Supplementary Material of Are we pruning the correct channels in image-to-image translation models

Yiyong Li[1]
haoja625@163.com
Zhun Sun†[1]
zhunsun@gmail.com
Chao Li[2]
chao.li@riken.jp

[1] Bigo Technology pte. ltd.
Singapore
[2] Tohoku University
Sendai, Miyagi, Japan
[3] Center for Advanced Intelligence Project (AIP), RIKEN
Tokyo, Japan

In this supplementary material, we provide detailed information on the following topics.

1. Proof of Lemma and Theorem (in Section 1).
2. Detailed pruning algorithm (in Section 2).
3. Detailed experiments configurations (in Section 3).
4. Additional experimental results (in Section 4).

1 Proofs

Below, we give detailed proofs of the results in the manuscript.

1.1 Detailed Definitions of IN and Relu

Instance normalization (IN) [12]. Suppose a feature map $\mathcal{X} \in \mathbb{R}^{C\times W\times H}$, then its channel-wise normalization, denoted by $\mathcal{Y} \in \mathbb{R}^{C\times W\times H}$, is calculated by

$$\mathcal{Y}_i = \frac{\mathcal{X}_i - \mu_i}{\sqrt{\sigma_i^2 + \varepsilon}}, \quad i \in [C]. \tag{1}$$

where ε denotes a positive constant to avoid the zero value of the denominator and μ_i, σ_i^2 are the mean and variance of $\mathcal{X}_i$, *i.e.*,

$$\begin{aligned} \mu_i &= \frac{1}{WH} \sum_{j\in[W],k\in[H]} \mathcal{X}(i,j,k), \\ \sigma_i^2 &= \frac{1}{WH} \sum_{j\in[W],k\in[H]} \big(\mathcal{X}(i,j,k) - \mu_i\big)^2, \end{aligned} \tag{2}$$

†Corresponding author

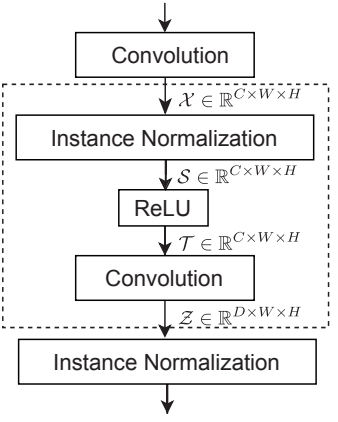

Figure 1: Illustration of the basic architecture of mage-to-image translation GANs, which are set up by concatenating the block in the dotted box. The letters beside the arrows represent the feature maps for each layer, which are used for derivation in Section 3 of the main manuscripts.

respectively.

Subsequently, two trainable variables are imposed to rescale the mean and variance of $\mathbb{Y}_i$, and the output of `IN`, denoted by $\mathcal{S} \in \mathbb{R}^{C\times W\times H}$, is thus equal to

$$\mathcal{S}_i = \gamma_i \mathcal{Y}_i + \beta_i, \quad i \in [C] \tag{3}$$

where γ_i, β_i represent the two trainable rescalars.

Rectified linear unit (`relu`) [11]. In the theoretical analysis, we use `relu` as the non-linear activation function without loss of generality. Given a feature map $\mathcal{X} \in \mathbb{R}^{C\times W\times H}$, `relu` outputs

$$\mathcal{Y}(i_1,i_2,i_3) = \texttt{relu}(\mathcal{X}(i_1,i_2,i_3)) = \begin{cases} 0 & \text{if } \mathcal{X}(i_1,i_2,i_3) < 0 \\ \mathcal{X}(i_1,i_2,i_3) & \text{otherwise} \end{cases} \tag{4}$$

for all $i_j \in [n_j], \forall j = 1,2,3$. If fixing the ith channel, we then have

$$\mathcal{Y}_i = \texttt{relu}(\mathcal{X}_i). \tag{5}$$

Note from Eq. (3) and (5) that, each channel of $\mathcal{X}$ contributes the output of `IN` and `relu` independently, that is, the ith channel of output is only determined by the ith channel of the input. It allows us to analyze the pruning perturbation in a *channel-wise* fashion.

1.2 Proof of Lemma 1

Lemma 1. *Let* $\mathbf{y}_i = vec(\mathcal{Y}_i) \in \mathbb{R}^{WH}, \mathbf{z} \in \mathbb{R}^{DWH}$ *be the vectorization of* $\mathcal{Y}_i$ *and* $\mathcal{Z}$ *respectively, and define the vectors* $\mathbf{w}_{i,j} \in \mathbb{R}^{WH}$, $i \in [C]$, $j \in [D]$, *which satisfy*

$$\mathbf{w}_{i,j}(l) = \begin{cases} \mathcal{W}(i,j,g,p) & \text{if} \quad l = (p-1)W + g \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where $l \in [WH]$. We then have the following equation hold:

$$\mathbf{z} = \sum_{i\in[C]} \underbrace{\begin{bmatrix} circ(\mathbf{w}_{i,1}) \\ circ(\mathbf{w}_{i,2}) \\ \vdots \\ circ(\mathbf{w}_{i,D}) \end{bmatrix}}_{\mathbf{G}_i:=} \texttt{relu}(\gamma_i \mathbf{y}_i + \beta_i), \tag{7}$$

where $\mathbf{G}_i \in \mathbb{R}^{DWH\times WH}$.

Proof. Follow the notations defined in the main manuscript, where $\mathcal{T}_i$ is the ith channel of the output after `relu`, by the definition of `conv`[1], the output $\mathcal{Z}$ can be rewritten as

$$\mathcal{Z} = \sum_{i\in[C]} \mathcal{Z}_i = \sum_{i\in[C]} \mathcal{W}(i,:,:,:) \star \mathcal{T}_i, \tag{8}$$

where $\mathcal{W}(i,:,:,:)$ denotes the sub-tensor of $\mathcal{W}$ by fixing the first index equal i and $\mathcal{Z}_i = \mathcal{W}(i,:,:,:,:) \star \mathcal{T}_i \in \mathbb{R}^{D\times W\times H}$ for all $i \in [C]$.

Assuming $\mathbf{h}_{i,j} = \mathbf{C}_{i,j}\mathbf{t}_i \in \mathbb{R}^{WH}$ for $i \in [C], j \in [D]$ where $\mathbf{C}_{i,j} = circ(\mathbf{w}_{i,j})$ and $\mathbf{t}_i = vec(\mathcal{T}_i)$, for all $k \in [W], l \in [H]$ the following equations obeys:

$$\begin{aligned}
&\mathbf{h}_{i,j}\left((l-1)W+k\right) && (9)\\
&= \sum_{q\in[WH]} \mathbf{C}_{i,j}\left((l-1)W+k,q\right)\mathbf{t}_i(q) && (10)\\
&= \sum_{q\in[WH]} \mathbf{w}_{i,j}\left(\left(q-k-(l-1)W\right)_{\mathrm{mod}(WH)}\right)\mathbf{t}_i(q) && (11)\\
&= \sum_{g,p\in[K]} \mathcal{W}(i,j,g,p)\mathbf{t}_i\left((l+p)W+k+g-1\right) && (12)\\
&= \left(\mathcal{W}(i,j,:,:)\star\mathcal{T}_i\right)(l,k). && (13)
\end{aligned}$$

Concatenating $\mathbf{h}_{i,j}$ for all $j \in [D]$ and using Eq. (8), we have

$$\begin{aligned}
\mathbf{z} &= \sum_{i\in[C]} \mathbf{z}_i && (14)\\
&= \sum_{i\in[C]} \begin{bmatrix} circ(\mathbf{w}_{i,1}) \\ circ(\mathbf{w}_{i,2}) \\ \vdots \\ circ(\mathbf{w}_{i,D}) \end{bmatrix} \underbrace{\texttt{relu}(\gamma_i\mathbf{y}_i+\beta_i)}_{\mathbf{t}_i=}. && (15)
\end{aligned}$$

The proof is done. □

[1] In this paper, we specify the convolution operations to be the commonly used 3D-Conv with stride 1 in vision models.

1.3 Proof of Theorem 1

First, recall the definition of the sensitivity measurement used in the manuscript; we additionally provide the formal definition of the pruning operation.

Definition 1 (sensitivity measurement)**.** *Given the kernel $\mathcal{W}$ in* `conv`*, and the variables γ, β in* `IN`*, we define the sensitivity measurement as a matrix function $F(\mathcal{W}, \gamma, \beta) : \mathbb{R}^{C\times D\times K\times K} \times \mathbb{R}^C \times \mathbb{R}^C \to \mathbb{R}^{C\times D}$, of which the (i, j)th entry $F_{i,j}$ obeys*

$$F_{i,j}(\mathcal{W}, \gamma, \beta) = \sqrt{WH}|\gamma_i| \sqrt{\sum_{p,q\in[K]} \mathcal{W}(i, j, g, p)^2} + |\beta_i| \left| \sum_{p,q\in[K]} \mathcal{W}(i, j, g, p) \right| , \tag{16}$$

where $\gamma_i, \beta_i, i \in [C]$ denotes the ith entry of γ and β, respectively.

Definition 2 (pruning the *i*th channel)**.** *To prune ith channel of the a tensor $\mathcal{A} \in \mathbb{R}^{C\times W\times H}$, is to let the queries in the ith channel be 0, that is*

$$\mathcal{A}_{pruned,j} = \begin{cases} \mathcal{A}_j & if\ j \neq j \\ \mathbf{0} & if\ j = i \end{cases} . \tag{17}$$

Remark. In the implementation, we apply a channel-wise weighting mask during training. Thus we prune a channel by setting its weighting mask to zero. After the training, We extract the compressed sub-network according to the mask.

Theorem 1. *Assume $\mathcal{Z} \in \mathbb{R}^{D\times W\times H}$ to be the output of* `conv`*, and $\mathcal{Z}^{\bar{i}}$ to denote the result by pruning the ith channel of the input. then the norm of perturbation $\Delta_i = \mathcal{Z} - \mathcal{Z}^{\bar{i}}$ is bounded by the following conditions: if $\gamma_i = 0$, then $\|\Delta_i\|_{\ell_1} = 0$; otherwise, we have*

$$\|\Delta_i\|_{\ell_1} \leq \begin{cases} WH \sum_{j\in[D]} F_{i,j}(\mathcal{W}, \gamma, \mathbf{0}), & \beta_i \geq \tau_i \\ WH \sum_{j\in[D]} F_{i,j}(\mathcal{W}, \gamma, \beta), & |\beta_i| < \tau_i \\ 0, & otherwise \end{cases} , \tag{18}$$

where $\mathbf{0} \in \mathbb{R}^C$ denotes the full-zero vector and $\tau_i = \sqrt{WH}|\gamma_i|$ for all $i \in [C]$.

Proof. Because $\mathcal{Y}_i = (\mathcal{X}_i - \mu_i)/\sigma_i$, where μ_i and σ_i are defined as in `IN`, $\mathbf{y}_i = vec(\mathcal{Y}_i)$ and $i \in [C]$, we have $\|\mathbf{y}_i\|_2 = \sqrt{WH}$. Then by Lemma 1 and consider the influence of `relu`, we

have

$$\Delta_i = \mathcal{Z} - \mathcal{Z}^{\tilde{I}} \tag{19}$$

$$= \begin{bmatrix} circ(\mathbf{w}_{i,1}) \\ circ(\mathbf{w}_{i,2}) \\ \vdots \\ circ(\mathbf{w}_{i,D}) \end{bmatrix} \texttt{relu}(\gamma_i \mathbf{y}_i + \beta_i) \tag{20}$$

$$= \begin{cases} \begin{bmatrix} circ(\mathbf{w}_{i,1}) \\ circ(\mathbf{w}_{i,2}) \\ \vdots \\ circ(\mathbf{w}_{i,D}) \end{bmatrix} (\gamma_i \mathbf{y}_i + \beta_i) & \text{if } \beta_i \geqslant \tau_i > 0 \\ 0 & \text{if } \beta_i \leqslant -\tau_i < 0 \\ sgn(\beta_i)\,|\beta_i| \begin{bmatrix} circ(\mathbf{w}_{i,1}) \\ circ(\mathbf{w}_{i,2}) \\ \vdots \\ circ(\mathbf{w}_{i,D}) \end{bmatrix} \mathbf{e} & \text{if } \tau_i = 0 \\ \begin{bmatrix} circ(\mathbf{w}_{i,1}) \\ circ(\mathbf{w}_{i,2}) \\ \vdots \\ circ(\mathbf{w}_{i,D}) \end{bmatrix} \texttt{relu}(\gamma_i \mathbf{y}_i + \beta_i) & \text{if } |\beta_i| < \tau_i \end{cases} \tag{21}$$

where $sgn(\cdot)$ denote the sign function, $\mathbf{e} \in \mathbb{R}^C$ denotes the full-one vector and $\tau_i = \sqrt{WH}|\gamma_i|$ for all $i \in [C]$.

Due to the properties of IN, we have the following equations holds:

$$\texttt{IN}(circ(\mathbf{w}_{i,j})(\gamma_i \mathbf{y_i} + \beta_i \mathbf{l})) = \texttt{IN}(circ(\mathbf{w}_{i,j})(\gamma_i \mathbf{y_i})), \tag{22}$$

$$\texttt{IN}(circ(\mathbf{w}_{i,j})(\mathbf{e})) = 0, \tag{23}$$

Therefore Eq. (21) can be further simplified as

$$\Delta_i = \begin{cases} \gamma_i \begin{bmatrix} circ(\mathbf{w}_{i,1}) \\ circ(\mathbf{w}_{i,2}) \\ \vdots \\ circ(\mathbf{w}_{i,D}) \end{bmatrix} \mathbf{y}_i & \text{if } \beta_i \geqslant \tau_i > 0 \\ 0 & \text{if } \beta_i \leqslant -\tau_i < 0 \\ 0 & \text{if } \tau_i = 0 \\ \begin{bmatrix} circ(\mathbf{w}_{i,1}) \\ circ(\mathbf{w}_{i,2}) \\ \vdots \\ circ(\mathbf{w}_{i,D}) \end{bmatrix} \texttt{relu}(\gamma_i \mathbf{y}_i + \beta_i) & \text{if } |\beta_i| < \tau_i \end{cases} \tag{24}$$

We now discuss the non-zero conditions, if $\beta_i \geqslant \tau_i > 0$, then

$$\left\| \gamma_i circ(\mathbf{w}_{i,j}) \mathbf{y}_i \right\|_{\ell_1} \leqslant WHF_{i,j}(\mathcal{W}, \gamma, \mathbf{0}), j \in [D].$$

Hence for $\beta_i \geqslant \tau_i > 0$,

$$\begin{aligned} \|\Delta_i\|_{\ell_1} &= \sum_{j \in [D]} \left\| \gamma_i circ(\mathbf{w}_{i,j}) \mathbf{y}_i \right\|_{\ell_1} \\ &\leqslant WH \sum_{j \in [D]} F_{i,j}(\mathcal{W}, \gamma, \mathbf{0}). \end{aligned} \tag{25}$$

Similarly if $|\beta_i| < \tau_i$, then

$$\|\Delta_i\|_{\ell_1} \leqslant WH \sum_{j \in [D]} F_{i,j}(\mathcal{W}, \gamma, \beta). \tag{26}$$

Taking all together, the theorem is proved. □

2 Algorithm

In this section, we formalize the algorithm of our proposed pruning method. The definition of the variables follows those in Section 3 and Section 4 in the main manuscripts. We further denote all the learnable variables in the discriminator D as $\{\Theta\}$, the training set as $\{(\mathbf{x},\mathbf{y})\}$ and the maximum number of iterations as T, then a brief description of our pruning algorithm is summarized in Algorithm 1.

Algorithm 1: BIG pruning for GAN compression

Input: $\{(\mathbf{x},\mathbf{y})\}, T, \lambda_1, \lambda_2, \rho_1, \rho_2$.
Output: $\{\mathcal{W}_{[l]}\}$, $\{\gamma_{[l]}\}$ and $\{\beta_{[l]}\}$
Init: Initialization from pre-trained model
for $1 \leqslant t \leqslant T$ **do**
  Update $\{\mathcal{W}_{[l]}\}$ by SGD
  Update $\{\gamma_{[l]}, \beta_{[l]}\}$ by Eqs. (28) and (29)
  Update $\{\Theta\}$ by SGD
end for
return $\{\mathcal{W}_{[l]}\}$, $\{\gamma_{[l]}\}$ and $\{\beta_{[l]}\}$

During the training, we update the weights of G and D alternately, where the update of kernels in G is done by stochastic gradient descent (SGD). We assign the re-scalar parameters `IN` to 0 (Eq. (28)) to prune their corresponded channels once they satisfy conditions defined in Eq. (27). Otherwise, we update them by SGD (Eq. (29)). β, γ are updated using the gradient descent method which the derivation denotes the gradient from the overall loss to the β, γ parameters.

$$
\begin{aligned}
&(\mathtt{i}) \quad \beta_{[l],i}^{(t-1)} \leq -\tau_{[l],i}^{(t-1)}, \\
&(\mathtt{ii}) \quad \gamma_{[l],i}^{(t-1)} = 0, \\
&(\mathtt{iii}) \quad \frac{\sum_{j\in[D_l]} F_{i,j}\left(\mathcal{W}_{[l]}^{(t-1)}, \gamma_{[l]}^{(t-1)}, \mathbf{0}\right)}{\sum_{i\in[C_l]} P_{l,i}\left(\mathcal{W}_{[l]}^{(t-1)}, \gamma_{[l]}^{(t-1)}, \beta_{[l]}^{(t-1)}\right)} < \rho_1, \\
&(\mathtt{iv}) \quad \frac{\sum_{j\in[D_l]} F_{i,j}\left(\mathcal{W}_{[l]}^{(t-1)}, \gamma_{[l]}^{(t-1)}, \beta_{[l]}^{(t-1)}\right)}{\sum_{i\in[C_l]} P_{l,i}\left(\mathcal{W}_{[l]}^{(t-1)}, \gamma_{[l]}^{(t-1)}, \beta_{[l]}^{(t-1)}\right)} < \rho_2.
\end{aligned}
\tag{27}
$$

where $\rho_1, \rho_2 > 0$ denotes the hyper-parameters controlling the tolerance.

$$
\gamma_{[l],i}^{(t)} = \beta_{[l],i}^{(t)} = 0 \tag{28}
$$

$$
\begin{aligned}
\gamma_{[l],i}^{(t)} &= \gamma_{[l],i}^{(t-1)} - \eta^{(t-1)} \nabla L_{all,\gamma}^{(t-1)}, \\
\beta_{[l],i}^{(t)} &= \beta_{[l],i}^{(t-1)} - \eta^{(t-1)} \nabla L_{all,\beta}^{(t-1)},
\end{aligned}
\tag{29}
$$

3 Detailed Experiments Configurations

3.1 The L_{GAN} and L_{dist}

Recall the overall loss function represented in the main scripts:

$$L_{all} = L_{GAN} + \lambda_1 L_{dist} + \lambda_2 L_{BIG}, \tag{30}$$

where L_{GAN} and L_{dist} denote the ordinary GAN loss [2] and distillation loss [13] respectively.

Specifically, we denote G, D as the generator and discriminator in GANs, respectively. Then the above two losses can be calculated as

$$L_{dist} = \mathbb{E}_{x\in\Psi}[\boldsymbol{d}(\mathsf{G}(x), \mathsf{G}_0(x))], \tag{31}$$

$$L_{GAN} = \mathbb{E}_{y\in\Omega}[log(\mathsf{D}(y))] + \mathbb{E}_{x\in\Psi}[log(1-\mathsf{D}(\mathsf{G}(x)))], \tag{32}$$

where G_0 is the unpruned large model, $\boldsymbol{d}(\cdot,\cdot)$ is perceptual loss[5]. The model distillation loss is used to enforce the pruned generator G to mimic the behaviour of the original one [1, 4, 13]. We initialize the G, D with pre-trained $\mathsf{G}_0, \mathsf{D}_0$, respectively. Where D_0 is the paired pre-trained generator with G_0.

Remark. We use the code provided by [13], which involves hybrid quantization [6, 14]. While the original code has the quantization term, we eliminate it and keep all the weights to be 32-bit floats to focus on our main contributions.

3.2 Hyper-parameters

Optimizer. We use Adam to update $\{W, \theta, \beta\}$ with $lr = 1e-8, betas = (0.5, 0.999)$ and SGD to update γ with momentum=0.5. We also use two groups of learning rates $\alpha^{(t)}$ and $\eta^{(t)}$ for updating $\{W, \theta\}$ and $\{\gamma, \beta\}$ respectively. $\alpha^{(t)}$ and $\eta^{(t)}$ are decayed using a cosine annealing scheduler.

Loss weights and Threshold. The weight of L_{dist} is fixed at $\lambda_1 = 20$ following [13]. We adjust the weight of L_{BIG}, *i.e.*, λ_2; and the thresholds for pruning channels, *i.e.*, ρ_1, ρ_2. We use 3 sets of these hyper-parameters denoted as `Stage1`, `Stage2`, `Stage3`. The details are shown in Table 1.

Remark. Indeed, recent GAN compression methods, including the sota distillation base methods, are much more sensitive to the initialization of weights rather than hyperparameters. For instance, some hyperparameters that produce the best FID scores won't converge at the beginning of training for half of the time. Therefore, the choice of hyperparameters follows the design of the method. For instance, we raise the ρ_1, ρ_2 once the current stage is converged. Meanwhile, our method works for a border range of λ terms (several magnitudes larger or smaller).

Table 1: Multi-stage trick

Stage	# Chan.	λ_1	$\lambda_2(\times 10^{-5})$	$\rho_1(\times 10^{-3})$	$\rho_2(\times 10^{-2})$	l_{rate}
1	2700	20	0.4	0.10	0.10	1e-6
2	2000	20	1.25	0.15	0.15	1e-5
3	650	20	1.25	1.25	1.25	1e-5

4 Additional Experimental Results

4.1 Additional numerical results

We provide more numerical results on the generation tasks using the reverted `zebra-to-horse` and `winter-to-summer` datasets. The improvement shown in Table 2 is consistent with others in the main manuscripts.

Table 2: Additional results on `zebra-to-horse` and `winter-to-summer` datasets.

Model	`zebra-to-horse`	`winter-to-summer`
CycleGAN	148.8	73.3
GS32	151.8	74.7
Ours	140.1	73.1

4.2 Additional zero-shot model pruning

In this section, we conduct additional zero-shot model pruning experiments to examine the correctness of our error model defined in Eq.(16). We use magnitude-based weight pruning(MBWP)[7] and filter pruning via geometric median(FPGM)[3]. MBWP[7] measure the relative importance of a filter in each block by calculating the sum of its absolute weights $\sum_{i\in[C]}\sum_{p,q\in[K]}|\mathcal{W}(i,j,g,p)|$. FPGM calculates the Geometric Median (GM)[3] of the filters within the same block. According to the characteristics of GM, the filter near it can be represented by the remaining ones. The numerical results are shown in Table 3, and the visual comparison results are collectively displayed in Figure 2. We can observe that these former proposed methods prune filters with actually high impacts on the visual effects. The correlativity (`eff`) of these methods to the sub-optimal solution (sub-net) is far low than the 0.9291 achieved by BIGP.

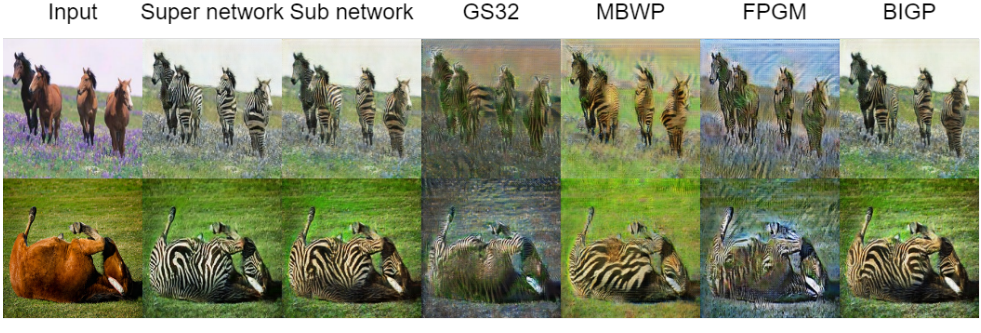

Figure 2: Examples of zero-shot pruning results on the `horse2zebra` dataset. The methods are annotated above the images, each row stands for a different sample.

Table 3: Quantitative evaluation of zero-shot pruning results on the `horse2zebra` dataset. The super-/sub-net are unpruned and reference small models from [8]. ↑ / ↓ means the larger/smaller value, the better compression performance.

Method	FLOPs	Memory	FID(↓)	`eff`(↑)
super-net	4.59G	2.1MB	60.10	—
sub-net	2.35G	1.5MB	68.77	1.0
GS32[13]	2.35G	1.5MB	188.56	0.5333
MBWP[7]	2.35G	1.5MB	138.90	0.5625
FPGM[3]	2.35G	1.5MB	199.99	0.5208
BIGP (Ours)	2.35G	1.5MB	88.47	0.9291

4.3 Stability of Training Progress

In this section, we follow the story of the measure effectiveness, and demonstrate that the instability happened during the training progress of the GS32 model. We first compare the learning dynamics of both BIGP and GS32, then we choose the epoch when the GS32 collapses to analyze its channel pruning selections.

Training Curve. We draw the epoch-channel and the epoch-FID curves in Figure 3 (a). It can be seen that, in the GS32 model, the number of channels drops rapidly in the first 50 epochs during training. However, the FID score also degrades dramatically. The model seems to have forgotten all the learned representations in the original model and begins to learn the translation task a second time until the end of training. Yet the number of channels barely decreases since the 50th epoch, and finally, the model converges to a near-stable solution with an FID score of 91.46.

On the other hand, our proposed BIGP approach spends its first 120 epochs to obtain a model that is sufficiently stable for pruning. Then in the second and third stages, with increased L_{BIG} weights and looser thresholds for pruning, it reaches the solution superior to that obtained by the GS32 method in terms of both numbers of channels and FID scores. The red dot lines in the figure mean the beginning of the next stage. In Figure 3 (b), we also present the examples of generated images when 1) the GS32 collapse; 2) BIGP converges in the stage2 (the best-balanced model); 3) two method has the same numbers of channels left. Notably, our proposed BIGP can preserve most of the details existing in the original unpruned model. In Figure 4, our pruned model generates samples that have similar stripes to the one generated by CycleGAN. Meanwhile, the stripes generated by GS32 have a totally different appearance.

Error Analysis of Channels. In order to fully understand the proposed BIGP measure and the stability during model pruning progress, we select the epoch when the GS32 approach begins to collapse and examine the channels that have been marked as important or unimportant mistakenly. We use the subscription to denote the number of the epoch. In the model GS32_{40}, there remain 1157 channels; among them, we find 33 channels satisfy the condition in Eq. (27) and are ready to be pruned. We prune them in the zero-shot style and find barely any change in the generated images (see columns 2 and 4 in Figure 3 (c)).

On the other hand, in GS32_{41}, only 1071 channels survived. That means a total of 86 channels are pruned during the training of epoch 41. Among them, we select two extreme cases to show the fallibility of the optimized ℓ_1 norm measure. We present them in Table 4.

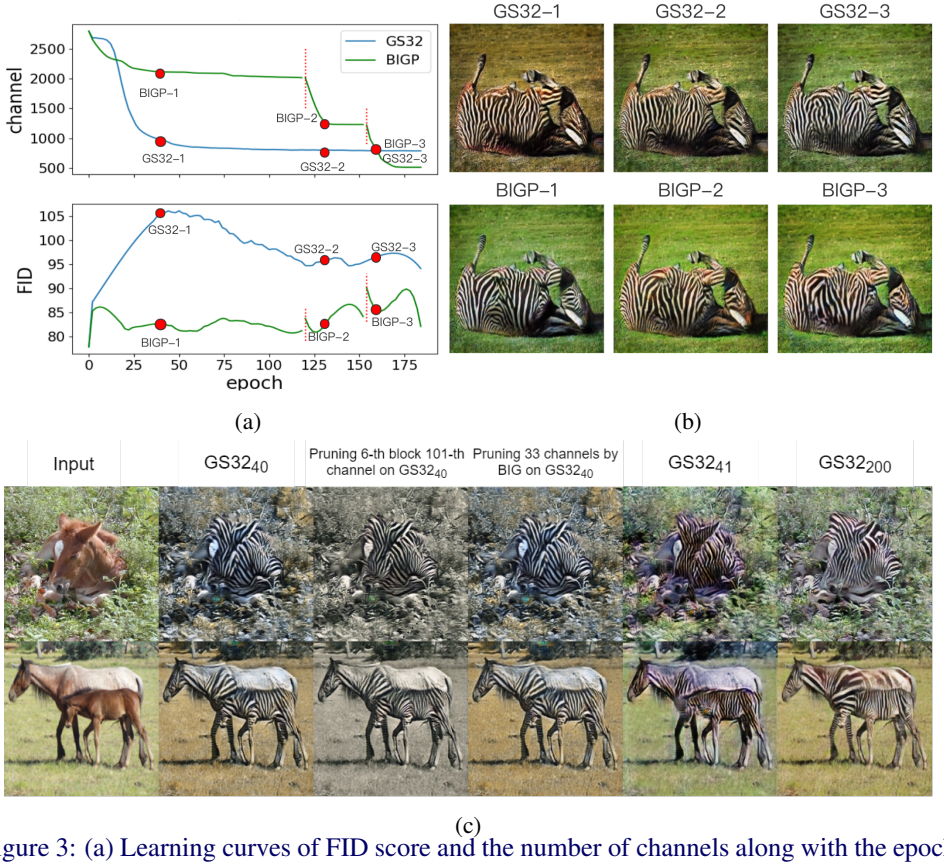

Figure 3: (a) Learning curves of FID score and the number of channels along with the epoch of a complete pruning run. (b) The generated zebra images from the models are marked as red dots in (a), and the methods and the model ids are annotated above the images. (c) Examples of the generated images corresponding to the input image (column 1), the GS32 model$_{\text{epoch}}$ collapse-recovery (columns 2,5,6) events, and pruning specified channels during the collapse (columns 3,4).

It can be seen that $\left|\gamma_{6,22}\right| > \left|\gamma_{6,101}\right|$, therefore the 6,101-channel is wrongly pruned, despite of being significant in generating visual concepts (see column 3 and 5 in the Figure 3 (c)).

4.4 Additional error analysis of channels

This section provides two more runs of the GS32 approaches named GS32-13G and GS32-bad, respectively. The GS32-13G run is the one we found with the best stability, while its final number of channels is the highest among different runs. On the other hand, the GS32-bad is a run that collapses badly during the training progress. The training progress is shown in Figure 5. The GS32 is the run we presented in the main manuscript.

Here we also conduct additional error analysis of channels for the GS32-13G run. We select epoch 26 when GS32-13G begins to collapse, shown seen in the top right of Figure 5. Same as the main manuscript, we use the subscription to denote the number of epochs. The model we analyze is denoted as GS32-13G$_{26}$. Recall the definition of the BIG condition; if

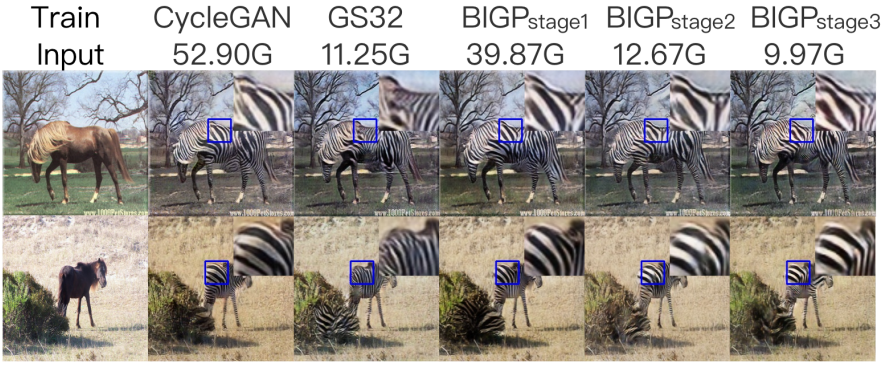

Figure 4: Examples of CycleGAN results on `horse2zebra` **train** set. The methods and their FLOPs (in G) are annotated above the images. Each row shows a different sample.

a channel satisfies the conditions defined in Eq. (27), then we prune it.

We select[2] the 23rd block of the model GS32-13G_{26} to analyze. There remain 34 channels; among them, we find the 11th channel satisfying the condition in Eq. (27) and are ready to be pruned. On the other hand, in the 23rd block of GS32-13G_{27} only 31 channels survived, where the 34th channel, the 39th channel, and the 59th channel are pruned during the training of epoch 27.

We prune one of the 34 channels at a time in the zero-shot style. The results of the corresponding outputs are shown in Figure 6. Barely any change can be seen in the generated images by pruning the 11th channel (row 2, column 4 in Figure 6). On the other hand, we can find a significant change in the generated images by pruning 34-channel or 39-channel (row 4 column 6 and row 5 column 4 in Figure 6).

We also present the optimized ℓ_1 norm measure and BIG for all the 34 channels in Table 4. It can be clearly seen that the 11th channel satisfies the BIG condition; hence the pruning is safe. Meanwhile, both the 34th channel and the 39th channel have large BIGP measures; pruning them will result in significant changes in the output.

[2] arbitrarily

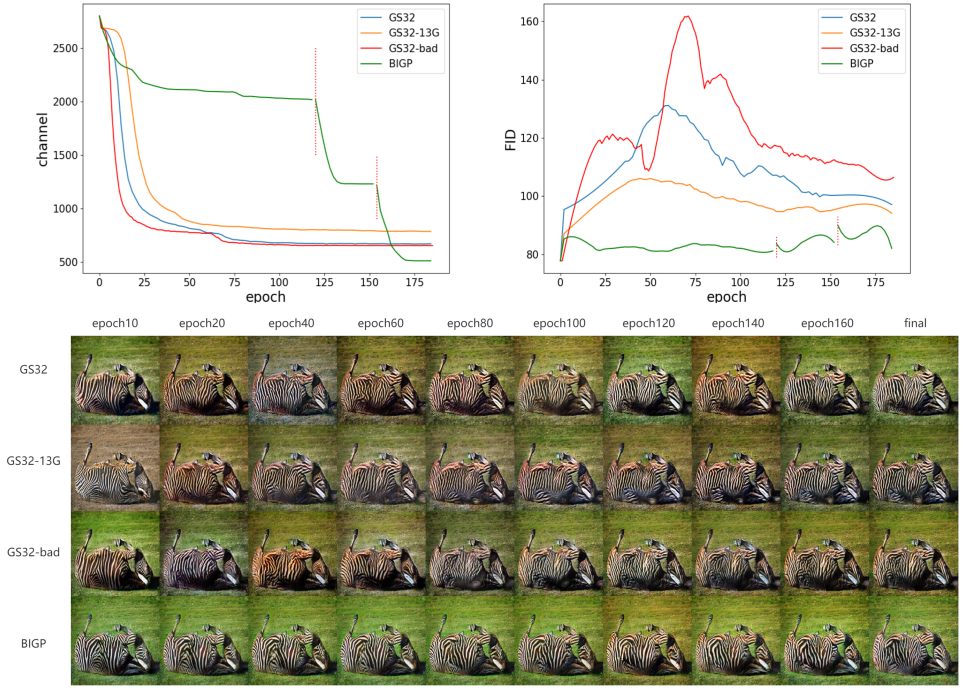

Figure 5: Learning curves of FID score and the number of channels along the epoch of a complete pruning run.

Figure 6: From top-left to bottom-right: Input, the general image by GS32-13G_{26},the general image by GS32-13G_{27}, the general image with pruning *i*-channel by GS32-13G_{26}. The *i* is marked in the upper right corner of the image.

Table 4: The comparison of channel importance measures. C stands for the C-th channel from the 23-th block.

C	$\lvert\gamma\rvert$ [9]	BIGP Measure	Satisfy Eq. (27)?	Influence
0	0.7387	1.5951	✗	Trivial
2	0.8990	1.5462	✗	Trivial
5	0.8400	1.1642	✗	Trivial
7	1.1775	1.3383	✗	Significant
9	1.0666	2.1659	✗	Trivial
10	0.6048	0.8723	✗	Trivial
11	0.5009	0.1940	✓	Trivial
12	0.6439	0.8685	✗	Trivial
14	0.7594	1.3825	✗	Trivial
15	0.8615	1.3165	✗	Trivial
17	0.6202	0.9134	✗	Trivial
21	2.4474	2.5143	✗	Significant
24	0.7317	1.0011	✗	Trivial
25	1.2615	1.7249	✗	Trivial
26	0.7066	1.4068	✗	Trivial
28	1.4819	2.1917	✗	Trivial
29	0.5792	1.0672	✗	Trivial
30	0.8984	1.0914	✗	Trivial
31	1.0790	2.0032	✗	Trivial
33	1.1128	2.9770	✗	Significant
34	1.2088	1.9504	✗	Significant
35	1.9061	2.7983	✗	Significant
36	0.4754	1.0369	✗	Trivial
37	1.0808	1.8591	✗	Trivial
39	0.4713	2.4302	✗	Significant
42	1.3134	1.6557	✗	Trivial
43	1.9453	3.4817	✗	Trivial
45	0.5872	0.9523	✗	Trivial
48	0.9820	1.9796	✗	Trivial
53	0.7464	1.3464	✗	Trivial
59	0.4580	0.8681	✗	Trivial
60	0.8489	1.6040	✗	Trivial
62	1.5269	3.0062	✗	Significant
63	0.8970	1.2491	✗	Trivial

4.5 Additional qualitative results

We provide more visualization results of on-training weight pruning in Figure 7 and Figure 8, results of zero-shot pruning state-of-the-art models in Figure 9 and Figure 10.

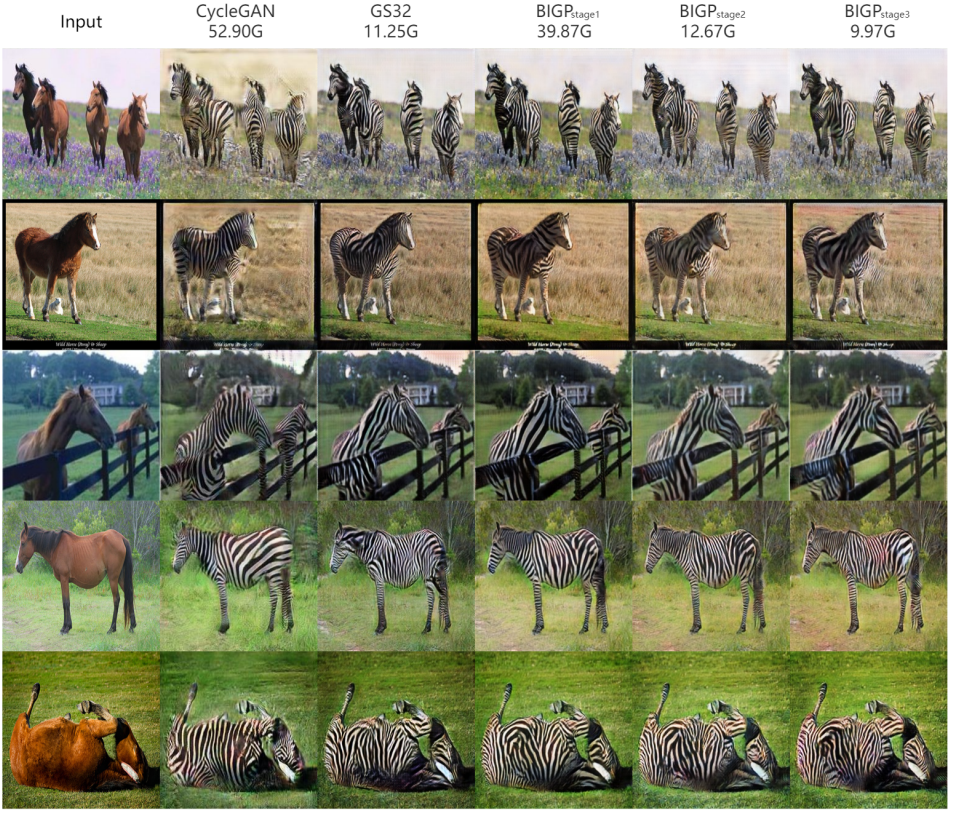

Figure 7: More examples of CycleGAN pruning results on `horse2zebra` test set. The methods and their FLOPs (in G) are annotated above the images. Each row shows a different sample.

Figure 8: More examples of CycleGAN pruning results on `summer2winter` test set. The methods and their FLOPs (in G) are annotated above the images. Each row shows a different sample.

Figure 9: More examples of zero-shot pruning OMGD [] results on `horse2zebra` test set. The methods and their FLOPs (in G) are annotated above the images. Each row shows a different sample.

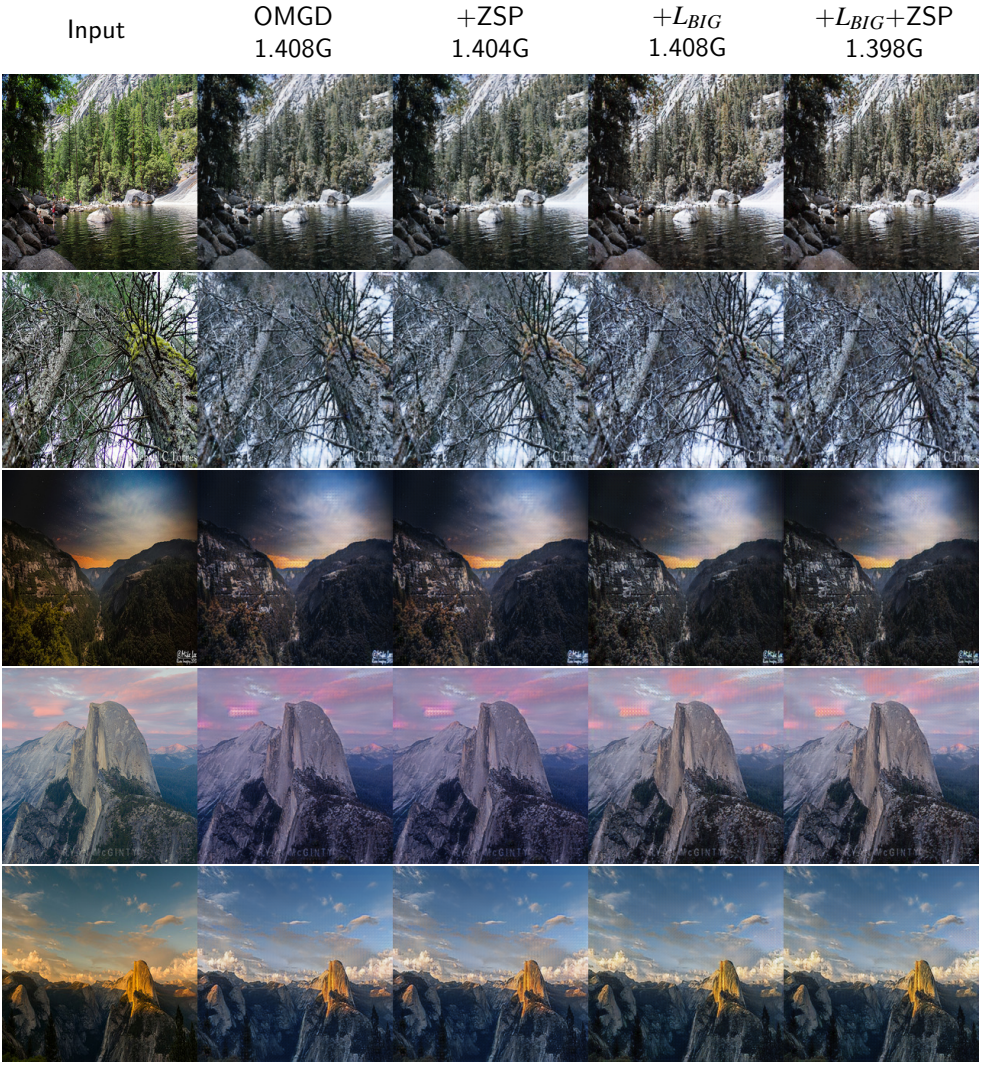

Figure 10: More examples of zero-shot pruning OMGD [▭] results on `summer2winter` test set. The methods and their FLOPs (in G) are annotated above the images. Each row shows a different sample.

References

[1] Hanting Chen, Yunhe Wang, Han Shu, Changyuan Wen, Chunjing Xu, Boxin Shi, Chao Xu, and Chang Xu. Distilling portable generative adversarial networks for image translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3585–3592, 2020.

[2] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[3] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[4] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[5] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[6] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019.

[7] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[8] Muyang Li, Ji Lin, Yaoyao Ding, Zhijian Liu, Jun-Yan Zhu, and Song Han. Gan compression: Efficient architectures for interactive conditional gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5284–5294, 2020.

[9] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[10] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[11] Yuxi Ren, Jie Wu, Xuefeng Xiao, and Jianchao Yang. Online multi-granularity distillation for gan compression. *arXiv e-prints*, 2021.

[12] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[13] Haotao Wang, Shupeng Gui, Haichuan Yang, Ji Liu, and Zhangyang Wang. Gan slimming: All-in-one gan compression by a unified optimization framework. In *European Conference on Computer Vision*, pages 54–73. Springer, 2020.

[14] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8612–8620, 2019.