Multimedia Tools and Applications
https://doi.org/10.1007/s11042-022-13695-1

Exploiting semantic segmentation to boost reinforcement learning in video game environments

Javier Montalvo[1] · Álvaro García-Martín[1] · Jesús Bescós[1]

Received: 29 March 2022 / Revised: 29 May 2022 / Accepted: 15 August 2022

Abstract
In this work we explore enhancing performance of reinforcement learning algorithms in video game environments by feeding it better, more relevant data. For this purpose, we use semantic segmentation to transform the images that would be used as input for the reinforcement learning algorithm from their original domain to a simplified semantic domain with just silhouettes and class labels instead of textures and colors, and then we train the reinforcement learning algorithm with these simplified images. We have conducted different experiments to study multiple aspects: feasibility of our proposal, and potential benefits to model generalization and transfer learning. Experiments have been performed with the Super Mario Bros video game as the testing environment. Our results show multiple advantages for this method. First, it proves that using semantic segmentation enables reaching higher performance than the baseline reinforcement learning algorithm without modifying the actual algorithm, and in fewer episodes; second, it shows noticeable performance improvements when training on multiple levels at the same time; and finally, it allows to apply transfer learning for models trained on visually different environments. We conclude that using semantic segmentation can certainly help reinforcement learning algorithms that work with visual data, by refining it. Our results also suggest that other computer vision techniques may also be beneficial for data prepossessing. *Models and code will be available on github upon acceptance.*

Keywords Semantic segmentation · Reinforcement learning · Domain adaptation · Synthetic data

✉ Javier Montalvo
javier.montalvor@estudiante.uam.es

Álvaro García-Martín
alvaro.garcia@uam.es

Jesús Bescós
j.bescos@uam.es

[1] VPULab, Universidad Autónoma de Madrid, Ciudad Universitaria de Cantoblanco, Madrid, 28049, Spain

1 Introduction

Oftentimes, when humans try to solve problems, they usually think of similar situations that they have faced before, and in many cases, problems are more an issue of association of already known situations and possible solutions. For example, when playing a videogame we may reach a new level, with a completely different appearance for the environment and enemies we may find, but we can still understand the environment and what are the enemies, and therefore we are able to complete it.

In some way, we could say that we *semantically* understand the environment in a higher or more abstract level. The texture and appearance variability is not really useful except to identify the class of the element in the screen: enemies, obstacles, coins, etc. And then, we perform an action or another using this knowledge. Therefore, if we only used this semantic information from the game, without textures, we should be able to beat it normally. This is not the case for many AI models: as they learn how to play the game based on images from the video game, when presented with a different scenario, although it may vary only visually, features do not activate in the same way, so the AI model is not able to complete it.

Computer vision is a field of AI that focuses on gaining high-level understanding from images and videos. This high-level understanding can then be used for different purposes, such as image classification, object detection or tracking, semantic segmentation, scene understanding, etc. If we apply computer vision techniques to the frames from a video game, we can retrieve additional information from them, like for example the semantic labels for each pixel, element localization or level recognition. Our main objective is boosting the performance of an AI model that plays a video game by feeding it with complementary information obtained using computer vision, instead of by enhancing the AI model.

In this work, we use semantic segmentation as an image processing technique, in order to make different environments from a video game look the same for a reinforcement learning AI model that learns how to play the video game. We have performed our tests using the Super Mario Bros video game. As we will show later, using semantically segmented images offers some advantages in comparison to using the normal images as input when training the AI model. Our results show that semantic segmentation helps the tested reinforcement learning models to converge faster, learning to complete game-levels in less episodes and obtaining a higher average reward. It also enables transfer learning between game-levels with different appearance, and training in multiple levels at the same time.

To establish the framework for our tests, we have first prepared the model that would perform the semantic segmentation of the video game frames. As the video game has no ground truth for semantic segmentation, we have built a synthetic frame generator, which we then use to create a semantic segmentation dataset. The semantic segmentation model is trained using this dataset of synthetic video game frames. Afterwards, we prepare an environment to train our reinforcement learning agent that learns how to play the Super Mario Bros video game. Finally, we include the segmentation model on the image processing part of the pipeline, so the AI model receives segmented images as input.

The paper is organized as follows. In Section 2 on Related Work, we present a review of different algorithms used for semantic segmentation and reinforcement learning, as well as information about synthetic datasets and data generation techniques used in the literature. In Section 3 we explain our framework, the different parts it is composed of, and the training and validation methodology and settings used. Finally, in the Results Section, we present our results, and our testing methodology.

2 Related work

In this work, we combine three different topics in AI research: Semantic Segmentation, Synthetic Data generation, and model-free Reinforcement Learning. Our objective is to increase the performance of a model-free reinforcement learning algorithm, not by changing this algorithm, but instead by using semantic segmentation to provide better, more relevant data for the reinforcement learning model to work with. Specifically, we simplify the input images to a semantic domain where all game-levels look the same, regardless of how they were in the natural RGB domain. For this purpose, we generate a synthetic semantic segmentation dataset to train our semantic segmentation model.

2.1 Semantic segmentation

A semantic segmentation of an image consists of a new image of the same size in which every pixel is assigned a value or a label that indicates the semantic class (e.g., ground, sky, grass) of that pixel in the original image. The applications of semantic segmentation are wide, from aerial imaging [22], to autonomous driving [24], brain tumor segmentation [33], and many more. In our work, we will use the semantic segmentation as an image-processing tool to obtain a simplified image from a videogame frame. In Fig. 1, we show an example of semantic segmentation, in this case using an image that resembles a frame from the Super Mario Bros videogame.

Fully Connected Networks supposed a breakthrough for semantic segmentation [17], greatly improving performance over hand-crafted methods, and since then, different deep-learning approaches have been proposed to build more accurate semantic segmentation models. For this purpose, some methods try to better utilize contextual information, like DeepLabV3 [5], a semantic segmentation model that relies on atrous convolutions [4] to control the receptive field of the network without requiring additional parameters. This model also includes what the authors called an Atrous Spatial Pyramid Pooling (ASPP) module to extract the information from different scale atrous convolutions. This information is then concatenated and fed to a 1x1 convolution, whose output is upsampled to the

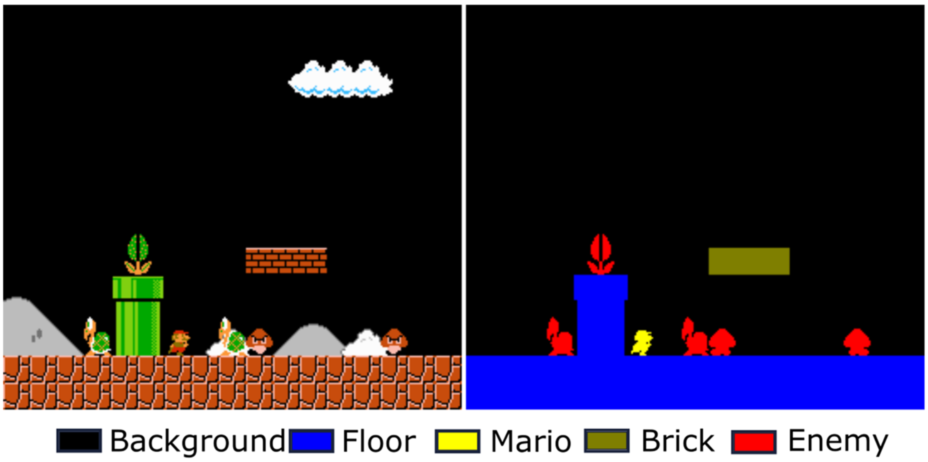

Fig. 1 Semantic Segmentation over a Super Mario Frame. In this image we have 5 different labels: Background, Enemy, Mario, Floor and Brick

required resolution. A different approach to building semantic segmentation models relies on self-attention modules to improve long-range dependencies in semantic segmentation. For example, DANet [12] presents an approach to improve semantic segmentation models by using two different attention branches after the feature extraction network, to capture information in the spatial and channel dimensions respectively. Then, features from the two branches are aggregated to produce the output. Another example of attention-based networks in semantic segmentation but in this case for RGBD images, is ACNET [14], in which the authors present what they call an Attention Complementary Module, that extracts weighted features from RGB and depth branches by using channel attention, and enables their fusion. Other attention-based methods, such as CCNet [15], use what the authors call *Criss-Cross Attention Module* that collects contextual information in both the horizontal and vertical directions to enhance pixel-wise representative capability; or ResNeSt [38] which applies channel-wise attention on different network branches to better capture cross-feature interactions and learn more diverse representations. A new approach to semantic segmentation surged after checking how well transformer-based approaches worked for image recognition with ViT [10], with models such as SETR [40], which uses the previously mentioned ViT with extra CNN decoders to enlarge feature resolution, or SegFormer [36] which performs even better than SETR while being faster and less memory intensive, by using a hierarchical architecture on the encoder capable of capturing high-resolution coarse and low-resolution fine features, and with a more compact and lightweight MLP decoder than SETR. Transformer for Transparency [39] is another example of a transformer-based semantic segmentation model which obtains state-of-the-art accuracy while being a lightweight transformer-based architecture that can be run on portable devices. Other semantic segmentation models are based on relational context approaches, such as OCR [37] which relies on an Object-Contextual Representation module, where they exploit the representation of object classes to characterize pixels.

In our work, we have decided to use the DeepLabV3 [5] model for our experiments, as it is easily available through Pytorch [27], and has been proved to perform well on different semantic segmentation tasks, such as Unsupervised Video Object Segmentation with COSNet [19] or Zero-Shot Video Object [20] detection. While we have also considered video-based semantic segmentation models that establish relationships between frames, such as [21] or [18], we decided to settle on segmentation of individual frames for simplicity.

2.2 Synthetic datasets for semantic segmentation

One of the problems of semantic segmentation is that the image labeling process required to generate ground-truth data for training is a highly time-consuming task, as it requires to manually label each pixel. For this reason, semantic segmentation datasets usually have a limited amount of images. For example, Cityscapes [7] is one of the most common semantic segmentation datasets, which consists of images from an urban environment taken from a vehicle, but *only* includes 5000 images. The amount of time required to manually label images for a semantic segmentation dataset has inspired the creation of synthetic datasets, that rely on simulations or even videogames to generate synthetic images and their semantic-segmentation ground truth, i.e., class labels. The advantages of these datasets is that they could, depending on the implementation and purpose, generate images taken from any point in the simulation, and looking at any other point in the simulation, so the amount of labeled images available is much higher, and can be programmed to be generated without human intervention. As a comparison, in the same style of scenarios included in the Cityscapes dataset, two of the most common synthetic datasets are Synthia [29] and GTAV [28], which contain 9400 and 24966 images respectively, twice and five times the size of Cityscapes.

While in this paper we focus on using synthetic data for semantic segmentation, synthetic data can also be useful for other tasks. For example, Carla [11] is a simulator with maps that resemble real world environments, and its main purpose is the development, training and validation of autonomous driving systems, but it is also used to generate synthetic datasets for aerial semantic segmentation like BEVSEG-Carla [25], for LiDAR imaging and semantic segmentation like in Paris-CARLA-3D [9] (which includes both real and synthetic LiDAR scans) or for vehicle tracking and detection [26]. OmniScape [32] is another dataset that has omnidirectional images for two and four wheeled vehicles, with different modalities, and it was created using the previously mentioned Carla simulator and also the GTAV video game. There are other synthetic datasets that are not generated using simulators, such as Fishyscapes [1], which is a benchmark for anomaly detection in semantic segmentation using urban images that was generated by randomly overlying objects from classes not included in Cityscapes dataset over images from the validation set of Cityscapes, applying some image processing techniques to the added objects (like changes in illumination and coloring) so they look more like other objects in Cityscapes images and blend in better.

The main drawback of synthetic datasets is that there exists usually a significant domain gap between the synthetic and real images, and therefore, a model trained only with synthetic images will not be able to maintain the same level of performance on real images. Moreover, it has been proved that combining synthetic and real data together in training can be beneficial for a model, hence reaching better performance on real test images than that obtained just training with real images [3].

2.3 Reinforcement learning

Reinforcement learning is the third paradigm of machine learning, the other two being supervised and unsupervised learning. While supervised and unsupervised learning rely on using data, in reinforcement learning, models are trained using *experiences*. In general, reinforcement learning models are referred to as *agents*, and they learn how to behave in an *environment* by performing random actions, and after every action, a *reward* is received. The agent then learns how to maximize this reward, by performing the best action in each situation.

The problem with reinforcement learning algorithms is that the environment has to be set up in order to decide how rewards are going to be distributed to the agent, and this is not easy to perform outside of a simulated environment. On the other hand, many videogames are very easily modifiable to be used as environments to train and test reinforcement learning algorithms on. For this work, the environment used was the Super Mario Bros game, originally developed for the Nintendo Entertainment System, and using the library OpenAi gym [2], which has support for games of many different platforms.

We can distinguish between two different types of reinforcement learning algorithms: first, those that rely on expected future reward (or a distribution of future rewards) to decide what action to take, which are called model-based algorithms; some examples of these type of algorithms are PPO [31] or MuZero [30]. Second, those that only use a single prediction of the next reward to decide what action to perform; these are called model-free, like Q-Learning [35], Double Deep Q-Learning [34] and Implicit Quantile Networks [8] among others. In this work, we have decided to use model-free algorithms better than model-based ones as they require less computation resources.

Q-Learning [35] acts like a function that receives an action-state pair as input, and the output is the expected Q-value for each action-state pair, with this Q-value being an indicator of how good the action at that given state is. Then, the algorithm selects the pair with higher Q-value. The original algorithm (that did not rely on neural networks) implemented this with a table, which grew bigger every time a new state-action pair was discovered, and for big games could be unbearably big to explore and maintain in memory. This table is now replaced by a function approximator based on a neural network (Deep Q-Learning [23]), as it requires less memory, and does not have to perform a table search. Double Deep Q-Learning is the Deep variant of the Double Q-Learning algorithm [13], which improved the original Q-Learning algorithm by reducing the value overestimation of some actions at a given state. For this, it uses one Q-function to select the action performed and another Q-function to evaluate the state-action pair, with the former being updated normally, and the evaluation function being updated after a given number of *episodes*, with each *episode* being a execution of the environment until it ends (in our case, either when the game-level is completed or Mario dies).

Implicit Quantile Networks (IQN) [8] is similar to Deep Q-Learning, as both have a feature extraction network (a convolutional network in our case), but while in Deep Q-Learning we feed these convolutional features to a fully connected network that returns the Q-Values for each action, in IQN we feed the features to a deterministic parametric function trained to parametrize samples from these convolutional features V using a quantile threshold $\tau \in (0, 1)$, with the objective of obtaining the quantile of a target distribution at τ, for each V. During inference, the algorithm uses a fixed amount of different values for τ, computes the quantiles of the features according to them, and the action with the higher average quantiles is used.

3 Framework

Our framework is composed of three different subsystems (see Fig. 2): a synthetic dataset generator, a semantic segmentation model, and the reinforcement learning agent.

We have decided to use the Super Mario Bros video game for the reinforcement learning environment. With our dataset generator, we obtain synthetic frames that are very similar to screenshots from the actual Super Mario Bros video game, but that also have their semantic segmentation ground-truth. With this dataset we train a semantic segmentation model which is then used to transform the frames from the actual videogame to semantically segmented frames, which are then used as input for a reinforcement learning agent.

3.1 Dataset generator

To generate the synthetic frames, we have used *sprites* from the original Super Mario Bros game. *Sprites* are cutouts of the different elements that appear in the game, like enemies, Mario, blocks, etc. We have programmed a method to fill images by combining these sprites, so the images look like frames taken from the videogame. We have included some logic to the appearance and positions of the elements that may appear on the image, and then we placed them randomly within these restrictions. Additionally, we randomly select the appearance of the sprites, as there are different possible game-levels styles, such as day, night, underground or underwater game-levels, among others, and the style of the levels affects the color palette of the sprites. Finally, we generate the semantically segmented image by directly assigning each image pixel to one of the six identified abstract classes:

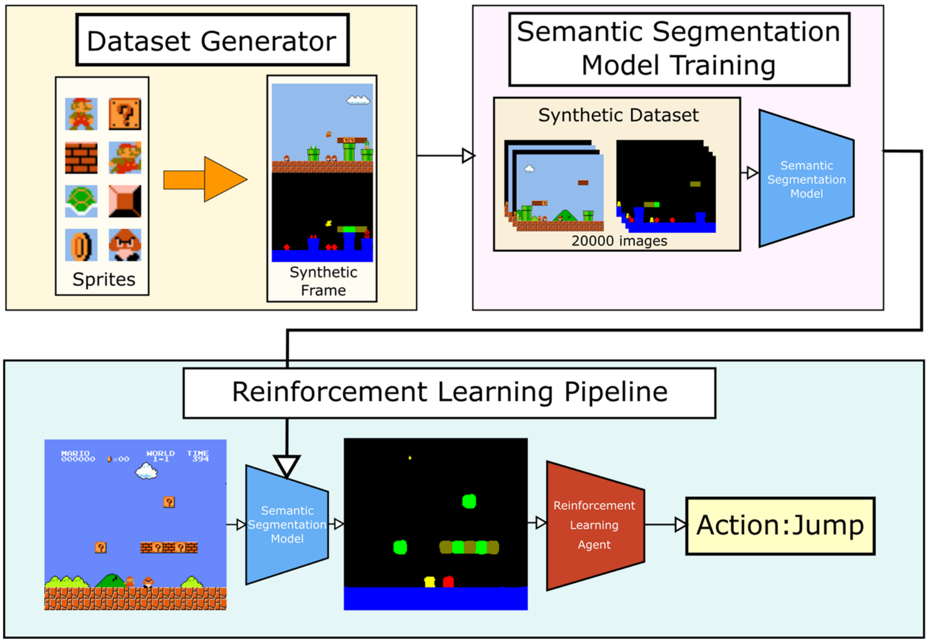
Dataset Generator
Sprites
Synthetic Frame
Semantic Segmentation Model Training
Synthetic Dataset
20000 images
Semantic Segmentation Model
Reinforcement Learning Pipeline
Semantic Segmentation Model
Reinforcement Learning Agent
Action:Jump

Fig. 2 Framework including a semantic segmentation dataset generator, a model for semantic segmentation, and the reinforcement learning pipeline

immutable objects (floor, pipes, and hard blocks), brick, question mark box, Mario, enemy, and background.

One of the advantages of this approach is that, as we are using *cutouts* of elements that appear in the video game, the domain gap between our synthetic images and the frames from the video game is very small, so model performance differences between real and synthetic frames would be reduced.

With this generator, we created a single semantic segmentation dataset for the Super Mario Bros game composed by 20.000 images of 256 by 240 pixels, that combines images in different styles to resemble the look of multiple game-levels. In Fig. 3, we display some examples of the generated synthetic frames.

Both, the dataset generator and the dataset used in our tests are available in the project github.

3.2 Semantic segmentation model

We have decided to use a DeepLabV3 [5] model to semantically segment frames from the videogame, as it shows a good balance between segmentation performance and computational efficiency, and could additionally be trained in a reasonable amount of time. For the implementation of the semantic segmentation model, we used the DeepLabV3 implementation available in PyTorch's [27] *torchvision* module. In all our semantic segmentation tests, models were trained for 45 epochs, using 18000 images for training and 2000 images for validation. We used an SGD (Stochastic Gradient Descent) optimizer with momentum, and a multi-step learning rate, which decayed by a factor of 0.1 at epochs 37 and 42. All the hyper-parameters and settings for the training process are listed on Table 1.

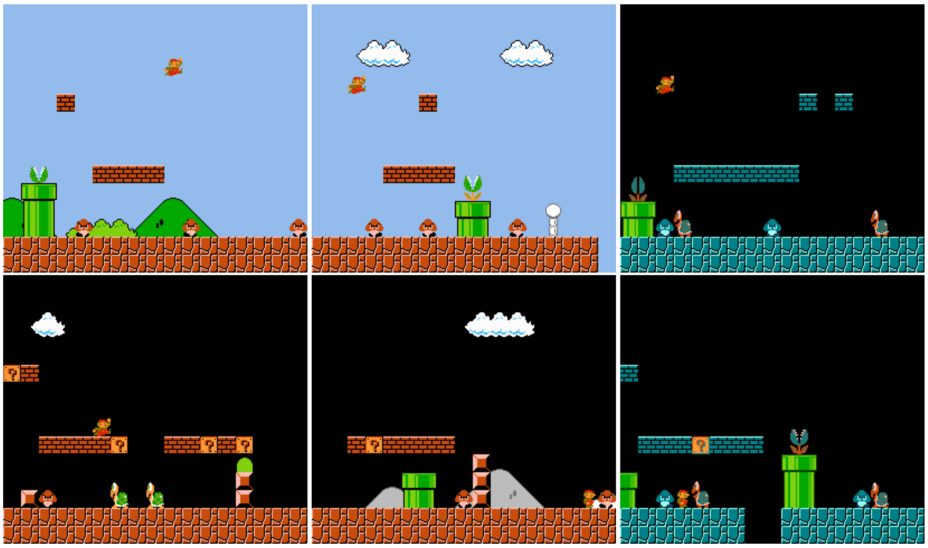

Fig. 3 Examples of synthetic frames created with our dataset generator

The DeepLabV3 model we used has three different options for the backbone: ResNet-50, ResNet-101 and MobileNetV3. In Table 2, we compare frames-per-second and mean Intersection over Union (mIoU) for all three backbones. From our results, ResNet-101 achieves the same accuracy in comparison to ResNet-50, but ResNet-50 is able to process 60.8% more frames-per-second. If we now compare the ResNet-50 backbone with the MobileNetV3 one, we find out that the latter one produces 20% more frames per second but has a 16% worse mIoU than ResNet-50. We believe the ResNet-50 backbone provides the best balance between computation efficiency and accuracy, and therefore we used this backbone in the rest of our experiments.

PyTorch's DeepLabV3 model can also be used with weights pre-trained on the COCO2017training dataset [16]. In order to decide whether to use these pre-trained weights or training from scratch, we have conducted a performance comparison. The model with

Table 1 Training hyperparameters and settings for the Semantic Segmentation model

Training Hyperparameters	Value
Learning Rate	0.001
Epochs	45
LR Decay at Epochs	37 and 42
LR Decay Factor	0.1
Optimizer Settings	Value
Momentum	0.9
Decay	10^{-4}
Other Settings	Value
Batch Size	16
Seed	271828

Table 2 Efficiency and Performance comparison between DeepLabV3 backbones on our Synthetic Dataset

Backbone	Fps	mIoU (%)
MobileNetV3	66.7	70.98
ResNet-50	55.5	87.88
ResNet-101	34.5	87.63

pre-trained weights only uses them as an starting point, and during training, no model parameters are frozen. From our testing, we conclude that the model that uses pre-trained weights is able to outperform the model that was trained from scratch by about 2.5% mIoU (from 85.24% to 87.88%), and therefore we decided to train using pre-trained weights as our starting point.

From the results obtained in these tests, we have decided to use DeepLabV3 with the ResNet-50 backbone and with pretrained weights as our final model. In Table 3, we show per-class performance of the final model in the evaluation portion of our synthetic dataset. To test our Semantic Segmentation model we have used real Super Mario Bros frames. But, as we do not have a ground truth for real frames, we performed a visual qualitative test to assert the model performance on real frames. We have selected the game-levels where the segmentation performed better to use them in our experiments, namely, game-levels 1-1, 4-1 and 6-1. In Fig. 7, we show some examples of images from the videogame and their semantically segmented counterparts for the three game-levels used in our tests.

3.3 Reinforcement learning

We have used OpenAI Gym [2] to set up our game environment. OpenAI Gym is a package that supports emulation for different video game consoles, and provides functionalities to interact with the games, like memory reading or editing or input simulation for the game. We have set up the input for the reinforcement learning model so the network receives a 6 channel image, with each channel containing a grayscale frame from the last 6 instants of time, and with a resolution of 84 by 84 pixels. The input frames for the baseline tests have been converted to grayscale, and for the segmented frames we have normalized the class label values between 0 and 255.

For the reinforcement learning algorithms we decided to use Double Deep Q-Learning [34] and IQN [8]. These reinforcement learning algorithms learn how to play the game by experience. Initially, the models perform random actions to explore in the video game environment, and store in a buffer all the input images, the actions performed and the rewards obtained during this random exploration process. This buffer is then used as the dataset to train the models, and it is updated constantly during the training process with new samples. The amount of random exploration decays as the training advances, in our case to a minimum of 2% of random action probability. During evaluation, the random action probability

Table 3 Performance Evaluation of the Semantic Segmentation model

Image	IoU (%)						
Type	Background	Floor	Bricks	*?* Blocks	Enemies	Mario	Mean
Synthetic	99.4	99.3	94.4	90.8	75.2	68.0	87.8

Table 4 Training hyperparameters and settings for the Reinforcement Learning model

Common Hyperparameters	Value
Learning Rate	0.00025
Batch Size	16
Gamma	0.90
Max exploration rate	1.0
Min exploration rate	0.02
Exploration decay	0.99
Buffer Size	4000
IQN Hyperparameters	Value
Tau	10^{-2}

is set to 0%. We have set common parameters for Double Deep Q-Learning and IQN to be equal, and the parameters exclusive to IQN have been set to the default values used by the original authors. Training episodes were set to 5000 for Reward Evolution and Repeatability Test experiments, to 9999 for the Learning in Multiple Levels experiment, and to 2500 for the Transfer Learning experiments. In Table 4 we list all parameters used by the reinforcement learning agents.

4 Results

We have performed different tests to evaluate how simplifying the input image with semantic segmentation affects the training process and the model behavior. In all tests, we denote as *baseline* the default algorithm that receives grayscale real frames as input, and as *segmented* our method that uses the simplified images with labels as input.

We have performed four different experiments, in order to test different aspects. First, we have evaluated the reward evolution during the training of a Double Deep Q-Learning algorithm, for both the baseline and segmented models to measure how our method improves the training process. Second, we have performed the same test using a different reinforcement learning algorithm IQN, to ensure repeatability across different reinforcement learning algorithms. Then, we have evaluated other advantages that our proposal provides, such as the possibility of training in multiple game-levels at the same time, and finally the ease of use for previously trained models to perform transfer learning on new game-levels, exploiting the *domain change* that semantic segmentation provides.

4.1 Reward evolution

We have first evaluated the average reward per training episode, using Double Deep Q-Learning [34] with both the baseline and the segmented models. We have trained both algorithms five times, for 5000 episodes, and to smooth the rewards per episode we have used an exponentially weighted mean.

The darker line represents the mean and the colored region represents a 95% confidence interval for the rewards.

As observed in Fig. 4, our method reaches an average reward of 2000 in about 1500 episodes, half the episodes required by the baseline (about 3000), and our method also

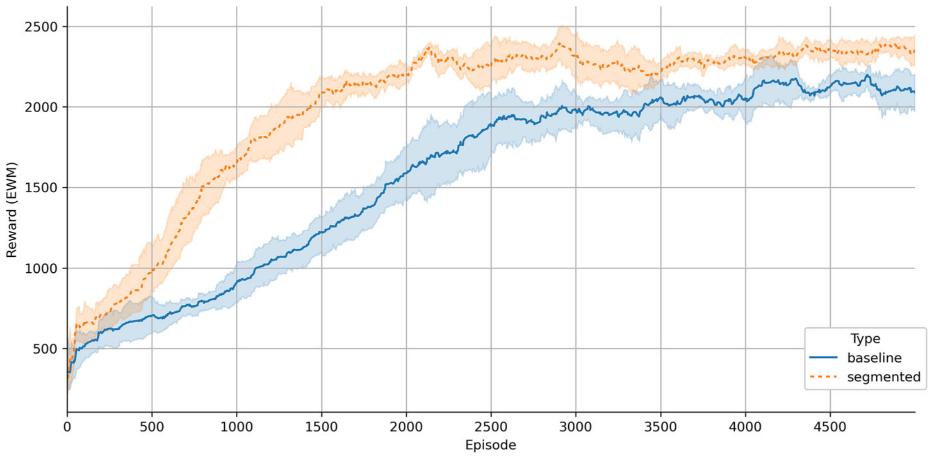

Fig. 4 Reward averages during training on game-level 1-1 using Double Deep Q-Learning [34]

reaches a higher maximum reward, about 2450 compared to the baseline which reaches 2150. These results clearly suggest that the segmented image is beneficial for the reinforcement learning algorithm, helping to train in less episodes and to achieve higher performance.

To measure the contribution of each semantic class, we have repeated this experiment by modifying the output of the semantic segmentation model so that it only includes elements of given class, re-labeling the other classes as background, in order to evaluate the impact of each class on the learning capabilities of the system.

In Fig. 5 we can see how the different classes impact performance, and it is clear that the model is not able to complete the level using isolated classes, although we can see how the Mario class reaches a higher performance level than the rest of the classes, followed by the class Floor, while the model seems to be unable to progress in the game-level at all with the rest of the isolated classes.

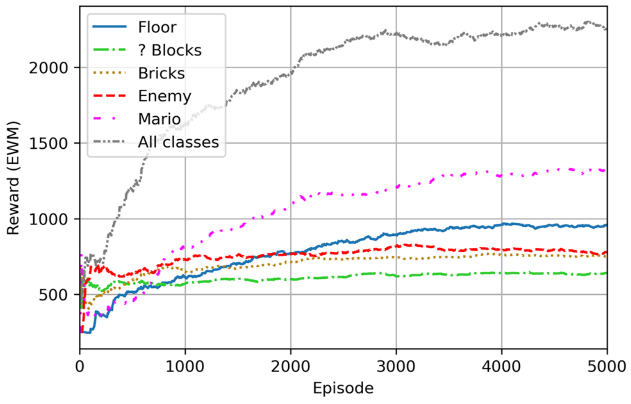

Fig. 5 Reward averages during training with isolated semantic classes. Trained on game-level 1-1

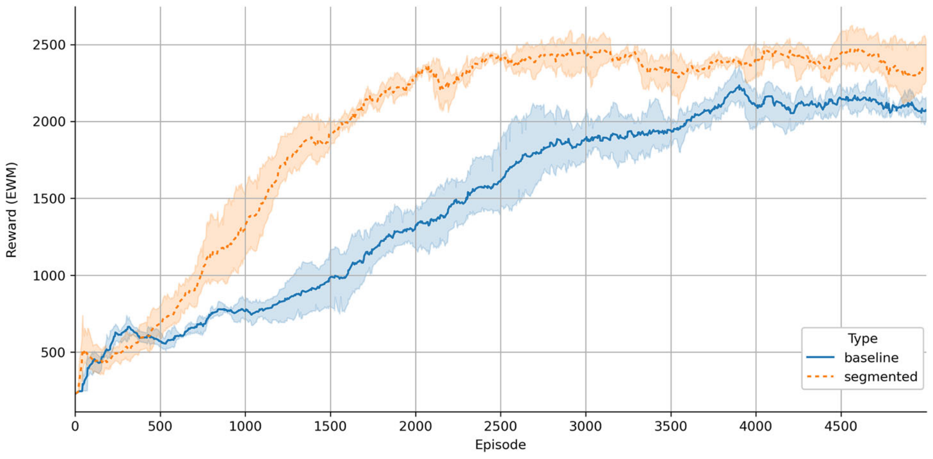

Fig. 6 Reward averages during training on game-level 1-1 using IQN [8]

4.2 Repeatability test

In order to assess repeatability of the results with other algorithms, we have also trained the models using Implicit Quantile Networks [8], in the same conditions established in the previous test.

As we can see in Fig. 6, results show the same pattern using IQN too. So, we can conclude that the visual simplification performed by the semantic segmentation does indeed help reinforcement learning models to train faster and better. Again, in Fig. 6, the darker line represents the mean and the colored region represents a 95% confidence interval for the rewards.

4.3 Training in multiple game-levels

Another advantage of using semantically simplified input images for training is that it acts as a domain adaptation method: after performing the segmentation, all images are in the same domain. Independently of how the game-level looked originally, they all share the same appearance for the network, as represented in Fig. 7.

We have performed some tests training on multiple game-levels at the same time, with the objective of obtaining a more general model. For this purpose, levels 1-1, 4-1 and 6-1 were used. Levels 1-1 and 4-1 have a similar visual style (same background, and object textures) but with different enemies and obstacles, and level 6-1 has a different background, layout, and obstacles, but shares object textures (see Fig. 7 for details).

For this test, we have trained the baseline and segmented models for 9.999 episodes each. To ensure every level was played by the models the same amount of times, we have used a round-robin approach, were on every episode a different level was played in order. So for example, in episode 1, game-level 1-1 was played; in episode 2, game-level 4-1 was played; in episode 3, game-level 6-1 was played; in episode 4, level 1-1 was played once again, and the cycle repeated during the training cycle. After the training, we have evaluated how well the model generalized, and the average rewards for each game-level.

First, we have quantified how each model performed when evaluated on both the levels used for training and on new unseen ones, to evaluate the model capability to generalize.

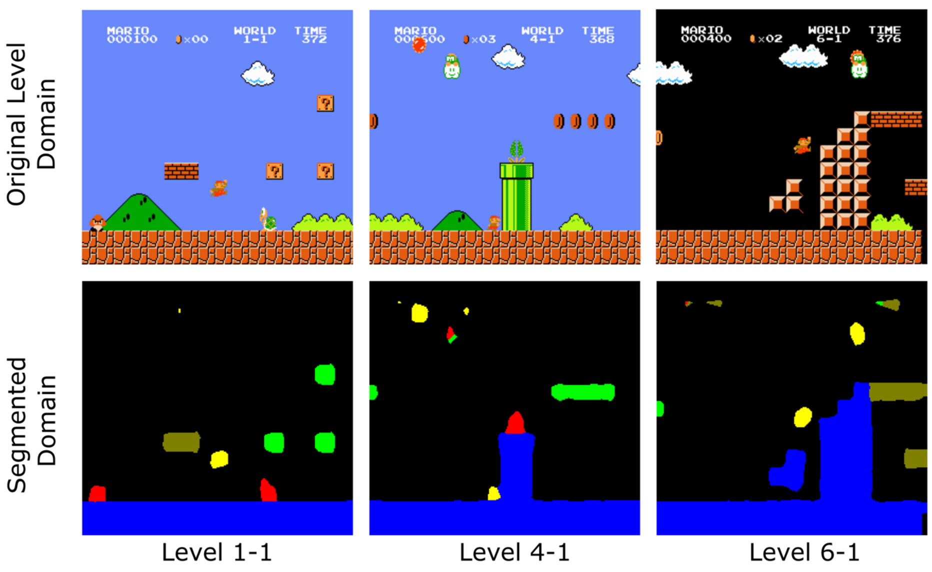

Fig. 7 Semantic Segmentation acts like a domain adaptation. Example images from game-levels 1-1, 4-1 and 6-1

Table 5 indicates that the baseline model is not able to learn how to play the three game-levels at the same time, while the model that uses semantically segmented images as input is able to complete them successfully (reaches 100% normalized reward on each game-level). This suggests that the changes in appearance between levels is a layer of additional complexity not required by the model to complete the task, but it seems to hinder its ability to learn.

Another conclusion we can extract from this experiment is that, as expected, only three game-levels are not enough for a reinforcement learning model to learn how to generalize well enough to complete unseen game-levels [6], and this is also the case while using semantic segmentation to simplify the game appearance. We believe this is for two reasons: firstly, three game-levels do not show enough variability to help the agent generalize properly, as in this game we find obstacles and challenges that are exclusive to each game-level, so the agent will not be able to learn how to solve them without specific training; secondly, in the unseen game-levels, the performance of the semantic segmentation model was not as good as on the ones it was trained on, so the artifacts generated during the segmentation may also have an impact on the agent performance.

Table 5 Generalization results, evaluated on training game-levels and new game-levels

Model	Normalized Reward On Training Levels (%)	Normalized Reward On Unseen Levels (%)
baseline	46.97	12.58
segmented	100	15.17

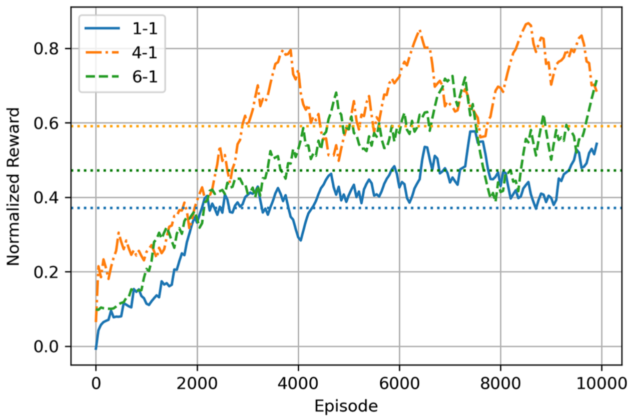

Fig. 8 Normalized per-game-level rewards while training on three levels simultaneously for the baseline configuration. Horizontal lines represent the average values for each level

Although our method is not enough to make a model generalize with three levels, we can see other interesting results from the reward graphs during the training process by performing evaluation runs every 50 episodes on the three different game-levels. First, we see the performance for the baseline agent, using grayscale real images as input.

In Fig. 8, we can see how the normalized reward increases at a different pace for each game-level, the 4-1 being the one that consistently gets a better average reward, higher than that of the other two levels. In this case, it seems like the agent is in some way overfitting to the 4-1 level, while the performance is noticeably worse for levels 1-1 and 6-1.

In Fig. 9, we have the same reward per-game-level curves but using the segmented input agent, with a much different result. In this case, the evaluation performance on each game-level increases at the same rate for all three game-levels, and the mean normalized reward is close for all of them, in contrast to the baseline. In Table 6, we can see how the rewards for the baseline model are in the range of 37.12 to 59.05%, while the rewards for the segmented

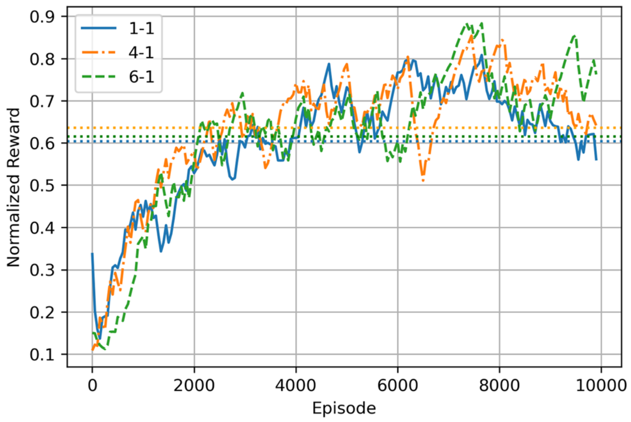

Fig. 9 Normalized per-game-level rewards while training on three levels simultaneously for the segmented configuration. Horizontal lines represent the average values for each level

Table 6 Normalized Reward Mean for each level during multi training

	Normalized Reward Mean (%)			
Model	Level 1-1	Level 4-1	Level 6-1	Average
baseline	37.12	59.05	47.13	47.76
segmented	60.33	63.54	61.47	61.78

one are in the 60.33 to 63.54% range. This suggests that our approach helps the network by preventing it to overfit to a specific level, as it seems to be happening for the baseline model.

4.4 Transfer learning

Finally, we have performed an additional study of the training process for reinforcement learning models that have been previously trained on a different game-level. For this purpose we have performed two types of experiments: transfer learning between similar looking levels and transfer learning between levels with different visuals. We have used the game-levels from the previous test: 1-1, 4-1 and 6-1 (see Fig. 7).

In Fig. 10, we compare the evolution of the reward during training in game-level 4-1 using a model previously trained on game-level 1-1. Both levels are similar in appearance, but game-level 4-1 has two new enemies (piranha plants and a flying enemy that drops shells). We can clearly see how the model trained using semantically segmented input with finetuning learns much faster and to a higher average reward than the other ones, reaching 2500 average reward around episode 550, while the baseline method with finetuning reaches it at episode 1500. In this experiment we can also see how the baseline model with finetuning reaches a higher average reward than our segmented approach trained from scratch.

In Fig. 11, we show the results of a similar experiment but now the models are pre-trained on the game-level 4-1, and then trained on the game-level 1-1. Here the pre-trained baseline model is also outperformed by the pre-trained segmentation model, and by episode 1000 it is also outperformed by the segmentation model trained from scratch. Here, our approach trained from scratch outperforms the baseline model with finetuning around episode 1000.

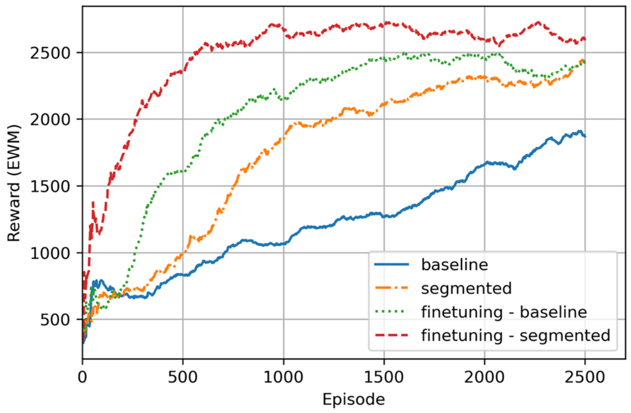

Fig. 10 Transfer learning between game-levels with similar appearance: rewards evolution training on 4-1 a model pre-trained on 1-1

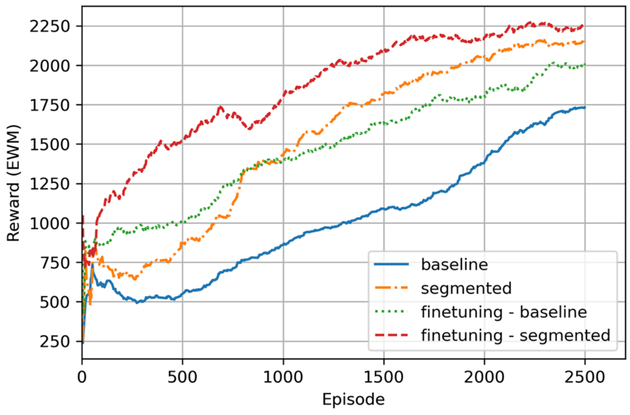

Fig. 11 Transfer learning between game-levels with similar appearance: rewards evolution training on 1-1 a model pre-trained on 4-1

Finally, we have performed the same experiment but in game-level 6-1, using reinforcement learning models previously trained on level 1-1, and trained from scratch. Levels 6-1 and 1-1 are different in appearance, as the background color changes, and so does the type of floor blocks and obstacles that appear in the level.

Figure 12, shows how using pre-trained weights for the baseline model does not give us any real advantage, as while it starts with a higher average reward by a small margin, around episode 1500 it reaches the same performance as the baseline trained from scratch, and the reward increases at a similar rate for both of them from that point on. On the other hand, the pre-trained segmented model is able to obtain the level of performance of the other three models in five times fewer episodes (about 750 instead of 2500), and reaches a higher average reward.

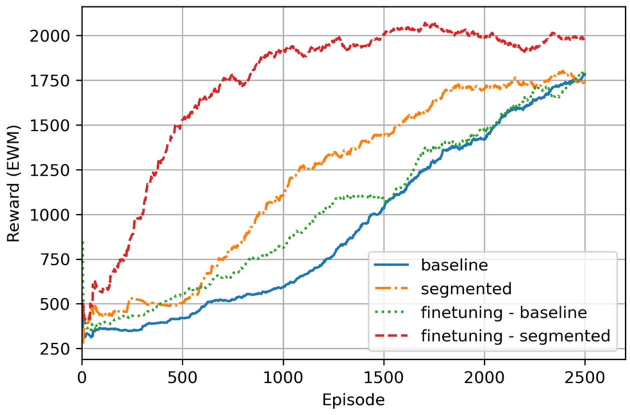

Fig. 12 Transfer learning between game-levels with different appearance: rewards evolution training on 6-1 a model pre-trained on 1-1

In the first two transfer learning experiments, while our approach with pre-trained weights is still on top, the baseline model with pre-trained weights has a similar performance to our approach trained from scratch. We believe this result is due to the similarities in appearance between the different game-levels, as shown in Fig. 7, and the results from the transfer learning test on game-level 6-1 support this reasoning, as with more visual differences the baseline with pre-trained weights does not improve the baseline trained from scratch.

We conclude that our approach enables transfer learning between environments that have different visual domains, allowing to train models for specific game-levels faster compared to training from scratch, while the baseline does not offer any advantage with transfer learning compared to training from scratch, unless game-levels are in the same visual domain.

5 Conclusions

Semantic Segmentation can be used as a tool to visually simplify the input of reinforcement learning agents to a simpler semantic-based representation which has been proved to offer multiple advantages: training in fewer episodes, reaching higher performance levels, preventing over-fitting to a specific level, and enabling transfer learning for reinforcement learning agents on environments with different visual domains. These results could potentially open new ways to optimize reinforcement learning pipelines, by just adding an image-processing step without any algorithm modifications. In a future, we plan to explore how changing the segmentation model affects reinforcement learning training, how loss weighting and importance-aware loss functions can improve the performance of the most relevant classes, and using the environment proposed by [6] with semantic segmentation to further explore model generalization.

Our method also has some limitations: as it requires images as input so they can be segmented, it can't be used on virtual environments that produce other type of data (like a chess environment, where the board status is available as an information matrix). Other disadvantages of this method include the requirement of a semantic segmentation dataset to train the segmentation model on, and the computation time taken by this model to perform the semantic segmentation, although as semantic segmentation models become better and faster, these two last disadvantages might be solved in a near future, so our results might be applicable to other environments and problems.

Acknowledgements This work is part of the preliminary tasks related to the Harvesting Visual Data (HVD) project (PID2021-125051OB-I00) funded by the Ministerio de Ciencia e Innovación of the Spanish Government.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature. This work is part of the preliminary tasks related to the Harvesting Visual Data (HVD) project (PID2021-125051OB-I00) funded by the Ministerio de Ciencia e Innovacin of the Spanish Government.

Code and Data availability We will make our code and dataset public in the github repository: https://github.com/Montyro/MarioSSRL

Declarations

Competing interests The authors declare that no conflict of interest exists in this manuscript.

References

1. Blum H, Sarlin P-E, Nieto J, Siegwart R, Cadena C (2021) The fishyscapes benchmark: measuring blind spots in semantic segmentation. Int J Comput Vis 129(11):3119–3135
2. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) OpenAI Gym
3. Chen Y, Li W, Chen X, Gool LV (2019) Learning semantic segmentation from synthetic data: a geometrically guided input-output adaptation approach. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)
4. Chen L-C, Papandreou G, Kokkinos I, Murphy K, Yuille AL (2017) Deeplab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans Pattern Anal Mach Intell 40(4):834–848
5. Chen L-C, Papandreou G, Schroff F, Adam H (2017) Rethinking Atrous convolution for semantic image segmentation
6. Cobbe K, Klimov O, Hesse C, Kim T, Schulman J (2019) Quantifying generalization in reinforcement learning. In: International Conference on Machine Learning, pp 1282–1289. PMLR
7. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding
8. Dabney W, Ostrovski G, Silver D, Munos R (2018) Implicit quantile networks for distributional reinforcement learning. In: International Conference on Machine Learning, pp 1096–1105. PMLR
9. Deschaud J-E, Duque D, Richa JP, Velasco-Forero S, Marcotegui B, Goulette F (2021) Paris-carla-3d: a real and synthetic outdoor point cloud dataset for challenging tasks in 3d mapping. Remote Sensing 13(22). https://doi.org/10.3390/rs13224713
10. Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S et al (2020) An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929
11. Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V (2017) CARLA: an open urban driving simulator. In: Proceedings of the 1st annual conference on robot learning, pp 1–16
12. Fu J, Liu J, Tian H, Li Y, Bao Y, Fang Z, Lu H (2019) Dual attention network for scene segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3146–3154
13. Hasselt H (2010) Double q-learning. Advan Neural Inform Process Syst 23:2613–2621
14. Hu X, Yang K, Fei L, Wang K (2019) Acnet: attention based network to exploit complementary features for rgbd semantic segmentation. In: 2019 IEEE International Conference on Image Processing (ICIP), pp 1440–1444. IEEE
15. Huang Z, Wang X, Huang L, Huang C, Wei Y, Liu W (2019) Ccnet: criss-cross attention for semantic segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV)
16. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: Common objects in context. In: European Conference on Computer Vision, pp 740–755. Springer
17. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440
18. Lu X, Wang W, Danelljan M, Zhou T, Shen J, Gool LV (2020) Video object segmentation with episodic graph memory networks. In: European conference on computer vision, pp 661–679. Springer

19. Lu X, Wang W, Ma C, Shen J, Shao L, Porikli F (2019) See more, know more: unsupervised video object segmentation with co-attention siamese networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)
20. Lu X, Wang W, Shen J, Crandall D, Luo J (2020) Zero-shot video object segmentation with co-attention siamese networks. IEEE transactions on pattern analysis and machine intelligence
21. Lu X, Wang W, Shen J, Crandall D, Van Gool L (2021) Segmenting objects from relational visual data. IEEE transactions on pattern analysis and machine intelligence
22. Marmanis D, Wegner JD, Galliani S, Schindler K, Datcu M, Stilla U (2016) Semantic segmentation of aerial images with an ensemble of cnss. ISPRS Annal Photogrammetry, Remote Sens Spatial Inform Sci 2016(3):473–480
23. Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing atari with deep reinforcement learning
24. Muthalagu R, Bolimera A, Kalaichelvi V (2021) Vehicle lane markings segmentation and keypoint determination using deep convolutional neural networks. Multimed Tools Appl 80(7):11201–11215
25. Ng MH, Radia K, Chen J, Wang D, Gog I, Gonzalez JE (2020) Bev-seg: bird's eye view semantic segmentation using geometry and semantic point cloud. arXiv:2006.11436
26. Niranjan DR, VinayKarthik BC (2021) Mohana: deep learning based object detection model for autonomous driving research using carla simulator. In: 2021 2nd international conference on smart electronics and communication (ICOSEC), pp 1251–1258. https://doi.org/10.1109/ICOSEC51865.2021.9591747
27. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J, Chintala S (2019) Pytorch: an imperative style, high-performance deep learning library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-buc F, Fox E, Garnett R (eds) Advances in neural information processing systems 32, pp 8024–8035
28. Richter SR, Vineet V, Roth S, Koltun V (2016) Playing for data: ground truth from computer games. In: European Conference on Computer Vision, pp 102–118. Springer
29. Ros G, Sellart L, Materzynska J, Vazquez D, Lopez AM (2016) The synthia dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3234–3243
30. Schrittwieser J, Antonoglou I, Hubert T, Simonyan K, Sifre L, Schmitt S, Guez A, Lockhart E, Hassabis D, Graepel T et al (2020) Mastering atari, go, chess and shogi by planning with a learned model. Nature 588(7839):604–609
31. Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. arXiv:1707.06347
32. Sekkat AR, Dupuis Y, Vasseur P, Honeine P (2020) The omniscape dataset. In: 2020 IEEE International conference on robotics and automation (ICRA), pp 1603–1608. IEEE
33. Sun J, Li J, Liu L (2021) Semantic segmentation of brain tumor with nested residual attention networks. Multimed Tools Appl 80(26):34203–34220
34. van Hasselt H, Guez A, Silver D (2016) Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence 30(1)
35. Watkins CJ, Dayan P (1992) Q-learning. Mach Learn 8(3):279–292
36. Xie E, Wang W, Yu Z, Anandkumar A, Alvarez JM, Luo P (2021) Segformer: simple and efficient design for semantic segmentation with transformers. Adv Neural Inf Process Syst 34
37. Yuan Y, Chen X, Wang J (2020) In: Vedaldi, A, Bischof, H, Brox, T, Frahm, J-M (eds.) Object-Contextual Representations for Semantic Segmentation, pp 173–190. Springer, Cham
38. Zhang H, Wu C, Zhang Z, Zhu Y, Lin H, Zhang Z, Sun Y, He T, Mueller J, Manmatha R et al (2020) Resnest: Split-attention networks. arXiv:2004.08955
39. Zhang J, Yang K, Constantinescu A, Peng K, Müller K, Stiefelhagen R (2021) Trans4trans: efficient transformer for transparent object segmentation to help visually impaired people navigate in the real world. In: Proceedings of the IEEE/CVF international conference on computer vision (ICCV) workshops, pp 1760–1770
40. Zheng S, Lu J, Zhao H, Zhu X, Luo Z, Wang Y, Fu Y, Feng J, Xiang T, Torr PH et al (2021) Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6881–6890