General response: Thanks a lot for your comments and suggestions! We will fix typos, add more details and re-organize the paper to improve clarity.

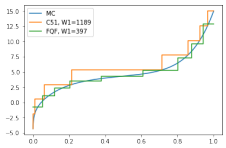

Figure 1

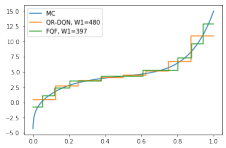

Figure 2

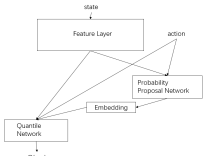

Figure 3

(A) *Toy Experiment for Distribution Approximation:* Figure 1 and 2 are toy cases of quantile function approximation using C51, QR-DQN and FQF (IQN should be viewed as sampling in a distribution instead of approximating it). W_1 loss suggests that FQF does have advantage in approximating distribution and thus achieve better estimation on values.

(B) *Experiment Details:* Figure 3 shows the general architecture of FQF. The learning rate of the probability network is set to 0.0001 but as long as its smaller than 0.001 FQF works quite well. We commit to add more experimental details in the camera-ready version. **(C)** *More Experiment under Different Settings:* We commit to compare FQF with IQN under different set of hyperparameters for ablative analysis in the camera-ready version. **(D)** *Inefficient Hyparameter:* By 'inefficient hyperparameter' we mean the atom locations (uniformly distributed between -10 and 10) in C51, quantiles locations in QR-DQN and sampled quantiles locations in IQN (it still requires a distribution to sample from). Note that FQF requires only the number of quantiles. **(E)** *State-of-Art Performance:* Thanks for pointing out [Kapturowski et al., 2019]! Yes, FQF is the best among single-actor algorithms, i.e., non-distributed methods. We will clarify this in the new version and make our claim more accurate. Besides, we believe that combining advantages of distributional RL and distributed RL is an exciting direction for further research.

To reviewer 1: (**I**) *Experiment Details:* Please refer to **(B)** in general response. (**II**) *IQN v.s. FQF:* IQN does have the advantage that it can sample as many quantiles as possible, but with the same number of quantiles FQF gives much more accurate approximations. It is not difficult to extend FQF to support arbitrary number of optimal quantiles as in IQN: we can modify the quantile network into a recurrent network so that it generates a single τ each time and takes state, action and previously outputted τs as input to generate next τ. FQF finds quantiles that can most efficiently express a distribution and computes the expectation of Q with such expression. Thus, intuitively the quantiles generated in FQF should be better than sampled quantiles (as in IQN) in terms of quantile function approximation in most cases. We leave the theoretical analysis and comparison between IQN and FQF to future work.

To reviewer 2: (**I**) *More Experiment under Different Settings:* Please refer to **(C)** in general response. We will add experiments regarding different K and N in the new version. It is a great idea to investigate what impacts the quantiles generated in FQF, and we will include the related results in the appendix of the new version as well as the results using sticky actions. (**II**) *Different Weights v.s. Uniform Weights:* We actually emailed the authors of IQN regarding the weighting scheme and tested it. We all found that using different weights or uniform weights does not really impact performance. (**III**) *The Motivation for Minimizing* L_w1: Why we minimize the square of the gradient instead of following the gradient, we agree that it should be explained. Although we cannot compute W_1, we derived its derivative for τ in the appendix and directly following the gradient is the most intuitive way to minimize W_1. Unfortunately, we found following the gradient makes the probability proposal network very hard to train at the beginning when quantile outputs are unstable. It often causes τs to converge to something like (0,0,0,1,1,1,1,1), while using square (not very different from following the gradient actually) performs much better. This may be a result of the cumulative softmax architecture. We will check if we can figure out why and explain it in detail.

To reviewer 3: (**I**) *How well the distribution is approximated/Inefficient Hyper-parameters::* Please refer to **(A)/(D)** in the general response. (**II**) *Supporting Experiments*: We agree with you on adding supporting experiments other than performance, some of those experiments are shown in the general response section and we will add more to the new version. We also agree that performance measurements on variance is important for further distributional RL research and should be included. (**III**) *Why distribution matters?* The reason why distribution matters had been studied by several previous works: 1). An Analysis of Categorical Distributional Reinforcement Learning (Rowland et al. 2018) 2). As Expected? An Analysis of Distributional Reinforcement Learning (Lyle and Bellemare, 2018). The general conclusion is that "the more complex the setting the less likely it is that distributional and expected RL algorithms will behave in the same way". So it is necessary for distributional RL algorithms to approximate the full distribution, even when we only need its mean. As shown in our toy case, FQF does achieve better distribution approximation. (**IV**) *"faster and more stable"* We apologize for the misleading choice of word in "faster and more stable". What we actually meant by 'faster' was higher sampling efficiency compared with IQN instead of training speed, and by stability we refer to smaller error bands. We will add numerical results to support this claim in the new version.