**UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HA NOI**

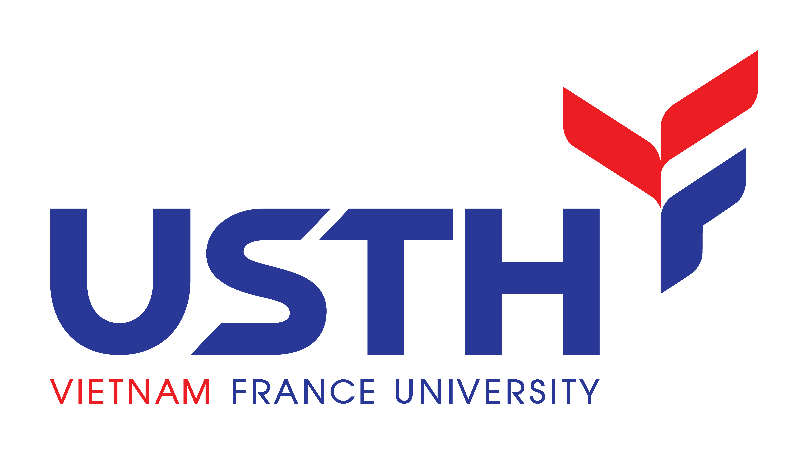
**INFORMATION AND COMMUNICATION TECHNOLOGY**

**DATA SCIENCE**

**DEEP LEARNING**

**MIDTERM REPORT**

**GROUP 41 – TOPIC 48**

****

**Lecturer: Dr. Nghiem Thi Phuong**

**Students: Trinh Van Quyet - 22BI13387**

**Do Duy Minh - 22BI13280**

**Nguyen Tien Cong - 22BI13067**

**Le Duc Dung - 22BI13103**

**Tran Tuan Kiet - 22BI13233**

**Dao Xuan Bach - 22BI13049**

**Nguyen Tran Minh Quan - 22BI13375**

Table of Contents

[I. INTRODUCTION 3](#_Toc178764622)

[1. Deep Learning 3](#_Toc178764623)

[2. Self-Attention Generative Adversarial Network (SAGAN) 3](#_Toc178764624)

[II. SPECIFIC RESEARCH 3](#_Toc178764625)

[1. Definition 3](#_Toc178764626)

[2. Theory 4](#_Toc178764627)

[3. Concrete case tudy 4](#_Toc178764628)

# INTRODUCTION

## Deep Learning

* Deep learning is a subset of machine learning inspired by the structure and function of the brain, specifically neural networks. It uses layers of artificial neurons to learn from vast amounts of data, automatically discovering representations that make sense of complex patterns. Deep learning models, especially deep neural networks, are built with multiple layers and have gained immense popularity due to their state-of-the-art performance in tasks such as image recognition, natural language processing, speech recognition, and more.
* Deep learning architectures include **Convolutional Neural Networks (CNNs)** for image data, **Recurrent Neural Networks (RNNs)** for sequential data, and **Generative Adversarial Networks (GANs)** for generating new data from a given dataset.

## Self-Attention Generative Adversarial Network (SAGAN)

* SAGAN (Self-Attention GAN) is a type of **Generative Adversarial Network (GAN)** that incorporates **self-attention mechanisms** to improve the generation of high-resolution images with better details and long-range dependencies in data.
* SAGANs are used in fields like computer vision, especially for tasks involving image synthesis, super-resolution, and style transfer where detailed, high-quality image generation is important.

# SPECIFIC RESEARCH

## Definition

* A **Self-Attention Generative Adversarial Network (SAGAN)** is a type of GAN architecture that incorporates **self-attention mechanisms**. The SAGAN model improves the quality of generated images by focusing on the most important parts of an image through self-attention, allowing the network to capture long-range dependencies between pixels in an image.
* This differs from traditional GANs, which mostly rely on local features via convolutional layers.
* **Generative Adversarial Network (GAN)**: GANs consist of two neural networks - a **generator** and a **discriminator** - competing against each other. The generator aims to create realistic fake data (e.g., images), while the discriminator attempts to distinguish between real and fake data.
* **Self-Attention**: It allows the network to weigh different parts of the input data based on their relevance to the task. This is important in scenarios like image generation, where distant parts of the image may have dependencies. For example, the positioning of the nose and eyes on a face image.

## Theory

* **Generator**: Takes random noise as input and generates an image.
* **Discriminator**: Receives real or generated images and predicts whether they are real or fake.
* **Self-Attention**: Added to both the generator and discriminator to compute the attention scores for different spatial locations. It allows the generator to learn the relationships between distant pixels, improving the quality and realism of generated images.
* The mathematical formulation of self-attention:
* Given an input sequence of vectors X = [x1, x2, …, xn], where xi is a vector representing the i-th element in the sequence, we compute the self-attention output Y as follows:

Y = softmax () V

* **Q**: Query matrix
* **K**: Key matrix
* **V**: Value matrix
* **dk**: Dimensionality of the key

## Concrete case tudy

* **Research Study using SAGAN on Dog images.**

1. **Expected goal**

* Generate high-resolution images of cats and dogs that look realistic and diverse using SAGAN.

1. **Dataset pre-processing**



* Dataset: 4615 images of dogs.
* Firstly, we resize images to 64x64 and normalize between [-1, 1].
* Then, we apply data augmentation such as cropping and random shuffle.
* Next, we check if this dataset have corrupted images or not.
* Finally, we update dataset loader by skip error images if there are any.

1. **Model setup**

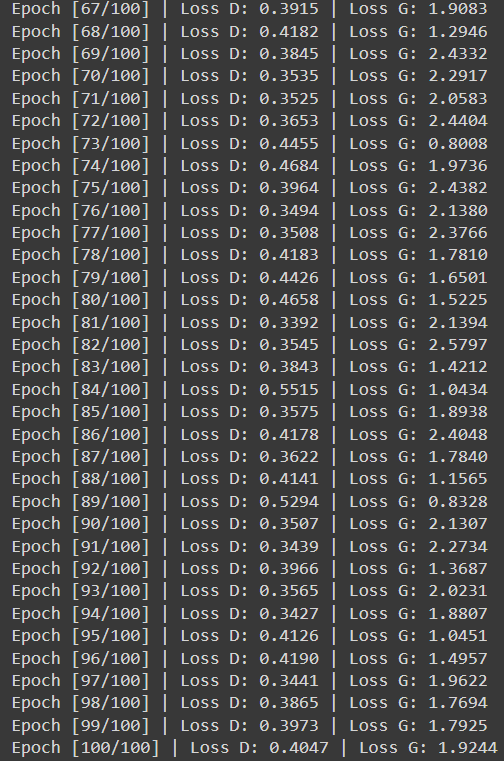
* **Self – attention layer:**
* Firstly, we initialize parameters as Q, K, V, and softmax to apply the above formula.
* Then, we scale the output from the attention mechanism.
* **Generator:**
* **ConvTranspose2d layers** are used for upsampling the noise input and generating images.
* **SelfAttention** is added after one of the layers to capture global dependencies in the image, which improves the generator's ability to model complex patterns.
* **BatchNorm2d** and **ReLU** are used for stable training and adding non-linearity, respectively.
* **Tanh** ensures that the output pixel values are in the range [-1, 1], a standard normalization for GANs.
* **Discriminator:**
* **Conv2d layers**: Used to downsample the input image and extract hierarchical features from low-level to high-level as the image passes through the network.
* **LeakyReLU activations**: Introduce non-linearity and help avoid the dying ReLU problem, where neurons get stuck and don’t learn.
* **SelfAttention**: Adds a self-attention mechanism to allow the discriminator to model long-range dependencies in the image, enhancing its ability to differentiate between real and fake images.
* **Final Conv2d layer**: Outputs a scalar value for each input image, which represents the probability that the image is real or fake.

1. **Training model**

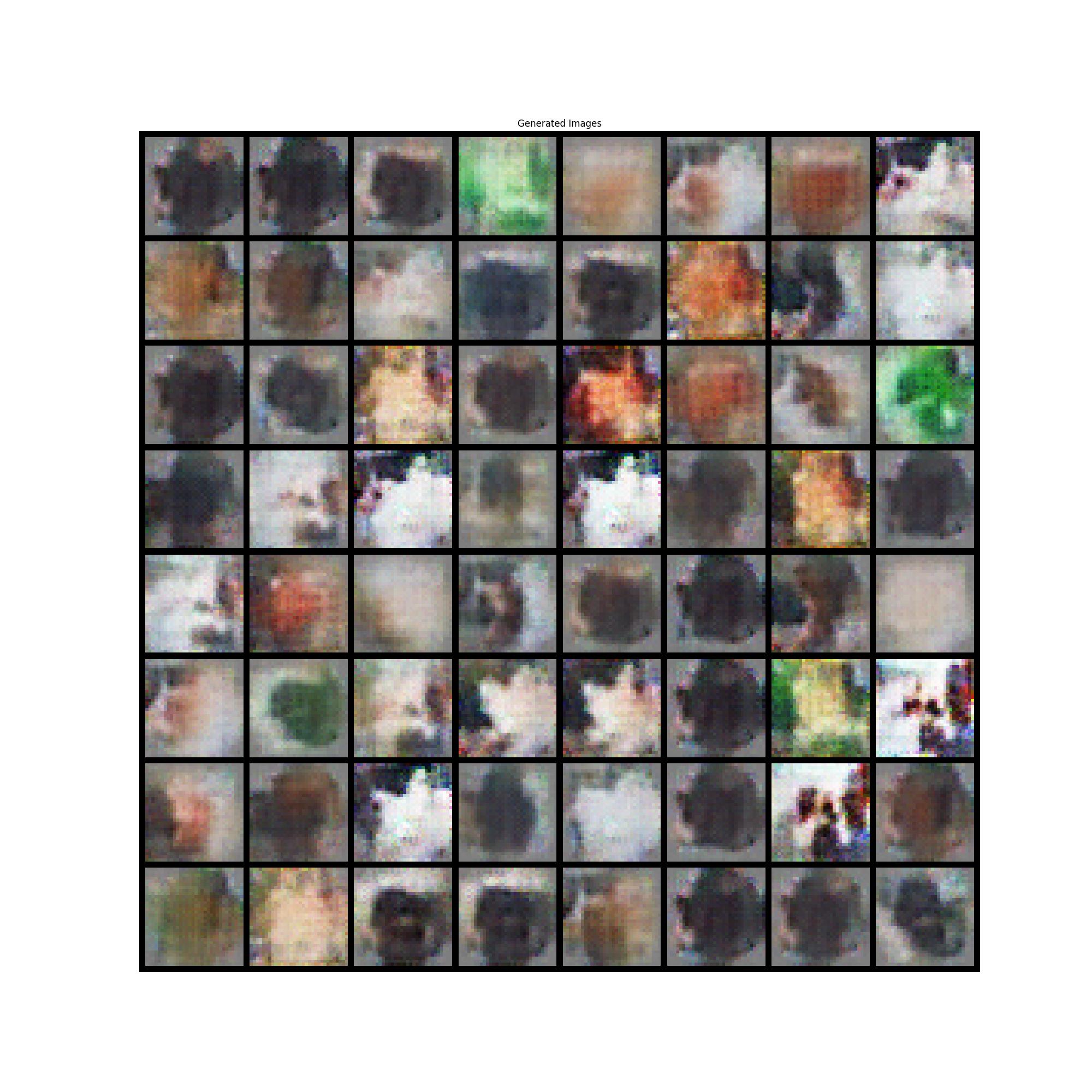
* The first, we initialize model and setup device for training.
* we want to avoid discriminator is overpowering than generator because when the discriminator is too good at its job, it leaves little room for the generator to improve. So that, we optimize generator with lr = 0.001, betas = (0.5, 0.999) and discriminator with lr = 0.0001, betas = (0.5, 0.999).
* Next, we compute loss by using the binary cross-entropy loss with logits.
* We create fixed noise to track how the generator improves over times.
* Finally, we create a training loop:
* Discriminator Training: The discriminator is trained to distinguish between real images (from the dataset) and fake images (generated by the generator).
  + The discriminator's prediction for real images is reshaped to a 1D tensor, and the loss is calculated using real labels.
  + The discriminator's prediction for fake images is computed, and the loss is calculated using fake labels. The .detach() method is used to prevent backpropagation through the generator during discriminator training.
  + The discriminator weights are updated.
* Generator Training: The generator is trained to generate images that can fool the discriminator.
  + Output: The discriminator's prediction for the fake images (this time without detaching).
  + Loss: The generator tries to fool the discriminator, so the loss is computed as the difference between the discriminator's prediction and real label.
  + The generator weights are updated.

1. **Results visualization**

* **Loss values:**

****

* **Results:**

****