tag{1.7}$$

where P is the state transition matrix that gives the probability of reaching the next state s' from state and R is the immediate reward.

In *Value Iteration*, we start from a random intialized V and the algorithm repeatedly updates Q and V values until they converges, with the guarantee that they will converges to the optimal values.

```
1 Initialize V(s) to arbitrary values
2 Repeat
3     for all s in S
4         for all a in A
5             Q(s,a) = E[r|s,a] + γ ∑_{s'∈S} P(s'|s,a)V(s')
6         V(s) = max_a Q(s,a)
7 until V(s) converges
```

Listing 1.1. Value iteration pseudo code from Alpaydin [2014]

Finally the policy π can be inferred from the Q function with:

$$\pi(s) = argmax_a Q(s,a) \tag{1.8}$$

Since the agent only cares about the finding the optimal policy, it could happen that policy converge before the value function. Therefore, the so-called *Policy Iteration* algorithm seeks to learn the policy directly by updating it at each step as shown in pseudo code 1.2

```
1 Initializea policy π' arbitrarily
2 Repeat
3     π = π'
4     Compute the values using π
5         V_π = E[r|s,π(s)] + γ ∑_{s'∈S} P(s'|s,π(s))V_π(s'))
6     Improve the policy at each state
7         π'(s) = argmax_a(E[r|s,a] + γ ∑_{s'∈S} P(s'|s,a)V_π(s')))
8 until π = π'
```

Listing 1.2. Policy iteration pseudo code from Alpaydin [2014]

Policy iteration is also guaranteed to converge to the optimal policy and it often takes less iterations to converge than the value iteration algorithm.

A major problem arises when the environment is not entirely known to the agent or is too big that is unfeasible to store all the Q and V values in a table. As well as, in the case of a continuous action space. Deep RL algorithms introduces Deep Neural Networks in order to approximate Q and V instead of storing them in huge tables. Function approximation allows also a better generalization of states never seen before, or with partial information, by exploiting values of similar states.

Another important difference between RL algorithm that is worth a brief mention, is in the way the algorithm updates the policy. On-Policy methods evaluates and improve the same policy which is being used to select actions. Off-Policy methods evaluates and improve a policy that is different from policy that is used for action selection bringing many advantages. Firstly, it allows a better exploration of new trajectories. Secondly, the agent can learn from demonstrations and finally it allows parallel learning speeding up the convergence.

1.2 OpenAI Gym interface

Gym is an open source library that defines a standard API to handle training and testing of RL agents, while providing a diverse collection of simulated environments.

The environment is of primary importance to a RL algorithm since it defines the world of the agent in which the agent lives and operates. The standard interface designed by Gym, makes it easier to interact with environments, both made available by Gym and externally developed.

The Gym interface is simple, pythonic, and capable of representing general RL problems. The documentation provides a reference template 1.3 that describe what are the fundamental methods a gym environment should implement to work properly.

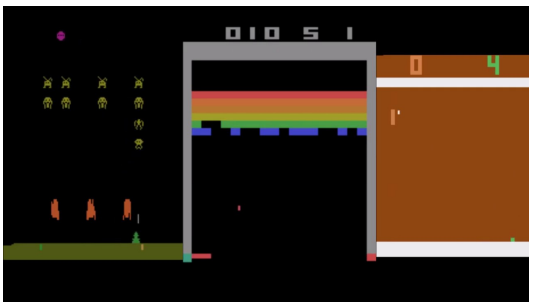

Figure 1.2. Atari games in Gym

Any exisisting environment built with Gym implements the following few methods which are enough to run any basic RL algorithm or eventually override them in case of custom environments:

```
1  class GymTemplate(gym.Env):
2    def __init__(self):
3      pass
4    def step(action):
5      pass
6    def reset():
7      pass
8    def render(mode='human'):
9      pass
10   def close(self):
11     pass
```

Listing 1.3. "Gym template"

- **init:** every environment should extend gym.Env and contain the variables *observation_space* and *action_space* specifying the type of possible observations and actions using spaces.Box or spaces.Discrete.
- **step:** this method is the primary interface between environment and agent, it takes as input the action and return informations (observation, reward, done) about the current state.
- **reset:** this method resets the environment to its initial values returning the initial state of the environment.
- **render:** this method pops up a window rendering the environment when a parameter *mode='human'* is passed.
- **close:** this method performs any necessary cleanup before closing the program.

Beside the environment, gym provides a set of wrappers to modify an existing environment without having to change the underlying code directly. The three main common things a wrapper wants to do are:

- Transform actions before applying them to the base environment
- Transform observations that are returned by the base environment
- Transform rewards that are returned by the base environment

The given set of wrappers to reach any of the aforementioned goal includes: *ActionWrapper*, *ObservationWrapper*, *RewardWrapper*. Alongside with more complex wrappers which che be found in the offical documentation. Furthermore, custom wrappers can be implemented by inheriting from *Wrapper*.

1.3 Soft Actor Critic - SAC

The soft actor critic algorithm [Haarnoja et al., 2018] illustrasted in Figure 1.3 is a state-of-the-art RL algorithm designed to outperform prior on-policy and off-policy methods in a range of continuous control benchmark tasks.

It aims and succeeds to reduce the sample complexity, since even relatively simple task can require millions of steps of data collection, and brittleness in convergence. A poor sample efficiency in deep RL methods may be due to on-policy learning since it requires new samples to be collected for each gradient step. In order to improve the sample efficiency, SAC draws on the maximum entropy framework. It introduces to the objective 1.3 an entropy maximization term which aids stochastic policies by augmenting the objective with the expected entropy of the policy:

$$J(\pi) = \mathbb{E}_{\pi}[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) + \alpha H(\pi(\cdot \mid s_t))] \tag{1.9}$$

where α is a temperature parameter that weighs the entropy term and thus controls the policy stochasticity.

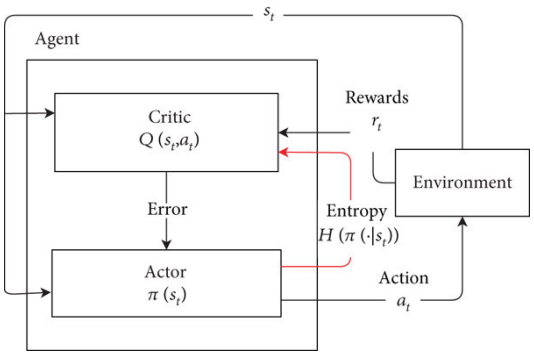

Figure 1.3. Soft actor critic

The maximum entropy framework used by SAC has several desirable properties. Firstly, it incentives a wider exploration during training. Secondly, the policy can capture multiple modes of near-optimal behavior. Lastly, it noticeably increases the learning speed over state-of-the-art methods that optimize the standard objective.

1.4 Generative Adversarial Networks - GAN

Generative Adversarial Network is a framework introduced by Goodfellow et al. [2014] for training generative models in an unsupervised fashion. GANs can be used, for example, to generate visual paragraph [Liang et al., 2017], realistic text [Zhang et al., 2017], photographs of human faces [Karras et al., 2017], Image-to-Image translation [Isola et al., 2017].

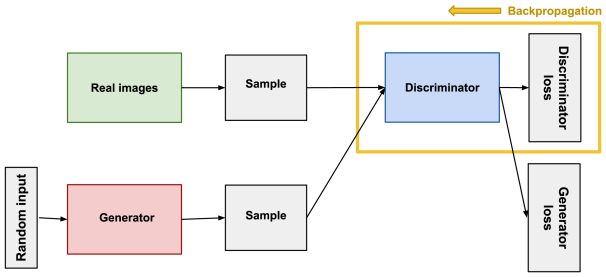

Figure 1.4. GAN diagram

The learning process involves two neural networks that are trained in an adversarial way, i.e. with an opposing objective.

Indeed, as described in Figure 1.4, the generator G generates inputs (e.g images) starting from random noise and the discriminator D needs to distinguish whether such inputs belong to the original dataset or not. GANs fall under the branch of unsupervised learning since the training process does not need labelled data as the generator is guided by the discriminator in order to generate inputs that resemble those of the original dataset.

D is trained to maximize the probability of returning the correct label to both training examples and G samples. At the same time G attempt to minimize:

$$L_G = log(1 - D(G(z))) \tag{1.10}$$

where $D(G(z))$ is the probability of $G(z)$ coming from the original dataset (p_X), then $1 - D(G(z))$ defines the probability of $G(z)$ not coming from p_X. Since the generator wants to fool the discrimitar, it needs to minimize 1.10.

In order to learn the generator's distribution p_g over the training dataset X such that $p_g \approx p_X$, a prior is defined on input noised variables $p_z(z)$, then it is mapped into the data space with the generator $G(z, \theta_g)$. Beside that, the discriminator $D(x; \theta_d)$, with $x \sim p_g$, outputs a single value which estimates the probability that x came from the dataset X rather than p_g. D and G are both differentiable function represented by a neural network with θ_d and θ_g respectively being their parameters.

In other words, the discriminator and the generator play a minimax game with the value function $V(G, D)$:

$$min_G max_D V(G, D) = \mathbb{E}_{x \sim \rho_{data}(x)}[log D(x)] + \mathbb{E}_{z \sim \rho_z(z)}[log(1 - D(G(z)))] \tag{1.11}$$

However, G is initally weak since it has not learned a good p_g yet, thus it cannot deceive the discriminator and D can say with high confidence (probability close to 1) that $G(z)$ does not come from p_X and therefore 1.10 would be at the minimum. To solve the aforementioned problem, 1.10 can be substituded with the maximization of the formula below in order to always have enough gradient:

$$log(D(G(z))) \tag{1.12}$$

1.5 CycleGAN

Image-To-Image translation is a complex task where the goal is to transform an image from one domain to another and viceversa, as shown in Figure 1.5.

Prior papers have been presented to translate images, however they often require paired training examples between the domains [Sangkloy et al. [2016], Karacan et al. [2016]]. Such paired datasets can be very expensive or even impossible to gather, as in the case of object transfiguration ($horse \leftrightharpoons zebra$).

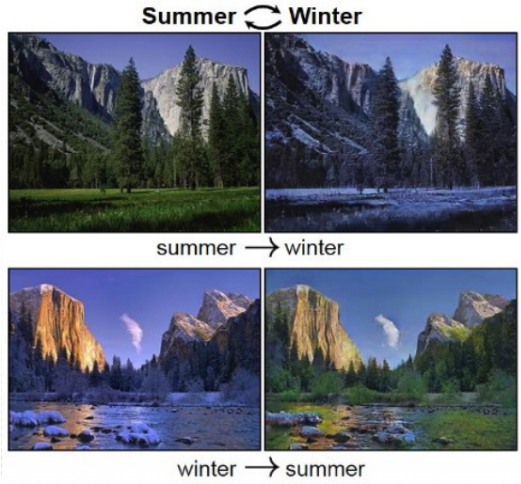

Figure 1.5. Image-to-image translation example

Cycle-Consistent Adversarial Networks from Zhu et al. [2017] (CycleGAN), aim to solve this problem in an unsupervised fashion. The main goal is to learn, using an adversarial loss, a mapping $G : X \rightarrow Y$ such that the image $G(x)$ with $x \in X$ is indistinguishable from a real image $y \in Y$. Since the mapping is highly under-costrained, an inverse mapping $F : Y \rightarrow X$ is introduced, together with a cycle-consistency loss to enforce $F(G(x)) \approx x$ and viceversa. To accomplish the goal two discriminator D_X and D_Y are provided. D_X tries to distinguish between training samples $\rho_{data}(X)$ and their translations $F(Y)$ and viceversa for D_Y. The full objective 1.13 includes the adversarial losses and the cycle-consistency loss to encourage a consistent translation from one domain to the other:

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F) \tag{1.13}$$

where the loss $L_{GAN}(G, D_Y, X, Y)$ and $L_{GAN}(F, D_X, Y, X)$ are described in 1.11 from standard GANs and the following is the cycle-consistency loss:

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim \rho_{data}(x)}[\|F(G(X)) - x\|_1] + \mathbb{E}_{y \sim \rho_{data}(y)}[\|G(F(y)) - y\|_1] \tag{1.14}$$

where λ is a temperature parameter to define the importance of such loss in Equation 1.13 and $\|\cdot\|_1$ is the L1 norm or rather the sum of the magnitudes of the vectors in a space, a measure of the distance between vectors.

1.6 AutoEncoder and Variational AutoEncoder

1.6.1 AutoEncoder

AutoEncoders (AEs) are artificial neural networks that fall under the branch of unsupervised learning since they learn efficient encoding into a latent space without the supervision of labelled data. They are generally used for vary purposes, for example, dimensionality reduction,

image compression, image denoising, image generation, feature extraction and sentence generation [Hinton and Salakhutdinov [2006], Cheng et al. [2018], Gondara [2016], Hou et al. [2017], Liu and Liu [2019]].

Taking as example the case of image dimensionality reduction, an AE is composed of two main parts, an encoder E and a decoder D.

$$E_\phi : X \to Z \qquad\qquad D_\theta : Z \to X^{'} \tag{1.15}$$

where $X = \mathbb{R}^{mxn}$ and $Z = \mathbb{R}^k$ for some m, n, k and $k \ll mxn$ to reach the goal of dimensionality reduction. The optimal case is reached when $X = X^{'}$. They are both parametrized function with ϕ and θ being respectively their parameters, to put it another way the parameters of the neural networks, generally multilayer perceptrons, of which they are composed of. As shown in Figure 1.6, the main goal of the encoder is to learn a mapping of each observation of the dataset $x \in X$ into a latent space of smaller dimensionality. Since a label is not available, in order to measure the quality of the embedded image into the latent space, the decoder is used to reconstruct to image and then compute the loss.

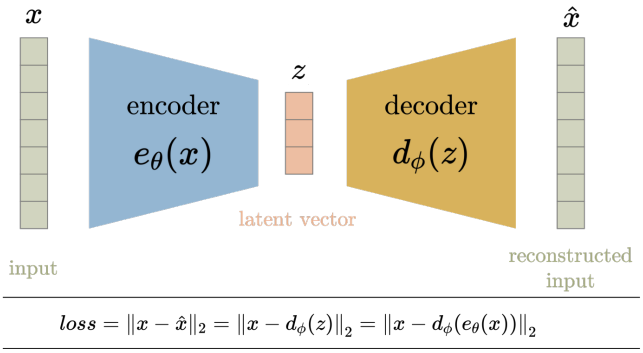

Figure 1.6. AE diagram

In other words, the encoder maps an image $x \in X$ into the latent space producing $z = E_\phi(x)$ with $z \in Z$, then z is reconstructed by the decoder to bring it back to the orginal space $x' = D_\theta(z)$ with $x^{'} \in X^{'}$. Finally, $x^{'}$ can be used as a label with any distance measure $d(x, x^{'})$. Thus the loss to be minimized is so computed:

$$L(\theta, \phi) = d(x_i, D_\theta(E_\phi(x_i))) \tag{1.16}$$

In Figure 1.7 is shown a typical example of how the MNIST dataset looks like once mapped into the latent space. After the AE is trained, a random generated observation could be given to it to generate a new sample from the dataset distribution. However, there are parts of the latent space that does not correspond to any data point. Thus using sample from the *white space* will not generate any meaningful image. That is why a ba-

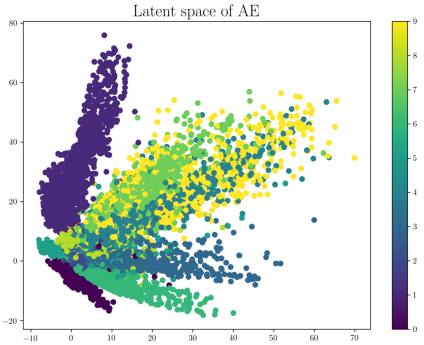

Figure 1.7. Example of an AE latent space Z on MNIST dataset

sic so-designed AE cannot be used as a general generative model, eventhough a random sample that falls into any cluster in the latent space can certainly produce a meaningful image, also an image never seen in training even if only with small variations.

AEs, results to be much more capable in data compression since the latent space of a linear autoencoder strongly resembles the eigenspace achieved during the principal component analysis of the data, with the added value of the non-linearity. Thus making AEs capable of learning rather powerful representations of the input data in lower dimensions with much less information loss.

1.6.2 Variational AutoEncoder

Variational AutoEncoders (VAEs) addresses the problem of *strong localization* of data point into the latent space thus providing a more powerful generative capability then AEs.

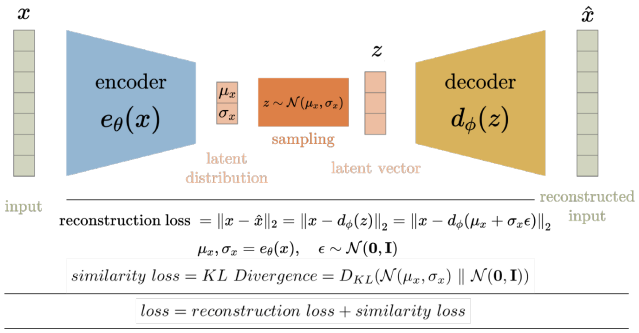

Figure 1.8. VAE diagram

As shown in Figure 1.8 only a small change with respect to AEs is introduced, the encoder instead of mapping directly samples into the latent space outputs parameters of a pre-defined distribution (usually Normal) in the latent space for every input. Then z is produced by sampling from a normal distribution with the outputted parameters.

In other words, the encoder, starting from an image $x \in X$, produces the gaussian parameters $[\mu_x, \sigma_x] = E_\phi(x)$, then z is sampled from a normal distribution $z \sim \mathcal{N}(\mu_x, \sigma_x)$. Consequently, the decoder brings it back to the orginal space $x' = D_\theta(z)$ with $x' \in X'$. Finally, x' can be used as a label with any distance measure $d(x, x')$. Thus the loss to be minimized is so computed:

$$L(\theta, \phi) = d(x_i, D_\theta(E_\phi(x_i))) + KL[\mathcal{N}(\mu_x, \sigma_x), \mathcal{N}(0, 1)] \tag{1.17}$$

where the firs term is equivalent to 1.16 and KL is the Kulback-Leibler divergence, a measure of how one probability distribution is different from another. The KL divergence acts as a regularization term by enforcing predicted distributions to be close to the identity, giving to the latent space two main properties, continuity (close points in the latent space should be close also when decoded) and completeness (any point sampled from the latent space should always be meaningful once decoded) as shown in Figure 1.9.

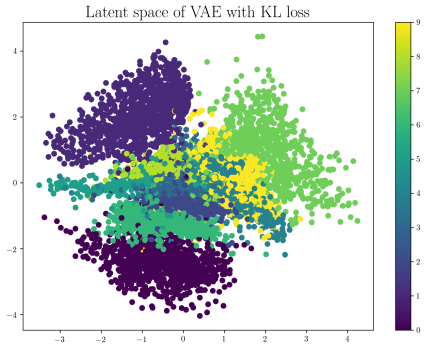

Figure 1.9. Example of a VAE latent space Z on MNIST dataset

1.7 DonkeyCar

DonkeyCar, shown in Figure 1.10, is an open source DIY platform providing software and hardware tools for the development of self-driving car algorithms. The basic car is a simple remote controlled car that can be 3D printed or bought as a kit for a reasonable and affordable price. The main models recommended by the official documentation are:

- Exceed Magnet Blue, Red
- Exceed Desert Monster Green
- Exceed Short Course Truck Green, Red
- Exceed Blaze Blue, Yellow, Wild Blue, Max Red

These cars are electrically identical but have different tires and mounting. They are also equipped with brushed motors which make ML training easier since they handle rough driving surfaces well and are inexpensive. The car can be customized with additional sensors as LIDARs and IMUs to provide more information about the surroundings of the car.

In particular, the car used for the purposes of this thesis, is a basic donkey car equipped with an 8-megapixel IMX219 sensor that features an 160 degree field of view. It is capable of taking photos with a resolution of 3280x2464 and video recording up to a resolution of 1080p at 30 frames per seconds. In order to process all the information coming from the camera, control the motors and run the self-driving car softwarethe self-driving car software it is equipped with an NVIDIA Jetson Nano microcontroller. To power comes from a LiPO battery 11.1V and 1800mAh that powers the electric motor and the microcontroller. Additionally, to expand

Figure 1.10. Assembled donkeycar

the operational life of the car, a power-bank can be added to exclusively power the micro-controller, while the LiPO battery is dedicated at powering the engine. A DonkeyCar can be remotely controlled either with a joypad or directly by the software.

Chapter 2

Related works

2.1 Describe here the related works

Appendix A

Some retarded material

A.1 It's over...

Ciaooo mbareeeeee

Glossary

Bibliography

Ethem Alpaydin. *Introduction to Machine Learning*. The MIT Press, 2014. ISBN 0262028182, 9780262028189.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Deep convolutional autoencoder-based lossy image compression. In *2018 Picture Coding Symposium (PCS)*, pages 253–257, 2018. doi: 10.1109/PCS.2018.8456308.

Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 241–246, 2016. doi: 10.1109/ICDMW.2016.0041.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL `https://arxiv.org/abs/1406.2661`.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018. URL `https://arxiv.org/abs/1801.01290`.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647. URL `https://www.science.org/doi/abs/10.1126/science.1127647`.

Xianxu Hou, Linlin Shen, Ke Sun, and Guoping Qiu. Deep feature consistent variational autoencoder. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1133–1141, 2017. doi: 10.1109/WACV.2017.131.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. 12 2016.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017. URL `https://arxiv.org/abs/1710.10196`.

Xiaodan Liang, Zhiting Hu, Hao Zhang, Chuang Gan, and Eric P. Xing. Recurrent topic-transition gan for visual paragraph generation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Danyang Liu and Gongshen Liu. A transformer-based variational autoencoder for sentence generation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2019. doi: 10.1109/IJCNN.2019.8852155.

Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color, 2016. URL `https://arxiv.org/abs/1612.00835`.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

Yizhe Zhang, Zhe Gan, Kai Fan, Zhi Chen, Ricardo Henao, Dinghan Shen, and Lawrence Carin. Adversarial feature matching for text generation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4006–4015. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/zhang17b.html`.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.