FastGAN: FASTER AND STABILIZED GAN

MAURO FICORELLA 1941639 | MARTINA TURBESSI 1944497 | VALENTINA SISTI 1952657

INTRODUCTION

Our work is based on the paper "Towards Faster And Stabilized Gan Training For High-fidelity Few-shot Image Synthesis".

OBJECTIVES:

- Allow users with **limited computing budget and resources** to train a GAN;
- Eliminate the requirement of a **big dataset** for training;
- Allow to **train models from scratch** in order to avoid biases typic of the pre-trained models and to let people use **their own set** of interest images;
- **Few hours** required for training time;

Python

TensorFlow

NVIDIA CUDA + cuDNN

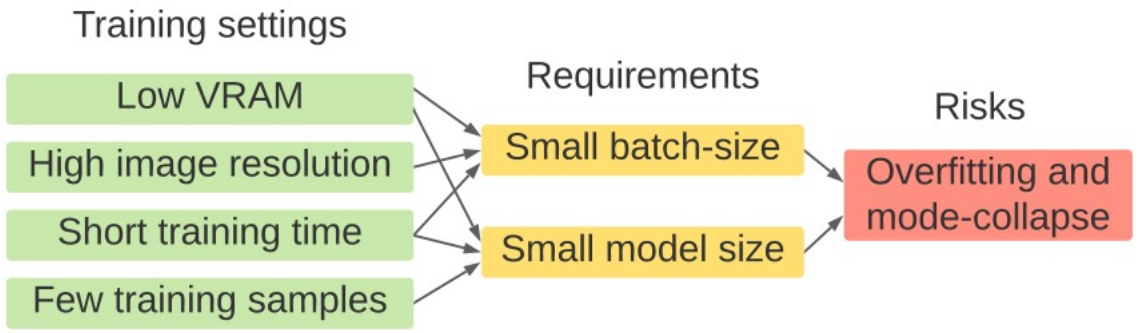
HARDWARE ADOPTED:

- NVIDIA GeForce RTX 2070 SUPER (8GB of VRAM)
- NVIDIA GeForce GTX 1050-Ti (4GB of VRAM)

IMPLEMENTATION

GAN'S PROBLEMS

- **Accelerate training:** this problem has been approached from various perspectives, but this brought only *very limited improvements* in training speed, while not enhancing quality within the shortened training time;
- **High resolution training:** this made GAN *much harder to converge*. There were some approaches trying to solve this problem, but they led to a slightly greater computational cost, consuming more GPU memory and requiring more training time;
- **Stabilize training:** *mode-collapse* on the generator is a big challenge when training GANs, given fewer training samples and lower computational power and budgets (smaller batch-size). There were approaches that tried to solve this problem, but they worked only on low resolution images with unlimited computing resources.



FastGAN SOLUTION

FastGAN reaches the objectives based on the following two techniques:

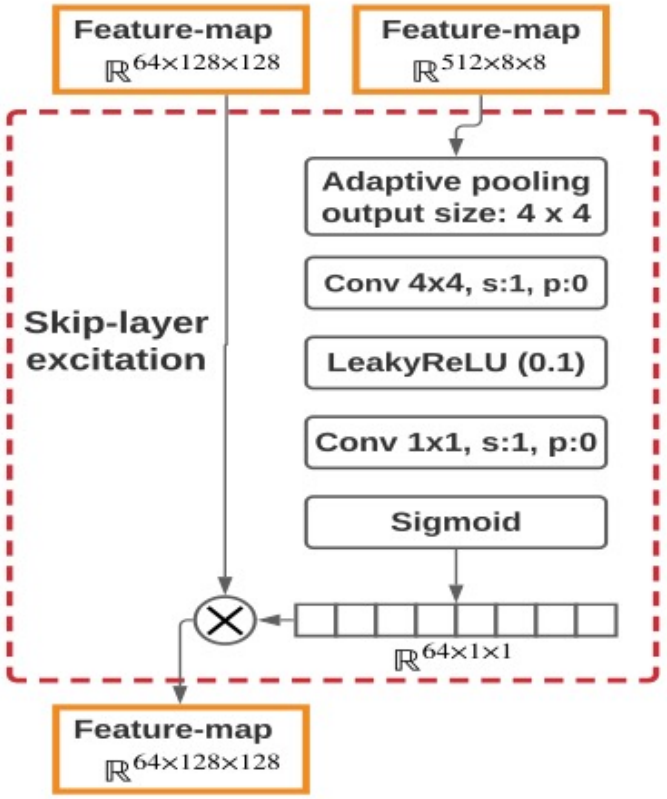
- **Skip-Layer channel-wise Excitation module:** revises channel responses on high-scale feature-maps using low-scale activations and allows *to reach a faster training* using a more robust gradient flow throughout the model weights;
- **Self-supervised discriminator D trained as a feature-encoder with an extra decoder:** this discriminator is forced to learn a feature-map that covers more regions from an image in input; in this way *it gives richer signals* in order to train G. It has been shown that auto-encoding is the best self-supervision strategy for D.

FastGAN METHOD

In order to optimize GANs with respect to the SOTA models, in FastGAN is adopted a **lightweight model**, using a single convolutional layer on each resolution in the generator.

On the high resolutions in both generator and discriminator, only three channels for the convolutional layers are applied in input and output. This structure helps making FastGAN *faster to train*, remaining, at the same time, strong on small datasets.

METHOD: SKIP-LAYER CHANNEL-WISE EXCITATION



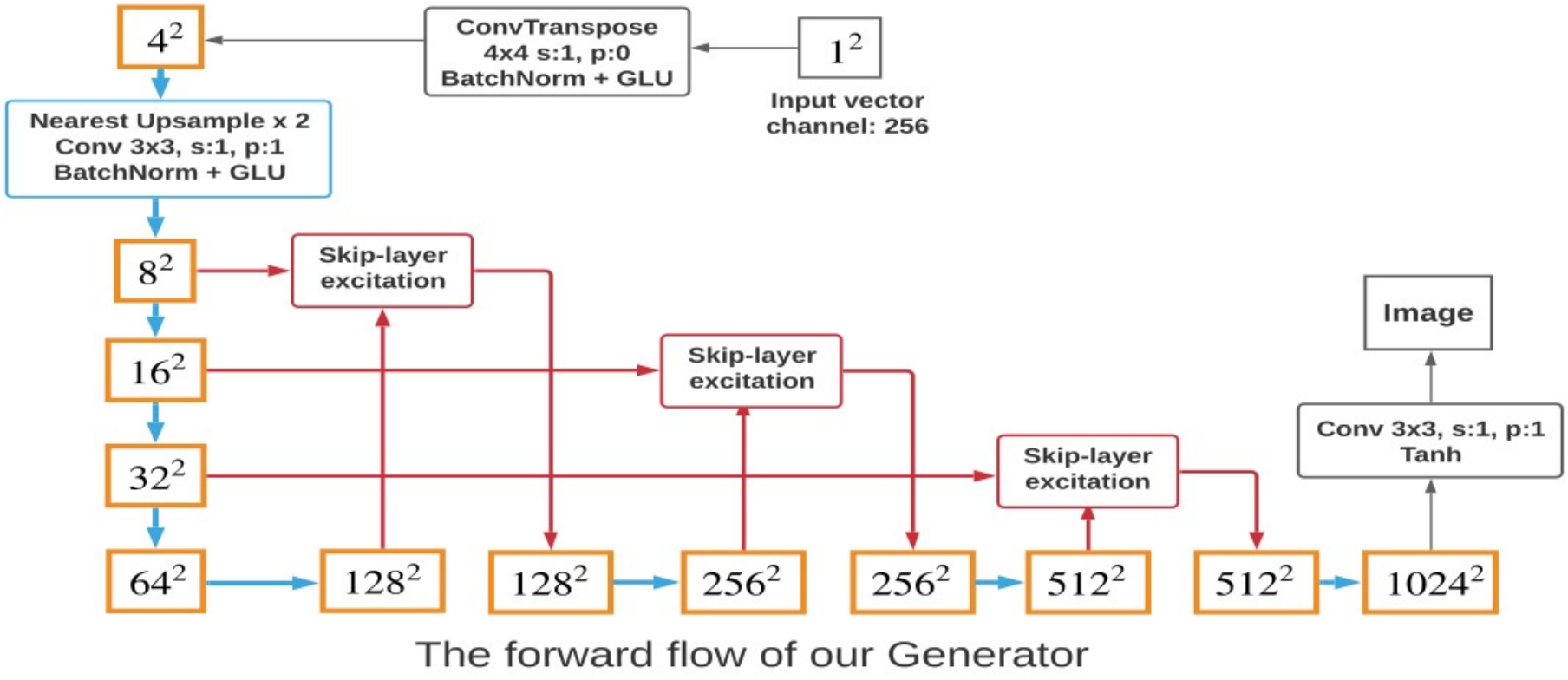
FastGAN reaches the objectives based on the following two techniques:

- **Multiplication between activations**, instead of addition, giving to one side of activations the spatial dimension of 1^2, to eliminate the complex computation of convolution;
- **Skip-connection** in a much *larger range* than previous residual-block used in GAN:
  - 8^2 – 128^2
  - 16^2 – 256^2
  - 32^2 – 512^2

The **Skip-Layer Excitation** module is defined as:

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}_{low}, \{\boldsymbol{W}_i\}) \cdot \boldsymbol{x}_{high}$$

METHOD: GENERATOR

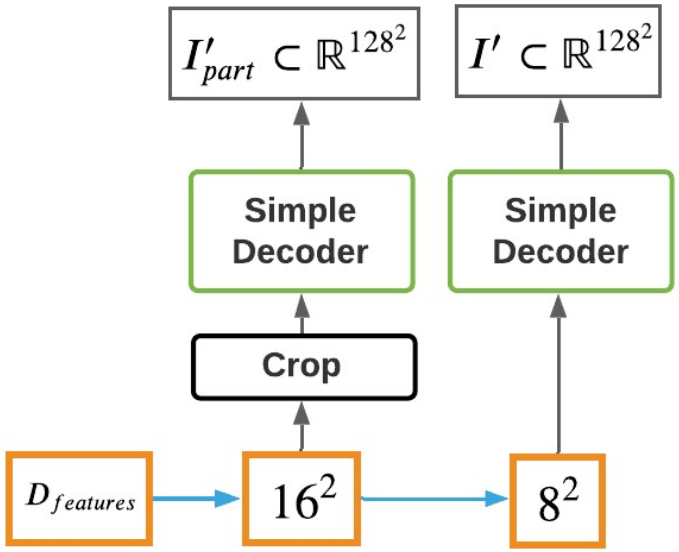


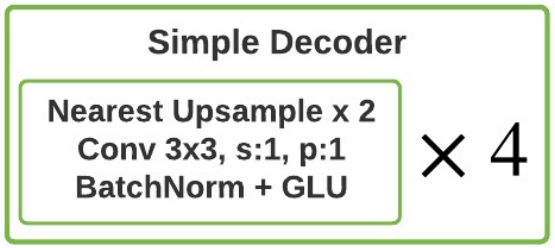
The forward flow of our Generator

Generator's adversarial **loss function**:

$$\mathcal{L}_G = -\mathbb{E}_{z\sim\mathcal{N}}[D(G(z))]$$

METHOD: SELF-SUPERVISED DISCRIMINATOR



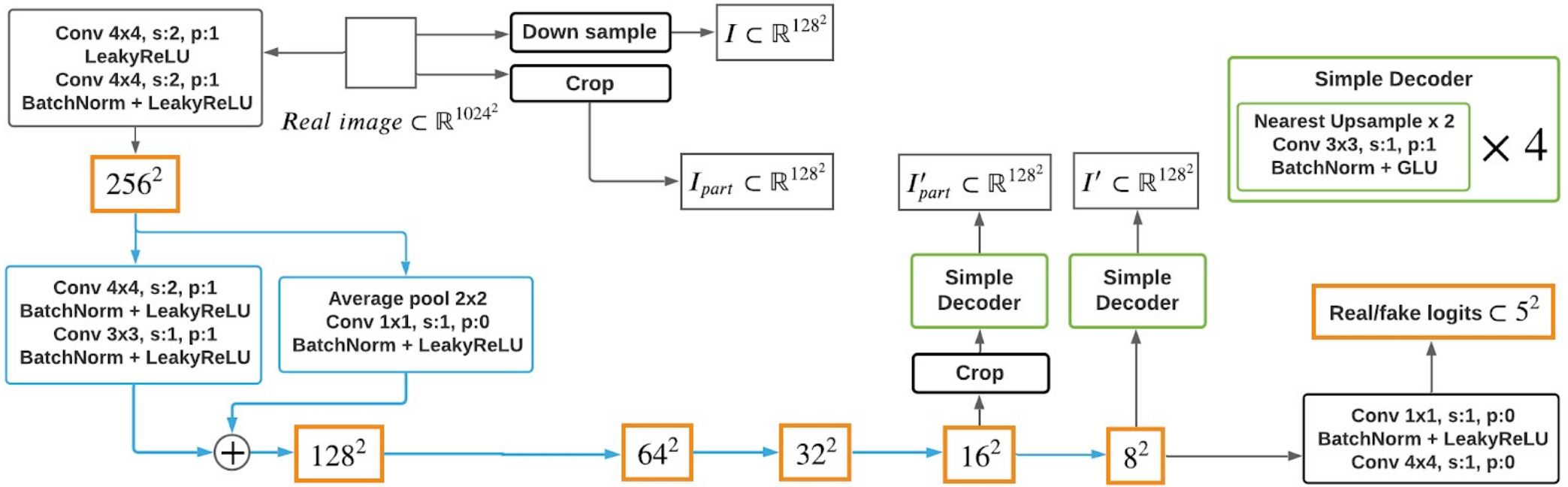


- D is treated as an **Encoder**, and paired with a simple **Decoder** to perform image reconstruction tasks;
- We use two decoders in practice, one takes 8^2 feature-map as input, one crops the 16^2 feature-map to 8^2, and both generate 128^2 images. The decoder consists only 4 convolutional layer, it actually takes very little extra computing cost, noticeably less than some regularization methods like gradient penalty.

The following **reconstruction loss** is used to optimize the decoders together with the discriminator:

$$\mathcal{L}_{recons} = \mathbb{E}_{\boldsymbol{f} \sim D_{encode}(x),\, x \sim I_{real}} [\| \mathcal{G}(\boldsymbol{f}) - \mathcal{T}(x) \|]$$

METHOD: SELF-SUPERVISED DISCRIMINATOR



The forward flow of our Discriminator

Discriminator's adversarial **loss function**:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim I_{real}}\left[min\left(0, -1 + D(x)\right)\right] - \mathbb{E}_{\hat{x}\sim G(z)}\left[min\left(0, -1 - D(\hat{x})\right)\right] + \mathcal{L}_{recons}$$

METRIC: FID

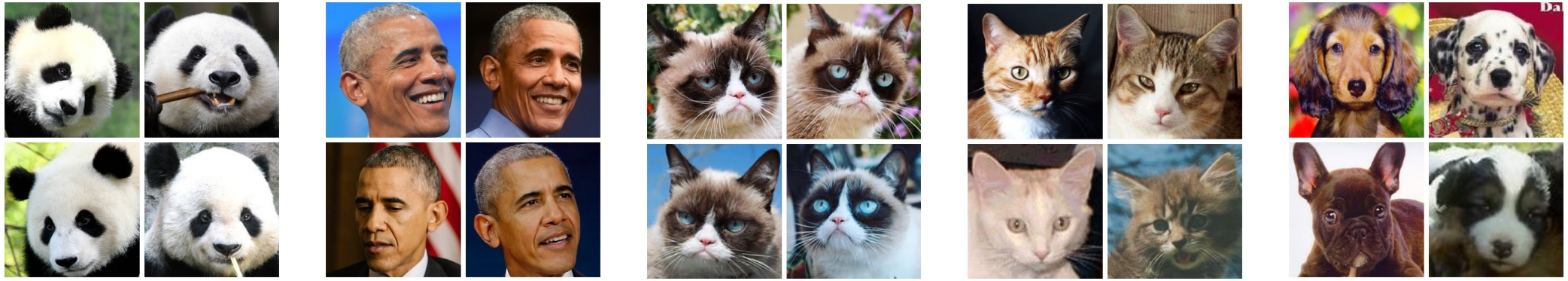
The *metric* used in our work is **FID**:

- Calculates the *distance* between feature vectors calculated for real and generated images;
- Summarizes *how similar* the two groups are in terms of statistics on computer vision features of the raw images;
- *Lower scores* indicate that the two groups of images are more similar.

DATASET

We used the following **datasets** of resolution 256 × 256:

- *Panda*
- *Obama*
- *Animal-Face Dog*
- *Animal-Face Cat*
- *Grumpy-cat*



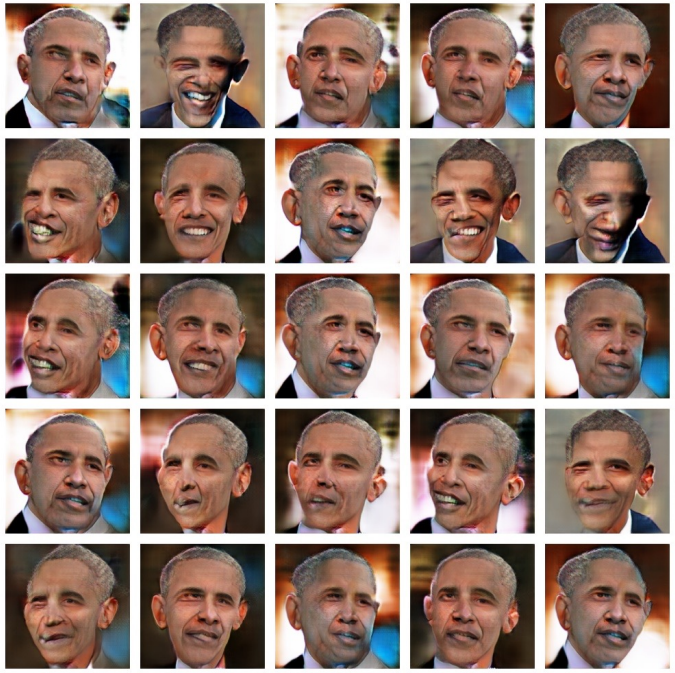
RESULTS

~13.5hrs for 1400 epochs on the *2070 SUPER*

~4.5hrs for 1000 epochs on the *2070 SUPER*

~45hrs for 1800 epochs on the *1050-Ti*

RESULTS



~7hrs for 1700 epochs on the *2070 SUPER*

~4.5hrs for 1000 epochs on the *2070 SUPER*

SUMMARIZING THE RESULTS

	NVIDIA GeForce RTX 2070 SUPER				NVIDIA GeForce GTX 1050-Ti
Dataset	**Panda**	**Dogs**	**Obama**	**Grumpy Cat**	**Cats**
Number of images	100	389	100	100	159
Resolution	256×256	256×256	256×256	256×256	256×256
Batch size	4	8	8	8	4
Epochs	1000	1400	1700	1000	1800
Epoch Time	∼15s	∼35s	∼15s	∼14s	∼103s
FID	∼28	∼35	∼75	∼129	∼108

Training parameters and **results** on *NVIDIA GeForce RTX 2070 SUPER* and *NVIDIA GeForce GTX 1050-Ti*

COMPARISON: PANDA DATASET

Our result

Original work's result

COMPARISON: DOGS DATASET

Our result

Original work's result

COMPARISON: GRUMPY CAT DATASET

Our result

Original work's result

COMPARISON WITH ORIGINAL WORK

	Panda	Dogs	Obama	Grumpy Cat	Cats
Number of images	100	389	100	100	159
FID on one RTX 2080-Ti	10.03	50.66	41.05	26.65	35.11
FID on one RTX 2070 SUPER	~28	~35	~75	~129	
FID on one GTX 1050-Ti					~108

FID score comparison between original paper (using *NVIDIA GeForce RTX 2080-Ti*) and ours (using *NVIDIA GeForce RTX 2070 SUPER* and *NVIDIA GeForce GTX 1050-Ti*)

CONCLUSIONS

Thanks to **FastGAN improvements**, given *small datasets* (sub-hundred images) and *limited computing resources*, GAN training can be stabilized with an *improved synthesis quality* using skip-layer excitation mechanism (SLE) and self-supervised regularization on the discriminator, which *boost synthesis performance* of the GAN.

Obviously, we obtained not as good results as the original work, since we had more limited computing resources; said that, these *results were still considerable*.

THANKS FOR THE ATTENTION