Q-Consistency Regularized VAE Facilitates Reinforcement Learning

YU Mingxin
School of Psychics
1700011374

WANG Yizhuo
School of Mathematical Science
1700010760

MAO Zihan
School of Psychics
1700011306

LU Zitong
School of Mathematical Science
1700010691

Abstract

Reinforcement learning (RL) is a generally applicable framework for finding actions bringing the maximum rewards, which is widely used in video game playing. However, when the input of RL is the each frame of the figure, there are too much useless information in them. Finding the feature of the task is a hard work for a simple network. In this paper, we propose a method called QC-VAE to encode figures into low-dimensional vectors, and combine it with an agent trained by DDPG to get the full model. We also explore fully replacing an actual image-based RL environment with a generated one, training our agent's controller only inside of the environment generated by the learned QC-VAE, which can extract useful features from the image, and apply the learned policy back to the actual environment. In evaluation of our method in a Car Racing Environment, we show that the full model offers an improvement of average score. Our code is available at `https://github.com/Mzhhh/VAEFRL`

1 Introduction

Reinforcement learning (RL) is a generally applicable framework for finding actions bringing the maximum rewards. In many RL problems, such as those modeled from video games, there are some features that will decide the reward while other features can be viewed as redundancy. However, using traditional networks, the agent will definitely unable to figure out which feature is useful for optimizing the strategy, causing much unnecessary cost in learning. In practice, we may simulate an environment for learning, in which the agent knows exactly what does the environment look like and the goal of RL is to find out the best strategy. Considering this situation, the redundancy of input is too huge to accept, leading us to think whether it can be learned that which features really matter.

In this task, we combined RL with variational autoencoder that aims to learn the latent variables, which we may consider as features extracted from the environment and thus learning from the latent variables instead of the input images.

2 Preliminaries

2.1 Reinforcement Learning and Q-Value

In Reinforcement Learning(RL), an agent learns how to act by interacting with an environment that feeds back a reward signal to the agent. The basic elements of RL are states, action, reward function

Final project of Team 10 for Bayesian theory and computation(2021 Spring), Peking University.

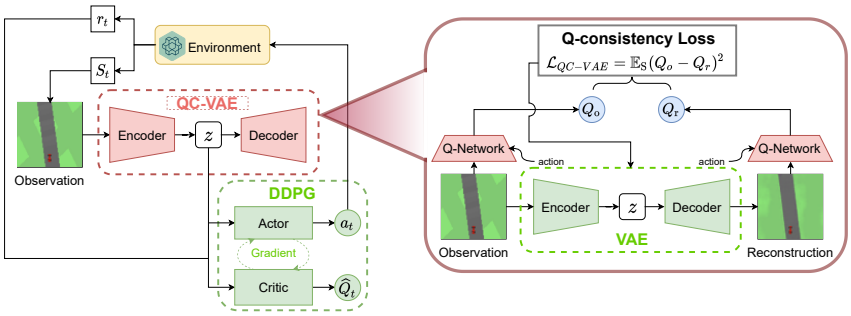

Figure 1: Overview of our method

and policy. A video game can be easily modeled as an environment in the RL setting, wherein players are modeled as agents with a finite set of actions that can be taken at each step, and the reward signal can be determined by the game score [1].

In RL, the agent relies on the reward signal. These signals can occur frequently, such as the change in score within a game, or it can occur infrequently, such as whether an agent has won or lost a game. Video games and RL go well together since most games give rewards for successful strategies.

The goal in reinforcement learning is to learn a policy that maximizes the expected cumulative discounted reward in a Markov decision process (MDP), which is defined by a tuple $(\mathcal{S}, \mathcal{A}, r, \gamma)$. $\mathcal{S}$ represents the states space, and $\mathcal{A}$ is the action space. $r(s, a)$ represents the reward function, and $\gamma \in (0, 1)$ is the discount factor.

A Q-value function (Q) shows us how good a certain action is, given a state, for an agent following a policy. Using the Bellman equation and with state (s) and action (a) as inputs, we can get the Q-Function:

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{N} \gamma^t R_t\right], \tag{1}$$

where R_t is the reward at the time t. Using the above function, we get the maximum expected future reward that the agent will get if it takes that action at that state.

Q-learning methods learn the Q-function iteratively and use exact or an approximate maximization scheme to recover the optimal policy.

2.2 Variational Autoencoder

Let $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ denote a set of input data, where n denotes the number of total input samples. The latent variables are denoted by vector $\boldsymbol{z}$. The encoder network includes network and variational parameters ϕ that produces variational probability model $q_\phi(\boldsymbol{z}|\boldsymbol{x})$. The decoder network is a probabilistic generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z})$ parameterized by θ. The estimation of log likelihood $\log p(\boldsymbol{x})$ is achieved by maximizing the Evidence Lower Bound (ELBO):

$$\mathcal{L}(\theta, \phi; \boldsymbol{x}) = \mathbb{E}_{z \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \mathrm{KL}\left[q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p(\boldsymbol{z})\right]. \tag{2}$$

The training process thus tries to optimize:

$$\max_{\theta, \phi}\left\{\sum_{i=1}^{N} \mathcal{L}(\theta, \phi; \boldsymbol{x}_i)\right\}. \tag{3}$$

3 Related Work

3.1 World Model

Ha and Schmidhuber built generative neural network models, called world models, in games to learn the state transitions of a game [2]. Instead of providing the action for a given state, the neural network

can learn to predict the next state for an action–state pair. Thus, the network is essentially learning a model of the game, which can then be used to play the game better or to perform planning.

In world model, the agent has a visual sensory component (V) that compresses what it sees into a small representative code by VAE. It also has a memory component (M) that makes predictions about future codes based on historical information. Finally, the agent has a decision-making component (M) that decides what actions to take based only on the representations created by its vision and memory components.

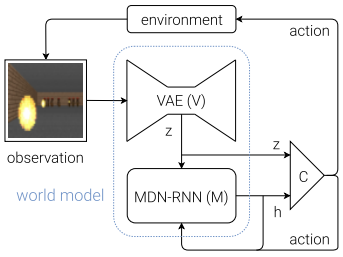

Figure 2: World Model

The role of the V model is to learn an abstract, compressed representation of each observed input frame. Specifically, it use a simple Variational Autoencoder (VAE) as V model to compress each image frame into a small latent vector z. Actually, the use of VAE inspires our method which will be introduced in the next section.

3.2 VAE with Consistency Regularization

Variational auto-encoder is a powerful approach to unsupervised learning. They enable scalable approximate posterior inference in latent-variable models. However the encoder of a VAE has the undesirable property that it maps a given observation and a semantics-preserving transformation of it to different latent representations.

Sinha and Dieng propose a regularization method to enforce consistency in VAEs, which is called CR-VAE [3]. The idea is to minimize the KL divergence between the variational distribution when conditioning on the observation and the variational distribution when conditioning on a random semantic-preserving transformation of this observation.

$$\mathcal{L}_{\mathrm{CR-VAE}} = \mathcal{L}_{\mathrm{VAE}}(x) + \mathbb{E}_{t(\widetilde{x}|x)}\left[\mathcal{L}_{\mathrm{VAE}}(\widetilde{x})\right] - \lambda \cdot \mathcal{R}(x, \phi),$$

$$\mathcal{R}(x, \phi) = \mathbb{E}_{t(\widetilde{x}|x)}\left[\mathrm{KL}\left(q_\phi(z|\widetilde{x}) \middle\| q_\phi(z|x)\right)\right],$$

where $t(\widetilde{x}|x)$ is a semantics-preserving transformation of data x (e.g. rotation or translation for imagines).

Consider the RL problem, how to encourage the VAE to learn a "good" representation that preserves task-related information? Based on this work, a consistency regularization punishment is a reliable answer. The formulation of consistency regularization in our method is modified according to our task, which can be seen in the below sections.

3.3 RL-Cycle GAN

RL-CycleGAN [4] trains a CycleGAN that encourages semantics preserving with a jointly trained RL model. Thus, the outcome of the GAN can adapt simulated images to realistic images while also preserving the original semantics relevant to the RL task. The RL task model here is a deep Q-learning network trained on simulated and real environment respectively, whose Q-functions are represented by $Q_{sim}(s, a)$ and $Q_{real}(s, a)$. The 6 images $\{x, G(x), F(G(x))\}$ and $\{y, F(y), G(F(y))\}$ derived from CycleGAN are passed to Q_{sim} and Q_{real}, giving 6 Q-values.

$$\begin{array}{lll} q_x = Q_{\text{sim}}(x, a), & q'_x = Q_{\text{real}}(G(x), a), & q''_x = Q_{\text{sim}}(F(G(x)), a), \\ q_y = Q_{\text{real}}(y, a), & q'_y = Q_{\text{sim}}(F(y), a), & q''_y = Q_{\text{real}}(G(F(y)), a). \end{array}$$

Triples $\{x, G(x), F(G(x))\}$ and $\{y, F(y), G(F(y))\}$ should each represent the same scene under Q-values, thus an RL-scene consistency loss is imposed by encouraging similar Q-values within the triple.

$$\begin{aligned}\mathcal{L}_{\text{RL-scene}}(G, F) =& d(q_x, q'_x) + d(q_x, q''_x) + d(q'_x, q''_x) \\ &+ d(q_y, q'_y) + d(q_y, q''_y) + d(q'_y, q''_y).\end{aligned}$$

While learning, both traditional loss of RL and CycleGAN and RL-scene consistency loss should be trained at the same time, each with a relative loss weight.

4 Our Method

The key for an image-based RL model is to extract task-relevant features as guidance for selecting actions under different scenarios. However, a simple generative model such as VAE is aimed at extracting features important to reconstructing input images rather than preserving the original semantics relevant to the RL task. For example with `CarRacing-v0`, a naive VAE model may produce very similar images to input images, but in the process may remove some of the objects (such as red track tiles) from the image if they only occupy a small amount of pixels in the images. However, high rewards are given when red track tiles are visited, so such alterations are detrimental to the RL task. What's more, there're also many distinctions between backgrounds (light square on grass field, etc) that do not affect the task. So we introduce QC-VAE , which trains a VAE that is encouraged to preserve task-relevant feature via a pre-trained "expert" model. Intuitively, the RL model's output should only depend on the task-specific features of the task, and constraining the VAE with the RL model encourages the VAE to preserve task-specific features.

The RL task model is a deep Q-learning network $Q(s, a)$. For a image-based task, s is the input image and a a candidate action. $Q(s, a)$ and $Q(\tilde{s}, a)$ represent Q-function applied on raw image s and reconstructed image $\tilde{s}$ respectively. The QC-VAE is trained with a pre-trained Q-estimator. And the encoder of QC-VAE is later used to train a more generalized agent solving this given task.

First, an "expert" agent for a simplified version of this task is trained. Unlike three-dimension continuous action space in original task, the action space for "expert" agent is constrained to 12 discrete actions, which drastically simplifies this task. With this "expert" agent, we train a Q-estimator $\hat{Q}(s, a)$ estimating Q-value approximately. $\hat{Q}$ is simply trained via standard TD-loss (4):

$$\hat{Q}^{\text{new}}(s_t, a_t) \leftarrow \hat{Q}(s_t, a_t) + \alpha \cdot \left(r_t + \gamma \cdot \max_a \hat{Q}(s_{t+1}, a) - \hat{Q}(s_t, a_t)\right). \tag{4}$$

To overcome the restriction of limited discrete action, a normal distribution noise is added to "expert" action during the process of training $\hat{Q}$.

Next, we'll train QC-VAE with pre-trained Q-estimator $\hat{Q}$. Apart from traditional reconstruction-loss and KL divergence loss, we also introduce a Q-consistency loss. For a sampled state s, corresponding reconstructed state $\tilde{s}$ and action a (selected by "expert" agent with normal noise), we'll estimate Q-values for both states respectively. d is some distance metric, and we use mean-squared error for the two Q-values to calculate Q-consistency loss. This loss penalizes difference between the Q-value calculated with state s and reconstructed state $\tilde{s}$, further encouraging VAE preserving the task-relevant features. And the full objective for our QC-VAE is defined in equation (5):

$$\mathcal{L}_{\text{QC}-\text{VAE}} = \lambda_{\text{QC}} \mathbb{E}_{s,a}(\widehat{Q}(s, a) - \widehat{Q}(\tilde{s}, a))^2 + \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{KL}}. \tag{5}$$

Last, we can now use our trained encoder to pre-process each frame at time t into z_t to train our agent in continuous action space, and we use DDPG in this work.

The pipeline of our full model is presented by Figure 1

5 Experiments

In this section, we describe how we train our agent model using features extracted by QC-VAE to solve a car racing task and present experiment results.

5.1 Car Racing Environment

The task we aim to solve is a continuous control task in which the agent needs to learn to drive from pixel inputs for a top-down car racing environment called `CarRacing-v0`.

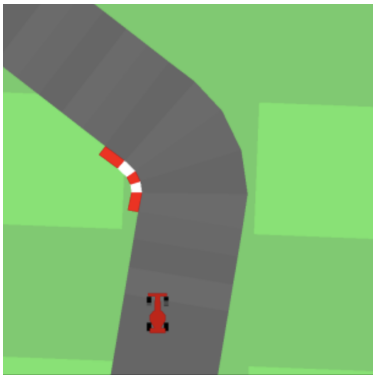

Figure 3: Agent learning to navigate in `CarRacing-v0`

In this environment, the race tracks are randomly generated for each trial, and our agent is rewarded for visiting as many tiles as possible in the least amount of time. The agent controls three continuous actions: steering left/right, acceleration, and brake, as summarized in table 1.

ACTION	RANGE
STERRING	$[-1, 1]$
ACCELERATION	$[0, 1]$
BREAK	$[0, 1]$

Table 1: Action space of `CarRacing-v0`

5.2 QC-VAE for Feature Extraction

To train our VAE network with Q-consistency (QC-VAE), we first train a Q-network that evaluates the Q-value function $Q(s, a)$ using states generated by actions a performed by an expert DQN agent. We then use the Q-network to compute Q-consistency loss defined in equation (5), which, together with reconstruction loss and KL-divergence, compose the full objective for the QC-VAE.

Further, to facilitate the training of VAEs and improve their performance, tricks listed below are employed:

- **Dropping zoom-in period**: The `CarRacing-v0` environment naturally includes a period of zooming-in in the first 50 time steps in each roll-out. The pixel distribution of images in the zoom-in period is drastically different from the timesteps afterwards, so we manually drop such period during the training of VAEs and agents.
- **KL-divergence tolerance**: The experiment of the world model [1] suggests that tolerating the KL-divergence of each dimension of z to be 0.5, and only punish the model if KL-divergence exceeds such threshold in the training of VAE may lead to better agent performance, so we also include such trick when training our QC-VAE.

To illustrate the validity of our tricks, figure 4 compares the quality of reconstructed images by VAEs trained with and without the tricks. The comparison clearly suggests that our VAE outperforms the ones trained without tricks.

We can use QC-VAE to reconstruct each frame using the encoded representation z at each timestep to visualize the quality of the information preserved by the latent representation during a roll out. Figure 5 demonstrates the original image, the latent feature z, and the reconstructed image.

[1] `https://github.com/hardmaru/WorldModelsExperiments`

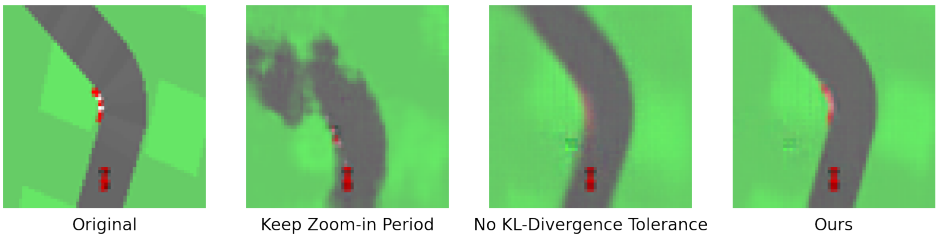

Figure 4: Comparison of VAEs with and without tricks

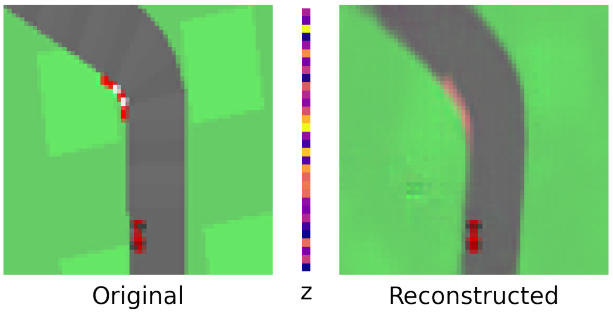

Figure 5: Demonstration of our QC-VAE

.

As expected, the latent feature z is able to preserve task-relevant information important for attaining reward, so the agent can only take latent feature z as input to guide its action decisions.

5.3 Procedure

To summarize our approach, below are the key steps taken when training out final model:

1. Train a Q-network using actions from an expert actor (DQN)
2. Collect states from the environment using actions from the same expert actor
3. Train VAE to encode the states into $z \in \mathbb{R}^{32}$, with consistency loss computed by the pretrained Q-network
4. Train DDPG agent using the latent representation encoded by VAE

MODEL	PARAMETER COUNT
CRITIC	724,033
VAE	4,349,961
AGENT	155,652

Table 2: Number of parameters in QC-VAE and corresponding DDPG agent for solving `CarRacing-v0` task.

5.4 Results

5.4.1 VAE without Q-Consistency Regularization

Training an agent to drive is not a difficult task if we have a good low-dimensional representation of the original states. Previous works have shown that with a good hand-crafted or extracted set of features, agents with a small feed-forward neural network can learn satisfactory policy in many tasks.

For this reason, we first train our VAE without Q-consistency loss, meaning that the VAE merely learns its encoding without guidance from the expert critic. We found that although the agent trained using features extracted by such VAE is able to navigate in the race track, it is outperformed by the agent with convolutional layers (CNN-DDPG, baseline) that learns directly from the images themselves, shown in Table 4.

5.4.2 QC-VAE

Typical training curves for QC-VAE with different Q-consistency regularization coefficient λ_{CR} are shown in Figure 6. QC-VAE with various λ_{CR} are all able to converge after training 2M steps. For QC-VAE, Table 3 shows significant improvement on Q-consistency as weight of Q-consistency loss increases, while maintaining stable performance on reconstruction and KL-divergence at the same time.

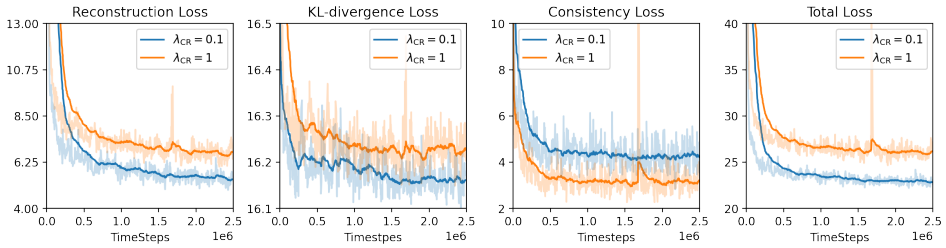

Figure 6: Training loss curves of QC-VAE with different Q-consistency regularization coefficient λ_{CR} in timesteps

λ_{QC}	Reconstruction	KL-divergence	Q-consistency
0	3.63	16.02	34.82 [2]
10^{-1}	5.46	16.16	4.33
10^{0}	6.64	16.22	3.15
10^{1}	10.35	16.40	2.55

Table 3: Effects of Q-consistency regularization coefficient λ_{CR} on QC-VAE loss.

5.4.3 Method comparison in RL environment

To demonstrate validity of our model, average score of various models are tested in Table 4. And learning curves of different agents are shown by Figure 7.

Our agent is able to achieve a score of 122.4 ± 13.9 in first 250 timesteps of 10 random trials. Previous attempt of vanilla-VAE using DDPG methods obtained average scores of 82.1 ± 30.0 range, and DDPG with same CNN and MLP layers as full model achieved average scores of 103.1 ± 22.7. The best reported solution we tested ("expert" model with discrete action) obtained an average score of 113.9 ± 26.0 over 10 random trials.

MODEL	AVG. SCORE
RANDOM POLICY	-7.7 ± 1.9
DQN EXPERT	113.9 ± 26.0
CNN-DDPG (Baseline)	103.1 ± 22.7
VANILLA-VAE-DDPG	82.1 ± 30.0
FULL MODEL (Ours)	$\mathbf{122.4} \pm 13.9$

Table 4: `CarRacing-v0` scores achieved in the first 200 steps using different models. DQN expert and CNN-DDPG works with raw image, while the last two methods works in latent space of different VAE encoders. All agents are trained with 1M timesteps.

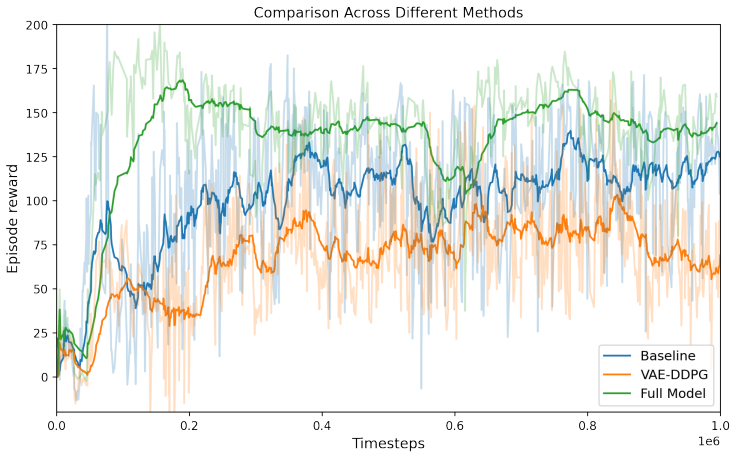

Figure 7: Learning curves of different agents

By comparing these results, we can see that though with same CNN structure, agent trained with QC-VAE and latent feature z outperforms both vanilla VAE-DDPG (without Q-consistency) and CNN-DDPG (agent not working in latent space). And our agent also outperforms discrete-action "expert" model used to train Q-estimator $\hat{Q}$, which proves that our model is not sacrificing performance when faced with much bigger action space.

6 Conclusion

In this paper, we presented the QC-VAE to encode figures into low-dimensional vectors, and showed it can significantly encourage task-related feature extraction while maintaining stable reconstruction performance. We also explore fully replacing an actual image-based RL environment with a generated one, training our agent inside the latent space generated by the learned QC-VAE, and apply the learned policy back to the actual environment.

Incorporating an Q-consistency loss along with the traditional VAE losses provides the relevant semantics for RL that must be preserved. This removes the need for task-specific feature and improves performance of agents. In evaluation of our method in a Car Racing Environment, we show that the full model offers an improvement of average score.

Acknowledgments

YU Mingxin, Wang Yizhuo, MAO Zihan and LU Zitong reviewed the literature and discussed about the method and model together. We wrote the beamer of presentation and this report together. MAO Zihan and YU Mingxin contributed to the codes and MAO Zihan finished the experiments.

References

[1] Niels Justesen et al. “Deep learning for video game playing”. In: *IEEE Transactions on Games* 12.1 (2019), pp. 1–20.

[2] David Ha and Jürgen Schmidhuber. “World models”. In: *arXiv preprint arXiv:1803.10122* (2018).

[3] Samarth Sinha and Adji B Dieng. “Consistency Regularization for Variational Auto-Encoders”. In: *arXiv preprint arXiv:2105.14859* (2021).

[4] Kanishka Rao et al. *RL-CycleGAN: Reinforcement Learning Aware Simulation-To-Real*. 2020. arXiv: 2006.09001 [cs.RO].

[5] Devanshi Dhall, Ravinder Kaur, and Mamta Juneja. “Machine Learning: A Review of the Algorithms and Its Applications”. In: *Proceedings of ICRIC 2019*. Ed. by Pradeep Kumar Singh et al. Springer International Publishing, pp. 47–63. ISBN: 978-3-030-29407-6.

[6] Samuel Alvernaz and Julian Togelius. “Autoencoder-augmented neuroevolution for visual doom playing”. In: *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE. 2017, pp. 1–8.

[7] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv preprint arXiv:1509.02971* (2015).