#Reviewer1. Thank you for acknowledging the key contributions of our paper. **R1.1 Novelty:** The novel practical impacts of our work are as follows. Firstly, to our best knowledge, JCL is the first that attempts to address the contrastive learning problem where infinite many of keys are considered. Secondly, by capitalizing on the proposed probabilistic model, JCL eases the aforementioned infinity scenario by simply using only a few key samples. Thirdly, JCL relies on a memory economical calculation, while its vanilla multi-key counterpart is less memory efficient when achieving similar performance (also see **R2.1 & R2.2**). We believe that all of these nontrivial efforts explore and shed further light on important properties of contrastive learning. **R1.2 Generalize to video:** As suggested, we conducted additional experiments on video data (UCF101) to evaluate the JCL pre-trained features for action recognition downstream task via linear classification. The top-1 accuracy of JCL pre-trained features is 48.6%, which outperforms MoCo v2 (47.3%). Generalization of JCL for other data modalities (sound, language, video) will be included in our future work.

#Reviewer2. We appreciate the comments. Regarding your concerns of the written quality and typos (e.g., Algorithm 1 line 8), we would carefully polish throughout the paper. **R2.1 & R2.2 Memory and accuracy:** Regarding the memory cost, the vanilla explicitly involves a batchsize that is M times larger than the conventional contrastive learning. In contrast, the batchsize for JCL remains the same as the conventional contrastive learning. Although M' positive keys are required to compute the sufficient statistics for JCL, the batchsize and the incurred multiplications (between query and key) are reduced. Particularly, JCL utilizes multiple keys merely to reflect the statistics, which helps JCL more efficiently exploit these samples. Therefore, JCL is more memory efficient than vanilla when they achieve the same performance, and JCL always offers better performance when the both cost similar memories. For fair comparison, we run the additional experiments to compare vanilla against JCL when the number of positive keys used is identical (i.e., $M = M' = 5$). The top-1 accuracy on ImageNet100 for vanilla (ResNet-50) is 80.9% while JCL achieves 82.0%. We will add the discussions. **R2.3 SimCLR:** The top-5 accuracy we reported (87.3%) for SimCLR was extracted from the primary Fig. B.1 in the arxiv version of SimCLR. **R2.4 Hyperparameters:** The experiments in Table1 are performed on ImageNet1K with ResNet-50, whereas all the experiments for Figure2 correspond to the training on a *subset* of ImageNet1K (ImageNet100 [37]) with ResNet-18. The detailed setting of Figure2 can be referred to Supplementary Material (Section S.3). Thus, there is no one-one correspondence between the data in Table1 and Figure2. In fact, MoCo v2 [8] uses $\tau = 0.2$, which differs from $\tau = 0.07$ used in MoCo [17]. For SimCLR, we directly extracted the results from the paper, which uses default $\tau = 0.1$ ($\tau = 0.5$ is used for CIFAR10 instead of ImageNet). **R2.5 MoCo v2 in Table 2:** The MoCo v2 paper [8] only reports the results on ImageNet1K and VOC (a relatively small dataset for object detection), while our paper instead presents the results on MS COCO (more challenging benchmark for object detection & instance segmentation). As suggested, we conducted additional experiments and evaluated MoCo v2 on MS COCO for object detection & instance segmentation tasks. The results are 37.6% (AP^{bb}) and 35.3% (AP^{mk}), which are lower than ours (38.1% AP^{bb}, 35.6% AP^{mk}). We will add the results. **R2.6 Fig.3(b):** Yes, we use $diag(\mathbf{\Sigma}_{k_i^+})$ for the histogram in Fig.3(b). **R2.7 Impact of λ:** According to Gaussian model and Eq.(8), a relatively small λ introduces reasonable variance into the mean positive keys that is beneficial for JCL, as more variants of positive key are potentially included. But when λ grows large enough, the introduced variance $\mathbf{\Sigma}_{k_i^+}$ starts to ***dominate*** the positive mean $\boldsymbol{\mu}_{k_i^+}$ value. This large λ therefore will distort the magnitude scale of positive keys and dilute the impact of negative keys in Eq.(8). Consequently, the distribution of the $\boldsymbol{k}_{i,m}^+$ and $\boldsymbol{k}_{i,j}^-$ would also significantly be distorted. When λ grows to infinity, the effect of negative keys completely vanishes owing to the overwhelming λ and the associated positive keys, and JCL has no motivation to distinguish between positive and negative keys. **R2.8 References:** Thanks for the references, we would carefully check these and discuss the differences in detail.

#Reviewer3. Many thanks for the positive comments. **#R3.1 More iterations:** As MoCo sees more samples, JCL also benefits from seeing more variants of positive keys (owing to the role of $\mathbf{\Sigma}_{k_i^+}$, see Algorithm 1) as training proceeds longer. In addition, JCL enforces intra-class invariance, and thus intrinsically offers structural advantage that is absent in MoCo, even if both run till the convergence. To verify this, we run JCL and MoCo v2 for more epochs ([200, 500]) and evaluated both of the algorithms on linear classification in a subset of ImageNet1K (ImageNet100 [37]). As shown in Figure 1, although the performance gap between the two indeed drops from 2.0% to 1.2% when the epoch number increases to 300, we observe consistent performance gains ($\sim$ 1%) offered by JCL when epoch number > 300. The results further justify the effectiveness of JCL even in longer iterations.

Figure 1: Accuracy on ImageNet100.

Epochs	200	300	400	500
MoCo v2	80.0	82.7	83.5	84.1
JCL	82.0	83.9	84.5	85.0

#Reviewer4. **R4.1 Second contribution:** Thanks. Regarding the second contribution, Line 46 means the derivations in our algorithm lead to analytic closed form of gradients that can be easily approached, e.g., by using PyTorch (although we did not explicitly present the calculation of these gradients and these could be added in the revision). Lines 47-48 indicate that the experiments in Fig. 3 justify the hypotheses in Sec. 3.3 (e.g., JCL favors a more consistent feature invariance within each instance-specific distribution). We will rephrase these to ensure clarity. **R4.2 Code:** We actually provided the link of all source codes in supplementary material. **R4.3 200 epochs:** For fair comparisons with the most recent competitors (SimCLR, MoCo, MoCo v2) under the same epochs, we follow the commonly adopted settings (200 epochs) as in [8] to pre-train our JCL. **R4.4 Writing:** Thanks, we will carefully modify the paper structure.