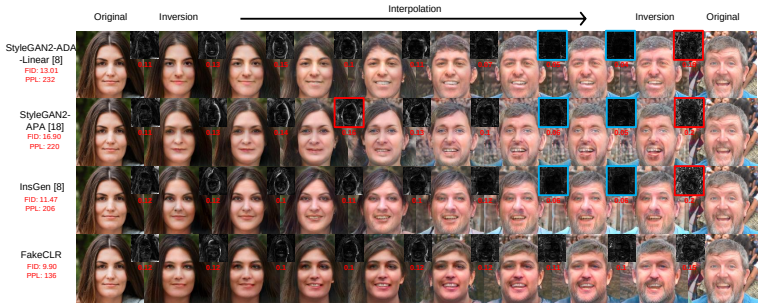


Fig. 1: Training GANs with 2K training samples (FFHQ-2K dataset) typically results in severe discontinuity in latent space, *i.e.* under-diversity interpolation on the top three rows. Compared to some reference studies, generator trained with FakeCLR show more accurate inversion, smoother latent space, more diverse interpolation, and better FID and PPL. The small grey images visualize the difference between the two face images. The red numbers are mean of pixel-wise difference, we highlighted the difference score$>$ 0.15 with red border and the difference score$\leq$ 0.6 with blue border

FakeCLR: Exploring Contrastive Learning for Solving Latent Discontinuity in Data-Efficient GANs

Ziqiang Li[1 [0000−0001−9484−2310]] ⋆, Chaoyue Wang[2], Heliang Zheng[2], Jing Zhang[3], and Bin Li[1]

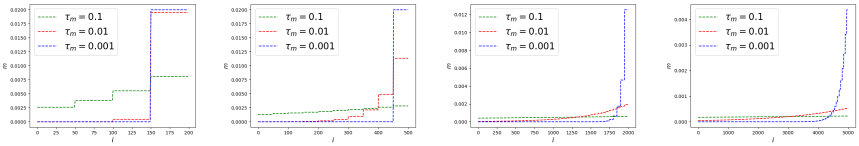
[1] University of Science and Technology of China, China
[2] JD Explore Academy, China
[3] The University of Sydney, Australia
iceli@mail.ustc.edu.cn, chaoyue.wang@outlook.com, zhenghl@mail.ustc.edu.cn, jing.zhang1@sydney.edu.au, binli@ustc.edu.cn

1 Supplementary Materials

1.1 Qualitative Results of latent space continuity on FFHQ-100 dataset

Some visual interpolation results on FFHQ-2K dataset are shown in Fig. 1.

⋆ This work was performed when Ziqiang Li was visiting JD Explore Academy as a research intern.



(a) N=200 (b) N=500 (c) N=2000 (d) N=5000

Fig. 2: Weight distribution $\mathbf{m}$ under different temperature coefficient $\tau_{\mathbf{m}}$ and queue size N, where the batch size is set to 50 (for illustrative purposes only).

1.2 The Distribution of Forgetting Factor (m)

Fig. 2 illustrates the distribution of forgetting factor $\mathbf{m}$ under different temperature coefficient $\tau_{\mathbf{m}}$ and queue size N.

1.3 Proof and Discussion of Iteration-based contrastive learning

Adding the iteration-based weight ($\mathbf{m}_i$) to negative samples leads to a iteration-based InfoNCE loss:

$$\hat{\mathcal{C}}_{F(\cdot),\phi(\cdot)}\left(\mathbf{x}_q, \mathbf{x}_k^+, \left\{\mathbf{x}_{k_i}^-, \mathbf{m}_i\right\}_{i=1}^N\right) = -\log \frac{\exp\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_k^+\right)/\tau\right)}{\exp\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_k^+\right)/\tau\right) + \sum_{i=1}^N \exp\left(\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_{k_i}^-\right) + \mathbf{m}_i\right)/\tau\right)}, \tag{1}$$

It may be useful to let:

$$\begin{aligned} Y &= \exp\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_k^+\right)/\tau\right) + \sum_{i=1}^N \exp\left(\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_{k_i}^-\right) + \mathbf{m}_i\right)/\tau\right), \\ U &= \exp\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_k^+\right)/\tau\right). \end{aligned} \tag{2}$$

Proposition 1. *The gradient of iteration-based InfoNCE loss is* $\nabla \hat{\mathcal{C}}_{F(\cdot),\phi(\cdot)} = \frac{\nabla Y \cdot U - Y \cdot \nabla U}{Y \cdot U}$. *Specifically,*

$$\begin{cases} \nabla_{\phi(\mathbf{v}_q)} \hat{\mathcal{C}}_{F(\cdot),\phi(\cdot)} = \dfrac{\sum_{i=1}^N \exp\left(\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_{k_i}^-\right) + \mathbf{m}_i\right)/\tau\right) \cdot \left(\phi\left(\mathbf{v}_{k_i}^-\right) - \phi\left(\mathbf{v}_k^+\right)\right)}{Y \cdot \tau} \\ \nabla_{\phi(\mathbf{v}_k^+)} \hat{\mathcal{C}}_{F(\cdot),\phi(\cdot)} = -\dfrac{\sum_{i=1}^N \exp\left(\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_{k_i}^-\right) + \mathbf{m}_i\right)/\tau\right) \cdot \phi\left(\mathbf{v}_q\right)}{Y \cdot \tau} \\ \nabla_{\phi(\mathbf{v}_{k_i}^-)} \hat{\mathcal{C}}_{F(\cdot),\phi(\cdot)} = \dfrac{\exp\left(\left(\phi\left(\mathbf{v}_q\right)^T \phi\left(\mathbf{v}_{k_i}^-\right) + \mathbf{m}_i\right)/\tau\right) \cdot \phi\left(\mathbf{v}_q\right)}{Y \cdot \tau} \end{cases} \tag{3}$$

Proof.

$$
\begin{aligned}
&\nabla_{\phi(\mathbf{v}_q)}\hat{\mathcal{C}}_{F(\cdot),\phi(\cdot)} = \frac{\nabla_{\phi(\mathbf{v}_q)}Y}{Y} - \frac{\nabla_{\phi(\mathbf{v}_q)}U}{U} \\
&= \frac{\exp\left(\phi\left(\mathbf{v}_q\right)^T\phi(\mathbf{v}_k^+)/\tau\right)\cdot\phi(\mathbf{v}_k^+) + \sum_{i=1}^{N}\exp\left(\left(\phi\left(\mathbf{v}_q\right)^T\phi(\mathbf{v}_{k_i}^-) + \mathbf{m}_i\right)/\tau\right)\cdot\phi(\mathbf{v}_{k_i}^-) - Y\cdot\phi(\mathbf{v}_k^+)}{Y\cdot\tau} \\
&= \frac{\sum_{i=1}^{N}\exp\left(\left(\phi\left(\mathbf{v}_q\right)^T\phi(\mathbf{v}_{k_i}^-) + \mathbf{m}_i\right)/\tau\right)\cdot\left(\phi(\mathbf{v}_{k_i}^-) - \phi(\mathbf{v}_k^+)\right)}{Y\cdot\tau} \\
&\nabla_{\phi(\mathbf{v}_k^+)}\hat{\mathcal{C}}_{F(\cdot),\phi(\cdot)} = \frac{\nabla_{\phi(\mathbf{v}_k^+)}Y}{Y} - \frac{\nabla_{\phi(\mathbf{v}_k^+)}U}{U} = \frac{U\cdot\phi\left(\mathbf{v}_q\right)}{Y\cdot\tau} - \frac{\phi\left(\mathbf{v}_q\right)}{\tau} \\
&= -\frac{\sum_{i=1}^{N}\exp\left(\left(\phi\left(\mathbf{v}_q\right)^T\phi(\mathbf{v}_{k_i}^-) + \mathbf{m}_i\right)/\tau\right)\cdot\phi\left(\mathbf{v}_q\right)}{Y\cdot\tau} \\
&\nabla_{\phi(\mathbf{v}_{k_i}^-)}\hat{\mathcal{C}}_{F(\cdot),\phi(\cdot)} = \frac{\nabla_{\phi(\mathbf{v}_{k_i}^-)}Y}{Y} - \frac{\nabla_{\phi(\mathbf{v}_{k_i}^-)}U}{U} = \frac{\exp\left(\left(\phi\left(\mathbf{v}_q\right)^T\phi(\mathbf{v}_{k_i}^-) + \mathbf{m}_i\right)/\tau\right)\cdot\phi\left(\mathbf{v}_q\right)}{Y\cdot\tau}
\end{aligned}
\tag{4}
$$

As we can see, all of the gradient of InfoNCE loss in Eq. (3) are related to iteration-based weight $\mathbf{m}_i$. Eq. (3) makes intuitive sense: (i) The proposed $\mathbf{m}_i$ improves the gradient from the positive samples. (ii) In particular, for negative samples, the proposed $\mathbf{m}_i$ improves the gradient of end-of-queue samples that are more similar to the query and can be considered as hard samples, which significantly improves the efficiency of contrastive learning.

1.4 Pseudocode of Iteration-based Contrastive Learning

We also demonstrate the PyTorch-like pseudocode in Algorithm 1.

1.5 Generated Images on FFHQ

Fig. 3 illustrates some randomly synthesized images on FFHQ dataset.

1.6 Analysis of Computation Cost

Although three strategies have been adopted in our proposed FakeCLR, they are all lightweight and only a fraction of the cost has been introduced. The cost of time on FFHQ (256×256) datasets have been demonstrated in Fig. 1.

1.7 Nearest Neighbor Test on FFHQ-100 and Obama Datasets

We show the nearest neighbor test in pixel and LPIPS spaces on FFHQ-2K and Obama datasets in Fig. 4 and Fig. 5, respectively. The results indicates that our FakeCLR has more diverse generated images and does not simply memorize the training images even give small datasets.

1.8 Comparisons on PPL and KID Metrics on FFHQ-100 and Obama Datasets

We report the PPL and KID metrics on FFHQ-100 and Obama Datasets in Table 2. Both of them demonstrate the superiority of FakeCLR.

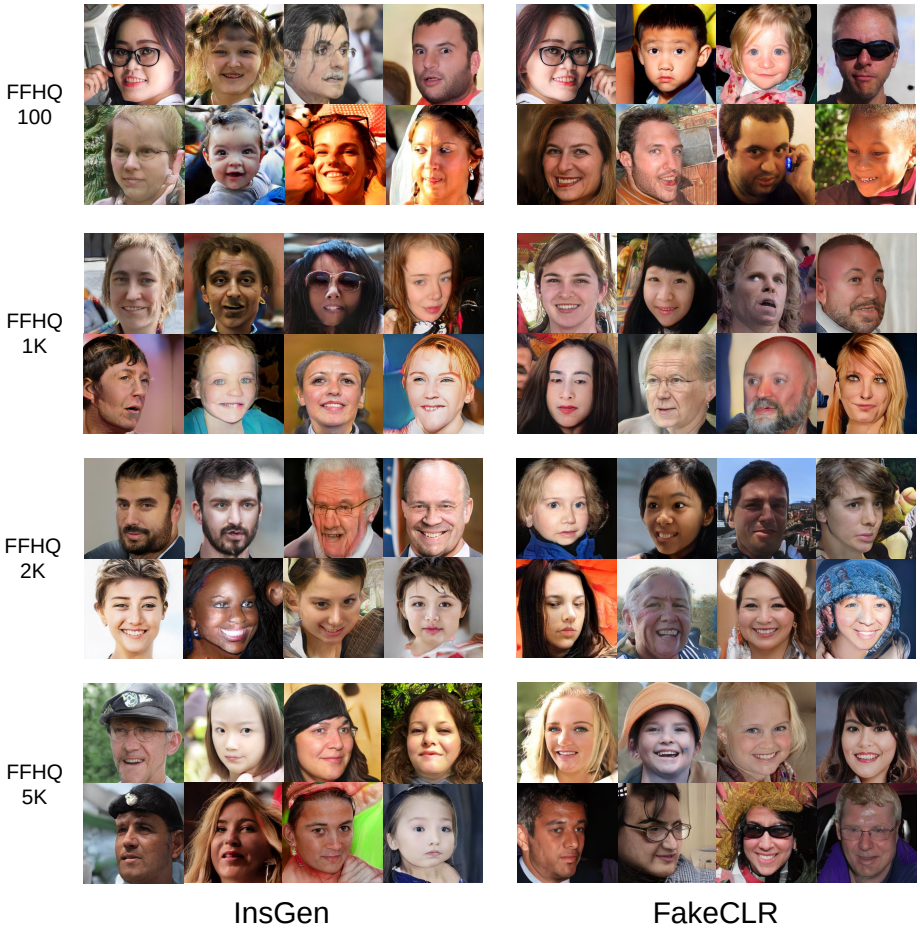


Fig. 3: Generated images with different number of training images on FFHQ dataset. All images are synthesized randomly without truncation.

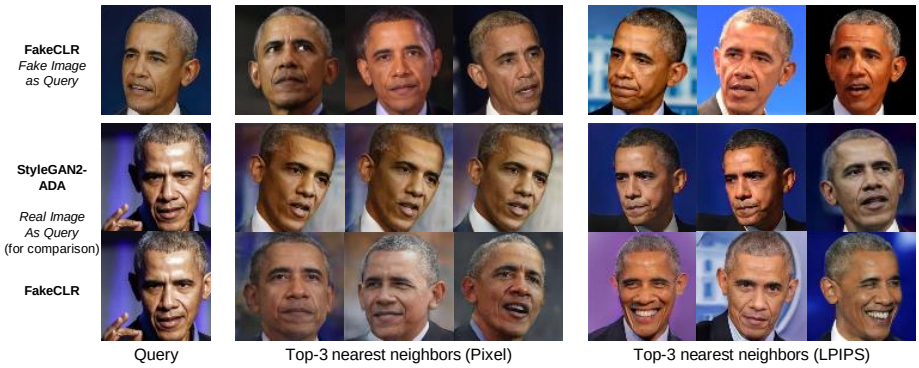


Fig. 4: Nearest neighbors in pixel space (left) and LPIPS feature space (right) on the Obama dataset. Followed as DiffAugment [1], we select fake image as query in the first row. Furthermore, we select the real image as new query for comparing with previous methods in the bottom two rows.

1.9 Generated Images on Few-shot Generation

Fig. 6 illustrates some randomly synthesized images on Obama (100), Grumpy Cat (100), Panda (100), AnimalFace Cat (160), and AnimalFace Dog (389) datasets.

1.10 Experiments Considering Path Length Regularization

Path length regularization in StyleGAN2 is another way to promote latent continuity. In this section, we consider the strength of Path length regularization in the FakeCLR. As shown in Table 3, path length regularization boosts latent continuity, yet decreases the FID a little bit. Our FakeCLR show consistent improvement under different regularization settings.

1.11 Experiments on other architectures and settings

As shown in Table 4, we conducted experiments on CIFAR-10 (using full and 20% training data) under the unconditional and conditional settings and StyleGAN2-ADA and BigGAN backbones. From both architectures and settings aspects, our FakeCLR can improve the generation performance.

1.12 What if adding real samples into the fake queue?

Table 5 presents some explorations of adding real samples into the fake queue of FakeCLR? *Instance-perturbation* is selected as the baseline for a fair comparison. In Config A, we add real images from the beginning of training, 50% real+50%

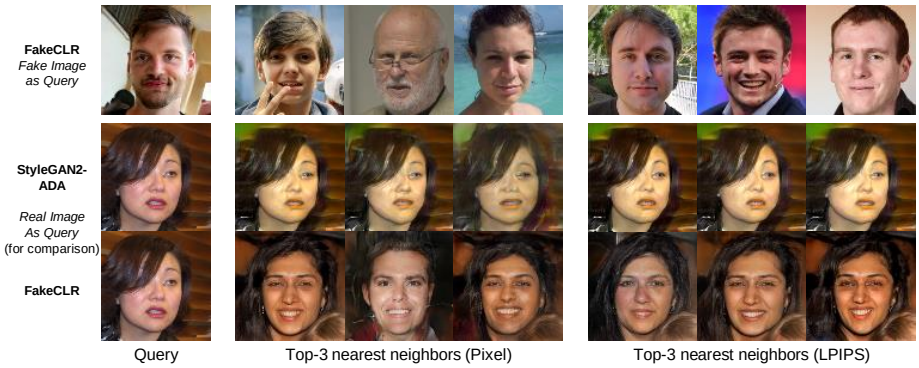


Fig. 5: Nearest neighbors in pixel space (left) and LPIPS feature space (right) on the FFHQ-100 dataset. Followed as DiffAugment [1], we select fake image as query in the first row. Furthermore, we select the real image as new query for comparing with previous methods in the bottom two rows.

fake samples form the queue. Similarly, we add real images from the half iterations of training (Config B). Experiments demonstrate that adding real samples into our queue is not conducive to the final results. The reasons could be: i) Our FakeCLR aims to solve latent space discontinuity, while adding real samples may have side effects on learning the correlation among fake samples. ii) In terms of contrastive learning, real samples can be regarded as easy negative samples for fake samples, but contrastive learning usually prefers hard negative samples.

References

1. Zhao, S., Liu, Z., Lin, J., Zhu, J.Y., Han, S.: Differentiable augmentation for data-efficient gan training. Advances in Neural Information Processing Systems **33** (2020)

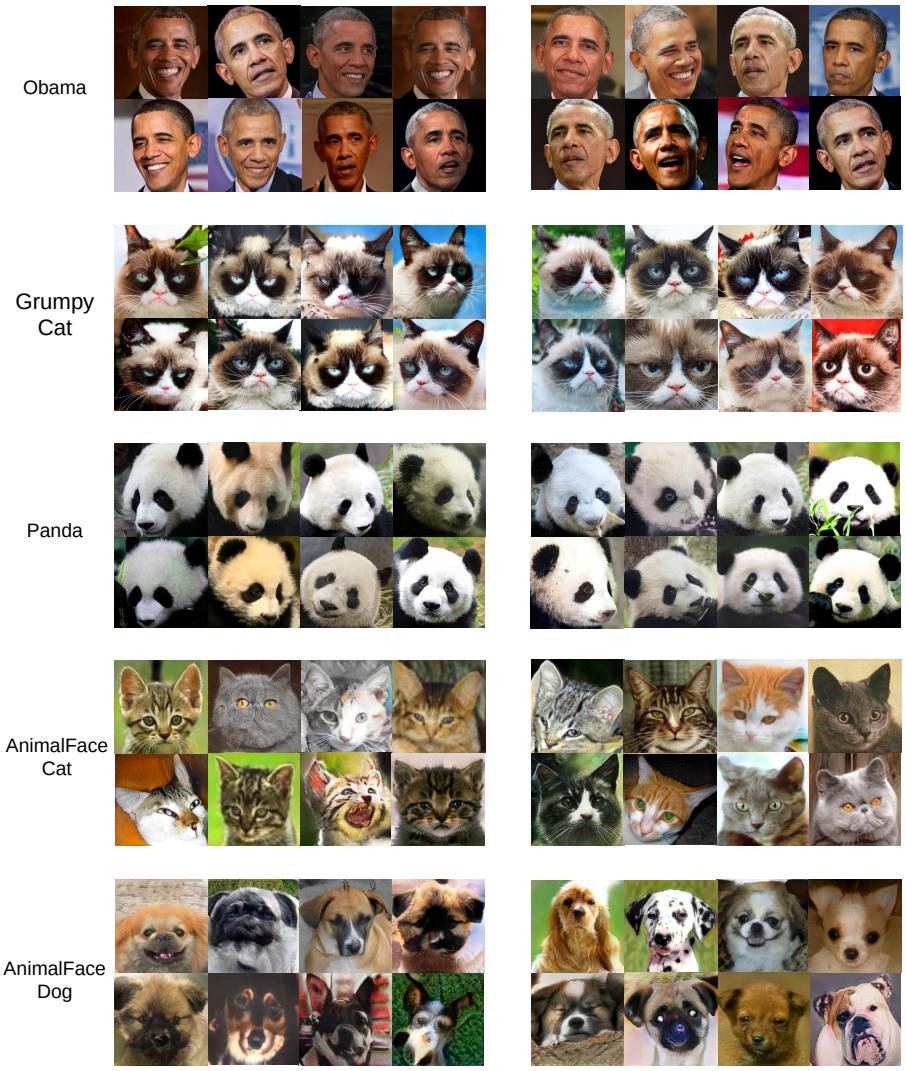


Fig. 6: Generated images with different number of training images on few-shot dataset. All images are synthesized randomly without truncation.

Algorithm 1: Pseudocode of iteration-based contrastive learning in a PyTorch-like style.

```
# f_q, f_k: encoder networks for query and key (NxC)
# queue: dictionary as a queue of K keys (CxK)
# queue_label: dictionary as a queue of K iteration-based label (1xK)
# m: momentum
# t: temperature of contrastive learning
# t_weight: temperature of iteration-based weight (0.01)
f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
    q = aug(x) # queries: NxC
    k = aug(x) # keys: NxC
    k = k.detach() # no gradient to keys

    # positive logits: Nx1
    l_pos = torch.einsum('nc,nc->n', [q, k]).unsqueeze(-1)
    # negative logits: NxK
    l_neg = torch.einsum('nc,ck->nk', [q, queue.clone().detach()])

    # iteration-based weight for negative samples
    iteration_weight=queue_label.clone().detach()
    iteration_weight=(iteration_weight-min(iteration_weight))
    /(max(iteration_weight)-min(iteration_weight))
    iteration_weight=torch.normalize(iteration_weight,p=2,dim=0)# normalize
    iteration_weight=F.softmax(iteration_weight/t_weight,dim=0)

    # iteration-weighted negative logits
    l_neg=l_neg+iteration_weight

    # logits: Nx(1+K)
    logits = cat([l_pos, l_neg], dim=1)

    # contrastive loss
    labels = zeros(N) # positives are the 0-th
    loss = CrossEntropyLoss(logits/t, labels)

    # Adam update: query network
    loss.backward()
    update(f_q.params)

    # momentum update: key network
    f_k.params = m*f_k.params+(1-m)*f_q.params

    # update dictionary
    enqueue(queue,queue_label,k) # enqueue the current minibatch
    dequeue(queue,queue_label)# dequeue the earliest minibatch
```

Table 1: The cost of time on FFHQ (256×256) datasets. All results are calculated by averaged over 10 times on two NVIDIA Tesla A100 GPUs with 64 batch size.

256×256 Resolution	sec/kimg
StyleGAN2-ADA	6.16
StyleGAN2-ADA-Linear	6.15
Instance-real	8.18
Instance-fake	10.06
InsGen	11.70
FakeCLR	10.67

Table 2: Comparison with previous methods over FFHQ-2K and Obama datasets: KID and PPL of w space are reported. Results with * are the searched best results for each dataset, which are better than InsGen with the default setting.

Methods	KID($\times 10^{-2}$)↓		PPL(w)↓	
	Obama	FFHQ-2K	Obama	FFHQ-2K
ADA[10]	1.29	0.466	843	232
APA[18]	1.12	0.562	836	220
InsGen*[8]	0.605	0.455	818	206
FakeCLR (our)	**0.296**	**0.345**	**752**	**136**

Table 3: FID and PPL on FFHQ-2K for different strength of PL regularization (All other experiments follow the default parameter PL weight=2). † indicates that we adopt the fixed queue size = 200 for a fair comparison.

Methods	FID			PPL (w)		
	2	5	10	2	5	10
InsGen [8]	12.11	13.87	12.29	199	183	180
FakeCLR†	**10.40**	**10.78**	**11.32**	**171**	**160**	**145**

Table 4: Comparisons of different architectures and settings (CIFAR-10 dataset).

Methods	Conditional		Unconditional	
	20%	full	20%	full
BigGAN-APA [18]	15.3	8.28	-	-
BigGAN-FakeCLR	13.6	7.62	-	-
StyleGAN-InsGen [8]	5.3	2.24	4.83	2.7
StyleGAN-FakeCLR	**4.28**	**2.13**	**4.6**	**2.32**

Table 5: FID on adding real samples into fake queue over FFHQ-2K dataset.

Methods	FID
Instance-perturbation (baseline)	**11.29**
Config A	12.46
Config B	13.71