Understanding VQGAN-CLIP Interaction

In the realm of generative models, **VQGAN** and **CLIP** play crucial roles, working collaboratively to produce meaningful and contextually relevant images based on textual prompts. Let's break down the acronym:

- **VQGAN** stands for **Vector Quantized Generative Adversarial Network**. This model is responsible for the generation of images. Its architecture employs techniques like vector quantization to create visually appealing and diverse outputs.
- **CLIP** stands for **Contrastive Language-Image Pretraining**. This model serves as the judge, assessing the alignment between generated images and a given text prompt. Its contrastive pretraining enables it to understand the relationships between textual and visual data.

When we refer to **VQGAN-CLIP**, we are describing the collaborative interaction between these two models. This interaction is integral to the creative process, guiding the generator to refine its outputs based on the evaluation provided by CLIP.

Workflow Overview

1. **Image Generation (VQGAN):** VQGAN takes the lead in the creative process by generating images. Its architecture incorporates techniques such as generative adversarial networks (GANs) and vector quantization to produce a diverse range of images.
2. **Textual Evaluation (CLIP):** Once an image is generated, it undergoes evaluation by CLIP. This involves assessing how well the image aligns with the given text prompt. CLIP leverages its contrastive pretraining to understand the semantic relationships between textual and visual representations.
3. **Guided Image Refinement:** The evaluation provided by CLIP serves as feedback to the generator (VQGAN). This feedback guides the generator in refining its outputs, aiming to create images that closely match the intended interpretation of the textual prompt.

This dynamic interplay between VQGAN and CLIP creates a feedback loop, iteratively improving the quality and relevance of the generated images in response to the given textual prompts.

Guiding Image Generation with CLIP and VQGAN

In the collaborative dance of creativity, **CLIP** takes on the role of the *Perceptor*, steering the creative process, while **VQGAN** acts as the *Generator*, bringing forth images from

the depths of its latent space.

Understanding the Models

- **VQGAN as the Generator:** Following the footsteps of typical GANs, VQGAN operates by taking in a noise vector and transforming it into a realistic image. It excels in the art of image creation, generating diverse outputs through its unique architecture.
- **CLIP as the Perceptor:** CLIP, on the other hand, is the guiding force. It functions as the Perceptor, capable of understanding both images and text. When presented with an image and text, CLIP extracts features from both modalities. The similarity between the image and text is then measured through the cosine similarity of the learned feature vectors.

In [1]:
```
# Install required PyTorch packages
!pip install --user torch==1.9.0 torchvision==0.10.0 torchaudio==0.9.0 torch
!git clone https://github.com/openai/CLIP

!git clone https://github.com/CompVis/taming-transformers.git

!pip install ftfy regex tqdm omegaconf pytorch-lightning kornia imageio-ffmp

# Create a directory for steps
!mkdir steps
```

```
Collecting torch==1.9.0
  Downloading torch-1.9.0-cp37-cp37m-manylinux1_x86_64.whl (831.4 MB)
     |████████████████████████████████| 831.4 MB 1.3 kB/s  eta 0:00:011
|████████████                    | 348.3 MB 72.1 MB/s eta 0:00:07     |████
████████████████████████████████| 816.5 MB 68.8 MB/s eta 0:00:01
Collecting torchvision==0.10.0
  Downloading torchvision-0.10.0-cp37-cp37m-manylinux1_x86_64.whl (22.1 MB)
     |████████████████████████████████| 22.1 MB 61.9 MB/s eta 0:00:01
Collecting torchaudio==0.9.0
  Downloading torchaudio-0.9.0-cp37-cp37m-manylinux1_x86_64.whl (1.9 MB)
     |████████████████████████████████| 1.9 MB 54.7 MB/s eta 0:00:01
Collecting torchtext==0.10.0
  Downloading torchtext-0.10.0-cp37-cp37m-manylinux1_x86_64.whl (7.6 MB)
     |████████████████████████████████| 7.6 MB 44.9 MB/s eta 0:00:01
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.
7/site-packages (from torch==1.9.0) (3.10.0.2)
Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packa
ges (from torchvision==0.10.0) (1.19.5)
Requirement already satisfied: pillow>=5.3.0 in /opt/conda/lib/python3.7/si
te-packages (from torchvision==0.10.0) (8.2.0)
Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-pa
ckages (from torchtext==0.10.0) (2.25.1)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packag
es (from torchtext==0.10.0) (4.62.3)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/lib/python3.
7/site-packages (from requests->torchtext==0.10.0) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/sit
e-packages (from requests->torchtext==0.10.0) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/pyth
on3.7/site-packages (from requests->torchtext==0.10.0) (1.26.6)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python
3.7/site-packages (from requests->torchtext==0.10.0) (2021.10.8)
Installing collected packages: torch, torchvision, torchtext, torchaudio
  WARNING: The scripts convert-caffe2-to-onnx and convert-onnx-to-caffe2 ar
e installed in '/root/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this
warning, use --no-warn-script-location.
Successfully installed torch-1.9.0 torchaudio-0.9.0 torchtext-0.10.0 torchv
ision-0.10.0
WARNING: Running pip as the 'root' user can result in broken permissions an
d conflicting behaviour with the system package manager. It is recommended
to use a virtual environment instead: https://pip.pypa.io/warnings/venv
Cloning into 'CLIP'...
remote: Enumerating objects: 251, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 251 (delta 3), reused 2 (delta 0), pack-reused 243
Receiving objects: 100% (251/251), 8.93 MiB | 37.61 MiB/s, done.
Resolving deltas: 100% (127/127), done.
Cloning into 'taming-transformers'...
remote: Enumerating objects: 1342, done.
remote: Counting objects: 100% (2/2), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 1342 (delta 0), reused 1 (delta 0), pack-reused 1340
Receiving objects: 100% (1342/1342), 409.77 MiB | 18.74 MiB/s, done.
Resolving deltas: 100% (281/281), done.
Collecting ftfy
  Downloading ftfy-6.1.1-py3-none-any.whl (53 kB)
     |████████████████████████████████| 53 kB 1.1 MB/s eta 0:00:011
Requirement already satisfied: regex in /opt/conda/lib/python3.7/site-packa
ges (2021.8.28)
Requirement already satisfied: tqdm in /opt/conda/lib/python3.7/site-packag
es (4.62.3)
Collecting omegaconf
```

```
  Downloading omegaconf-2.3.0-py3-none-any.whl (79 kB)
     |████████████████████████████████| 79 kB 4.9 MB/s  eta 0:00:01
Requirement already satisfied: pytorch-lightning in /opt/conda/lib/python3.
7/site-packages (1.4.4)
Requirement already satisfied: kornia in /opt/conda/lib/python3.7/site-pack
ages (0.5.8)
Collecting imageio-ffmpeg
  Downloading imageio_ffmpeg-0.4.9-py3-none-manylinux2010_x86_64.whl (26.9
MB)
     |████████████████████████████████| 26.9 MB 66.0 MB/s eta 0:00:01
Collecting einops
  Downloading einops-0.6.1-py3-none-any.whl (42 kB)
     |████████████████████████████████| 42 kB 850 kB/s  eta 0:00:01
Requirement already satisfied: wcwidth>=0.2.5 in /opt/conda/lib/python3.7/s
ite-packages (from ftfy) (0.2.5)
Collecting antlr4-python3-runtime==4.9.*
  Downloading antlr4-python3-runtime-4.9.3.tar.gz (117 kB)
     |████████████████████████████████| 117 kB 67.8 MB/s eta 0:00:01
Requirement already satisfied: PyYAML>=5.1.0 in /opt/conda/lib/python3.7/si
te-packages (from omegaconf) (5.4.1)
Requirement already satisfied: tensorboard>=2.2.0 in /opt/conda/lib/python
3.7/site-packages (from pytorch-lightning) (2.6.0)
Requirement already satisfied: torch>=1.6 in /root/.local/lib/python3.7/sit
e-packages (from pytorch-lightning) (1.9.0)
Requirement already satisfied: torchmetrics>=0.4.0 in /opt/conda/lib/python
3.7/site-packages (from pytorch-lightning) (0.5.0)
Requirement already satisfied: numpy>=1.17.2 in /opt/conda/lib/python3.7/si
te-packages (from pytorch-lightning) (1.19.5)
Requirement already satisfied: typing-extensions in /opt/conda/lib/python3.
7/site-packages (from pytorch-lightning) (3.10.0.2)
Requirement already satisfied: packaging>=17.0 in /opt/conda/lib/python3.7/
site-packages (from pytorch-lightning) (21.0)
Requirement already satisfied: pyDeprecate==0.3.1 in /opt/conda/lib/python
3.7/site-packages (from pytorch-lightning) (0.3.1)
Requirement already satisfied: fsspec[http]!=2021.06.0,>=2021.05.0 in /opt/
conda/lib/python3.7/site-packages (from pytorch-lightning) (2021.10.1)
Requirement already satisfied: future>=0.17.1 in /opt/conda/lib/python3.7/s
ite-packages (from pytorch-lightning) (0.18.2)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.7/site-
packages (from imageio-ffmpeg) (58.0.4)
Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-pa
ckages (from fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (2.25.
1)
Requirement already satisfied: aiohttp in /opt/conda/lib/python3.7/site-pac
kages (from fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning) (3.7.4.
post0)
Requirement already satisfied: pyparsing>=2.0.2 in /opt/conda/lib/python3.
7/site-packages (from packaging>=17.0->pytorch-lightning) (2.4.7)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/lib/python3.7/
site-packages (from tensorboard>=2.2.0->pytorch-lightning) (3.3.4)
Requirement already satisfied: protobuf>=3.6.0 in /opt/conda/lib/python3.7/
site-packages (from tensorboard>=2.2.0->pytorch-lightning) (3.19.0)
Requirement already satisfied: werkzeug>=0.11.15 in /opt/conda/lib/python3.
7/site-packages (from tensorboard>=2.2.0->pytorch-lightning) (2.0.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /op
t/conda/lib/python3.7/site-packages (from tensorboard>=2.2.0->pytorch-light
ning) (0.6.1)
Requirement already satisfied: wheel>=0.26 in /opt/conda/lib/python3.7/site
-packages (from tensorboard>=2.2.0->pytorch-lightning) (0.37.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /opt/con
da/lib/python3.7/site-packages (from tensorboard>=2.2.0->pytorch-lightning)
(0.4.6)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /opt/conda/
lib/python3.7/site-packages (from tensorboard>=2.2.0->pytorch-lightning)
```

```
(1.8.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in /opt/conda/lib/pyth
on3.7/site-packages (from tensorboard>=2.2.0->pytorch-lightning) (1.35.0)
Requirement already satisfied: grpcio>=1.24.3 in /opt/conda/lib/python3.7/s
ite-packages (from tensorboard>=2.2.0->pytorch-lightning) (1.38.1)
Requirement already satisfied: absl-py>=0.4 in /opt/conda/lib/python3.7/sit
e-packages (from tensorboard>=2.2.0->pytorch-lightning) (0.14.0)
Requirement already satisfied: six in /opt/conda/lib/python3.7/site-package
s (from absl-py>=0.4->tensorboard>=2.2.0->pytorch-lightning) (1.16.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/pyth
on3.7/site-packages (from google-auth<2,>=1.6.3->tensorboard>=2.2.0->pytorc
h-lightning) (0.2.7)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.7/si
te-packages (from google-auth<2,>=1.6.3->tensorboard>=2.2.0->pytorch-lightn
ing) (4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /opt/conda/lib/pyt
hon3.7/site-packages (from google-auth<2,>=1.6.3->tensorboard>=2.2.0->pytor
ch-lightning) (4.2.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /opt/conda/lib/p
ython3.7/site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard>
=2.2.0->pytorch-lightning) (1.3.0)
Requirement already satisfied: importlib-metadata in /opt/conda/lib/python
3.7/site-packages (from markdown>=2.6.8->tensorboard>=2.2.0->pytorch-lightn
ing) (4.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /opt/conda/lib/pytho
n3.7/site-packages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tens
orboard>=2.2.0->pytorch-lightning) (0.4.8)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/sit
e-packages (from requests->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lig
htning) (2.10)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/pyth
on3.7/site-packages (from requests->fsspec[http]!=2021.06.0,>=2021.05.0->py
torch-lightning) (1.26.6)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python
3.7/site-packages (from requests->fsspec[http]!=2021.06.0,>=2021.05.0->pyto
rch-lightning) (2021.10.8)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/lib/python3.
7/site-packages (from requests->fsspec[http]!=2021.06.0,>=2021.05.0->pytorc
h-lightning) (4.0.0)
Requirement already satisfied: oauthlib>=3.0.0 in /opt/conda/lib/python3.7/
site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.
4.1->tensorboard>=2.2.0->pytorch-lightning) (3.1.1)
Requirement already satisfied: attrs>=17.3.0 in /opt/conda/lib/python3.7/si
te-packages (from aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lig
htning) (21.2.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /opt/conda/lib/python
3.7/site-packages (from aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->pytor
ch-lightning) (5.1.0)
Requirement already satisfied: async-timeout<4.0,>=3.0 in /opt/conda/lib/py
thon3.7/site-packages (from aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->p
ytorch-lightning) (3.0.1)
Requirement already satisfied: yarl<2.0,>=1.0 in /opt/conda/lib/python3.7/s
ite-packages (from aiohttp->fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-li
ghtning) (1.6.3)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-p
ackages (from importlib-metadata->markdown>=2.6.8->tensorboard>=2.2.0->pyto
rch-lightning) (3.5.0)
Building wheels for collected packages: antlr4-python3-runtime
  Building wheel for antlr4-python3-runtime (setup.py) ... done
  Created wheel for antlr4-python3-runtime: filename=antlr4_python3_runtime
-4.9.3-py3-none-any.whl size=144575 sha256=0bbd7c78360d0238c880284ab273548c
7f271c1ed61f9116d05b5e24562b0c2c
  Stored in directory: /root/.cache/pip/wheels/8b/8d/53/2af8772d9aec614e3fc
65e53d4a993ad73c61daa8bbd85a873
```

```
Successfully built antlr4-python3-runtime
Installing collected packages: antlr4-python3-runtime, omegaconf, imageio-f
fmpeg, ftfy, einops
Successfully installed antlr4-python3-runtime-4.9.3 einops-0.6.1 ftfy-6.1.1
imageio-ffmpeg-0.4.9 omegaconf-2.3.0
WARNING: Running pip as the 'root' user can result in broken permissions an
d conflicting behaviour with the system package manager. It is recommended
to use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

In [2]:
```
import os
import torch
import argparse
import math
from pathlib import Path
import sys
sys.path.insert(1, './taming-transformers')
from omegaconf import OmegaConf
from PIL import Image
import matplotlib.pyplot as plt
from taming.models import cond_transformer, vqgan
import taming.modules
from torch import nn, optim
from torch.nn import functional as F
from torchvision import transforms
from torchvision.transforms import functional as TF
from tqdm.notebook import tqdm
from CLIP import clip
import kornia.augmentation as K
import numpy as np
import imageio
from PIL import ImageFile, Image
from urllib.request import urlopen
from tqdm import tqdm
from torch import optim
from PIL import Image
import clip
from nvidia_smi import nvmlDeviceGetUtilizationRates
# Allow loading truncated images
ImageFile.LOAD_TRUNCATED_IMAGES = True

# Initialize NVIDIA GPU monitoring
from pynvml.smi import nvmlInit, nvmlDeviceGetHandleByIndex, nvmlDeviceGetUt
nvmlInit()
handle = nvmlDeviceGetHandleByIndex(0)

# Suppress warnings
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
#Download VQGAN configuration file
torch.hub.download_url_to_file('https://heibox.uni-heidelberg.de/d/a7530b09
                               'vqgan_imagenet_f16_16384.yaml')

# Download pre-trained VQGAN model checkpoint
torch.hub.download_url_to_file('https://heibox.uni-heidelberg.de/d/a7530b09
                               'vqgan_imagenet_f16_16384.ckpt')
```

```
  0%|          | 0.00/692 [00:00<?, ?B/s]
  0%|          | 0.00/935M [00:00<?, ?B/s]
```

In [4]:
```
import torch
import torch.nn.functional as F
import math
```

```
def sinc(x):
    # Sinc function
    return torch.where(x != 0, torch.sin(math.pi * x) / (math.pi * x), x.new

def lanczos(x, a):
    # Lanczos interpolation kernel
    cond = (-a < x) & (x < a)
    out = torch.where(cond, sinc(x) * sinc(x/a), x.new_zeros([]))
    return out / out.sum()

def ramp(ratio, width):
    # Generate a ramp function
    n = math.ceil(width / ratio) + 1
    out = torch.arange(0, n) * ratio
    return torch.cat([-out[1:].flip([0]), out])[1:-1]

def resample(input, size, align_corners=True):
    # Resampling function using Lanczos interpolation
    n, c, h, w = input.shape
    dh, dw = size
    input = input.view([n * c, 1, h, w])

    if dh < h:
        # Resample height
        kernel_h = lanczos(ramp(dh / h, 2), 2).to(input.device, input.dtype)
        pad_h = (kernel_h.shape[0] - 1) // 2
        input = F.pad(input, (0, 0, pad_h, pad_h), 'reflect')
        input = F.conv2d(input, kernel_h[None, None, :, None])

    if dw < w:
        # Resample width
        kernel_w = lanczos(ramp(dw / w, 2), 2).to(input.device, input.dtype)
        pad_w = (kernel_w.shape[0] - 1) // 2
        input = F.pad(input, (pad_w, pad_w, 0, 0), 'reflect')
        input = F.conv2d(input, kernel_w[None, None, None, :])

    input = input.view([n, c, h, w])
    return F.interpolate(input, size, mode='bicubic', align_corners=align_co
```

In [5]:

```
# Define a custom autograd function for replacing gradients during backpropa
class ReplaceGrad(torch.autograd.Function):
    @staticmethod
    def forward(ctx, x_forward, x_backward):
        # Store the shape of x_backward for later use in backward pass
        ctx.shape = x_backward.shape
        return x_forward

    @staticmethod
    def backward(ctx, grad_in):
        # Replace the gradient with the sum of gradients along the stored sh
        return None, grad_in.sum_to_size(ctx.shape)

# Create an instance of the ReplaceGrad function for easy use
replace_grad = ReplaceGrad.apply

# Define a custom autograd function for clamping with gradients during backp
class ClampWithGrad(torch.autograd.Function):
    @staticmethod
    def forward(ctx, input, min, max):
        # Store min and max values for later use in backward pass
        ctx.min = min
        ctx.max = max
        # Save the input tensor for use in backward pass
        ctx.save_for_backward(input)
```

```
        # Perform the forward pass by clamping the input between min and max
        return input.clamp(min, max)

    @staticmethod
    def backward(ctx, grad_in):
        # Retrieve the saved input tensor
        input, = ctx.saved_tensors
        # Compute the gradient with respect to the input
        gradient = grad_in * (grad_in * (input - input.clamp(ctx.min, ctx.ma
        # Return the gradient and None for min and max since they are not us
        return gradient, None, None

# Create an instance of the ClampWithGrad function for easy use
clamp_with_grad = ClampWithGrad.apply
```

In [6]:

```
# Define a function for vector quantization
def vector_quantize(x, codebook):
    # Compute distances between input x and codebook vectors
    d = x.pow(2).sum(dim=-1, keepdim=True) + codebook.pow(2).sum(dim=1) - 2
    # Find indices of the nearest codebook vectors
    indices = d.argmin(-1)
    # Quantize the input based on the nearest codebook vectors
    x_q = F.one_hot(indices, codebook.shape[0]).to(d.dtype) @ codebook
    # Replace gradients during backpropagation
    return replace_grad(x_q, x)

# Define a class for a prompt module
class Prompt(nn.Module):
    def __init__(self, embed, weight=1., stop=float('-inf')):
        super().__init__()
        # Register buffer for the embedding
        self.register_buffer('embed', embed)
        # Register buffer for the weight
        self.register_buffer('weight', torch.as_tensor(weight))
        # Register buffer for the stop value
        self.register_buffer('stop', torch.as_tensor(stop))

    def forward(self, input):
        # Normalize input and embedded vectors
        input_normed = F.normalize(input.unsqueeze(1), dim=2)
        embed_normed = F.normalize(self.embed.unsqueeze(0), dim=2)
        # Compute distances between normalized input and embedded vectors
        dists = input_normed.sub(embed_normed).norm(dim=2).div(2).arcsin().p
        # Apply weight and replace gradients during backpropagation
        dists = dists * self.weight.sign()
        return self.weight.abs() * replace_grad(dists, torch.maximum(dists,

# Define a function to parse a prompt string
def parse_prompt(prompt):
    # Split the prompt string and ensure it has at least three elements
    vals = prompt.rsplit(':', 2)
    vals = vals + ['', '1', '-inf'][len(vals):]
    # Return prompt values as a tuple
    return vals[0], float(vals[1]), float(vals[2])
```

In [7]:

```
# Define a class for creating cutouts with augmentations
class MakeCutouts(nn.Module):
    def __init__(self, cut_size, cutn, cut_pow=1):
        super().__init__()
        self.cut_size = cut_size
        self.cutn = cutn
        self.cut_pow = cut_pow
```

```
        # Define a sequence of augmentations using kornia
        self.augs = nn.Sequential(
            K.RandomAffine(degrees=15, translate=0.1, p=0.7, padding_mode='b
            K.RandomPerspective(0.7, p=0.7),
            K.ColorJitter(hue=0.1, saturation=0.1, p=0.7),
            K.RandomErasing((.1, .4), (.3, 1/.3), same_on_batch=True, p=0.7)
        )

        self.noise_fac = 0.1
        self.av_pool = nn.AdaptiveAvgPool2d((self.cut_size, self.cut_size))
        self.max_pool = nn.AdaptiveMaxPool2d((self.cut_size, self.cut_size))

    def forward(self, input):
        slideY, slideX = input.shape[2:4]
        max_size = min(slideX, slideY)
        min_size = min(slideX, slideY, self.cut_size)
        cutouts = []

        # Generate cutouts using adaptive average and max pooling
        for _ in range(self.cutn):
            cutout = (self.av_pool(input) + self.max_pool(input)) / 2
            cutouts.append(cutout)

        # Apply augmentations to the concatenated cutouts
        batch = self.augs(torch.cat(cutouts, dim=0))

        # Add random noise if specified
        if self.noise_fac:
            facs = batch.new_empty([self.cutn, 1, 1, 1]).uniform_(0, self.no
            batch = batch + facs * torch.randn_like(batch)

        return batch
```

In [8]:
```
# Define a function for loading a VQGAN model
def load_vqgan_model(config_path, checkpoint_path):
    # Load the configuration from the specified path
    config = OmegaConf.load(config_path)

    # Check the target model type and initialize the appropriate model
    if config.model.target == 'taming.models.vqgan.VQModel':
        model = vqgan.VQModel(**config.model.params)
    elif config.model.target == 'taming.models.vqgan.GumbelVQ':
        model = vqgan.GumbelVQ(**config.model.params)
    elif config.model.target == 'taming.models.cond_transformer.Net2NetTrans
        parent_model = cond_transformer.Net2NetTransformer(**config.model.pa
        model = parent_model.first_stage_model
    else:
        raise ValueError(f'unknown model type: {config.model.target}')

    # Set the model to evaluation mode and disable gradient computation
    model.eval().requires_grad_(False)

    # Initialize the model from the specified checkpoint path
    model.init_from_ckpt(checkpoint_path)

    # Remove the loss attribute from the model
    del model.loss

    return model
```

In [9]:
```
# Define a function for resizing an image
def resize_image(image, out_size):
    # Calculate the aspect ratio of the original image
```

```
    ratio = image.size[0] / image.size[1]

    # Calculate the area based on the output size
    area = min(image.size[0] * image.size[1], out_size[0] * out_size[1])

    # Calculate the new size preserving the aspect ratio
    size = round((area * ratio)**0.5), round((area / ratio)**0.5)

    # Resize the image using Lanczos interpolation
    return image.resize(size, Image.LANCZOS)
```

Downloading Models

In [10]:
```
# Define parameters for the model and training
model_name = "vqgan_imagenet_f16_16384"
images_interval = 50
width, height = 512, 512
init_image = ""
seed = 42

# Create an argparse Namespace for managing parameters
args = argparse.Namespace(
    noise_prompt_seeds=[],
    noise_prompt_weights=[],
    size=[width, height],
    init_image=init_image,
    init_weight=0.,
    clip_model='ViT-B/32',
    vqgan_config=f'{model_name}.yaml',
    vqgan_checkpoint=f'{model_name}.ckpt',
    step_size=0.13,
    cutn=32,
    cut_pow=1.,
    display_freq=images_interval,
    seed=seed,
)

# Choose the appropriate device (GPU if available, otherwise CPU)
device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')
print('Using device:', device)

# Load the VQGAN model and CLIP perceptor
model = load_vqgan_model(args.vqgan_config, args.vqgan_checkpoint).to(device
perceptor = clip.load(args.clip_model, jit=False)[0].eval().requires_grad_(F
```

```
Using device: cuda:0
Working with z of shape (1, 256, 16, 16) = 65536 dimensions.
Downloading: "https://download.pytorch.org/models/vgg16-397923af.pth" to /r
oot/.cache/torch/hub/checkpoints/vgg16-397923af.pth
  0%|          | 0.00/528M [00:00<?, ?B/s]
Downloading vgg_lpips model from https://heibox.uni-heidelberg.de/f/6075038
59c864bc1b30b/?dl=1 to taming/modules/autoencoder/lpips/vgg.pth
8.19kB [00:00, 61.8kB/s]
loaded pretrained LPIPS loss from taming/modules/autoencoder/lpips/vgg.pth
VQLPIPSWithDiscriminator running with hinge loss.
Restored from vqgan_imagenet_f16_16384.ckpt
100%|████████████████████████████████████| 338M/338M [00:05<00:00, 60.6M
iB/s]
```

In [13]:
```
def inference(text,
              seed,
```

```
            step_size,
            max_iterations,
            width,
            height,
            init_image,
            init_weight,
            target_images,
            cutn,
            cut_pow,
            video_file
           ):
    # Initialize list to store generated frames
    all_frames = []

    # Extract parameters
    size = [width, height]
    texts = text.split("|") if text else []
    init_weight = init_weight

    # Handle optional parameters
    init_image = init_image if init_image else ""
    target_images = target_images.split("|") if target_images else []

    # Initialize or load the specified model
    model_names = {
        "vqgan_imagenet_f16_16384": 'ImageNet 16384',
        "vqgan_imagenet_f16_1024": "ImageNet 1024",
        'vqgan_openimages_f16_8192': 'OpenImages 8912',
        "wikiart_1024": "WikiArt 1024",
        "wikiart_16384": "WikiArt 16384",
        "coco": "COCO-Stuff",
        "faceshq": "FacesHQ",
        "sflckr": "S-FLCKR"
    }
    model_name = "vqgan_imagenet_f16_16384"  # Update with the actual model
    name_model = model_names[model_name]

    # Initialize target_images list
    target_images = target_images if target_images and target_images[0] != '

    # Seed initialization
    seed = torch.seed() if seed is None or seed == -1 else seed
    torch.manual_seed(seed)

    # Model-specific configurations
    cut_size = perceptor.visual.input_resolution
    f = 2 ** (model.decoder.num_resolutions - 1)
    make_cutouts = MakeCutouts(cut_size, cutn, cut_pow=cut_pow)
    toksX, toksY = size[0] // f, size[1] // f
    sideX, sideY = toksX * f, toksY * f

    if args.vqgan_checkpoint == 'vqgan_openimages_f16_8192.ckpt':
        e_dim = 256
        n_toks = model.quantize.n_embed
        z_min = model.quantize.embed.weight.min(dim=0).values[None, :, None,
        z_max = model.quantize.embed.weight.max(dim=0).values[None, :, None,
    else:
        e_dim = model.quantize.e_dim
        n_toks = model.quantize.n_e
        z_min = model.quantize.embedding.weight.min(dim=0).values[None, :, N
        z_max = model.quantize.embedding.weight.max(dim=0).values[None, :, N

    # Initialize or load latent vector (z)
    if init_image:
```

```
        img = Image.open(urlopen(init_image) if 'http' in init_image else in
        pil_image = img.convert('RGB')
        pil_image = pil_image.resize((sideX, sideY), Image.LANCZOS)
        pil_tensor = TF.to_tensor(pil_image)
        z, *_ = model.encode(pil_tensor.to(device).unsqueeze(0) * 2 - 1)
    else:
        one_hot = F.one_hot(torch.randint(n_toks, [toksY * toksX], device=de
        z = one_hot @ model.quantize.embed.weight if args.vqgan_checkpoint =
        z = z.view([-1, toksY, toksX, e_dim]).permute(0, 3, 1, 2)
        z = torch.rand_like(z) * 2
    z_orig = z.clone()
    z.requires_grad_(True)
    opt = optim.Adam([z], lr=step_size)

    # Normalization transformation
    normalize = transforms.Normalize(mean=[0.48145466, 0.4578275, 0.40821073
                                           std=[0.26862954, 0.26130258, 0.27577111
    pMs = []

    # Parse and create prompt objects for text prompts
    for prompt in texts:
        txt, weight, stop = parse_prompt(prompt)
        embed = perceptor.encode_text(clip.tokenize(txt).to(device)).float()
        pMs.append(Prompt(embed, weight, stop).to(device))

    # Parse and create prompt objects for image prompts
    for prompt in target_images:
        path, weight, stop = parse_prompt(prompt)
        img = Image.open(path)
        pil_image = img.convert('RGB')
        img = resize_image(pil_image, (sideX, sideY))
        batch = make_cutouts(TF.to_tensor(img).unsqueeze(0).to(device))
        embed = perceptor.encode_image(normalize(batch)).float()
        pMs.append(Prompt(embed, weight, stop).to(device))

    # Parse and create prompt objects for noise prompts
    for seed, weight in zip(args.noise_prompt_seeds, args.noise_prompt_weigh
        gen = torch.Generator().manual_seed(seed)
        embed = torch.empty([1, perceptor.visual.output_dim]).normal_(genera
        pMs.append(Prompt(embed, weight).to(device))

    # Synthesize function
    def synth(z):
        z_q = vector_quantize(z.movedim(1, 3), model.quantize.embed.weight)
        return clamp_with_grad(model.decode(z_q).add(1).div(2), 0, 1)

    # Check-in function
    @torch.no_grad()
    def checkin(i, losses):
        losses_str = ', '.join(f'{loss.item():g}' for loss in losses)
        tqdm.write(f'i: {i}, loss: {sum(losses).item():g}, losses: {losses_s
        out = synth(z)
        res = nvmlDeviceGetUtilizationRates(handle)
        print(f'gpu: {res.gpu}%, gpu-mem: {res.memory}%')

    # Ascend text function
    def ascend_txt():
        out = synth(z)
        iii = perceptor.encode_image(normalize(make_cutouts(out))).float()
        result = []
        if init_weight:
            result.append(F.mse_loss(z, z_orig) * init_weight / 2)
        for prompt in pMs:
            result.append(prompt(iii))
```

```
        img = np.array(out.mul(255).clamp(0, 255)[0].cpu().detach().numpy()
        img = np.transpose(img, (1, 2, 0))
        img = Image.fromarray(img).convert('RGB')
        all_frames.append(img)
        return result, np.array(img)

    # Training function
    def train(i):
        opt.zero_grad()
        lossAll, image = ascend_txt()
        if i % args.display_freq == 0:
            checkin(i, lossAll)
        loss = sum(lossAll)
        loss.backward()
        opt.step()
        with torch.no_grad():
            z.copy_(z.maximum(z_min).minimum(z_max))
        return image

    # Training loop
    i = 0
    try:
        with tqdm() as pbar:
            while True:
                image = train(i)
                if i == max_iterations:
                    break
                i += 1
                pbar.update()
    except KeyboardInterrupt:
        pass

    # Generate video file
    writer = imageio.get_writer(video_file + '.mp4', fps=20)
    for im in all_frames:
        writer.append_data(np.array(im))
    writer.close()

    return image
```

In [14]:

```
# Define a function to load an image
def load_image(infilename):
    # Open the image file using PIL
    img = Image.open(infilename)

    # Load the image data
    img.load()

    # Convert the image to a NumPy array
    data = np.asarray(img, dtype="int32")

    return data
```

In [15]:

```
# Define a function to display an image
def display_result(img):
    # Create a new figure with a specific size
    plt.figure(figsize=(9, 9))

    # Display the image
    plt.imshow(img)

    # Turn off axis labels
    plt.axis('off')
```

```
    # Show the plot
    plt.show()
```

In [17]:

```
def generate_and_display_image(
    text='',
    seed=43,
    step_size=0.12,
    max_iterations=700,
    width=512,
    height=512,
    init_image='',
    init_weight=0.004,
    target_images='',
    cutn=64,
    cut_pow=0.3,
    video_file=''
):
    # Generate the image using the provided parameters
    img = inference(
        text=text,
        seed=seed,
        step_size=step_size,
        max_iterations=max_iterations,
        width=width,
        height=height,
        init_image=init_image,
        init_weight=init_weight,
        target_images=target_images,
        cutn=cutn,
        cut_pow=cut_pow,
        video_file=video_file
    )

    # Display the generated image
    display_result(img)

generate_and_display_image(
    text='anime character with three swords',
    seed=43,
    init_weight=0.004,
    cutn=64,
    cut_pow=0.3,
    video_file="example_1"
)
```

```
0it [00:00, ?it/s]
```

```
i: 0, loss: 0.92817, losses: 0, 0.92817
gpu: 66%, gpu-mem: 18%
i: 50, loss: 0.877018, losses: 0.00169803, 0.87532
gpu: 72%, gpu-mem: 12%
i: 100, loss: 0.775899, losses: 0.00281867, 0.77308
gpu: 95%, gpu-mem: 21%
i: 150, loss: 0.828539, losses: 0.00316001, 0.825379
gpu: 72%, gpu-mem: 20%
i: 200, loss: 0.793168, losses: 0.00342172, 0.789746
gpu: 72%, gpu-mem: 17%
i: 250, loss: 0.742206, losses: 0.00369146, 0.738515
gpu: 100%, gpu-mem: 31%
i: 300, loss: 0.737453, losses: 0.00389778, 0.733555
gpu: 95%, gpu-mem: 20%
i: 350, loss: 0.734341, losses: 0.00411928, 0.730222
gpu: 95%, gpu-mem: 23%
i: 400, loss: 0.723635, losses: 0.00435511, 0.71928
gpu: 95%, gpu-mem: 24%
i: 450, loss: 0.83591, losses: 0.00453, 0.83138
gpu: 55%, gpu-mem: 12%
i: 500, loss: 0.768376, losses: 0.00473275, 0.763643
gpu: 55%, gpu-mem: 15%
i: 550, loss: 0.721116, losses: 0.00487109, 0.716244
gpu: 95%, gpu-mem: 25%
i: 600, loss: 0.778083, losses: 0.00503086, 0.773052
gpu: 58%, gpu-mem: 17%
i: 650, loss: 0.752871, losses: 0.00519929, 0.747672
gpu: 52%, gpu-mem: 12%
i: 700, loss: 0.735195, losses: 0.00536637, 0.729829
gpu: 67%, gpu-mem: 11%
```

In [18]:
```
from IPython.display import HTML
from base64 import b64encode
```

```
mp4 = open('example_1.mp4','rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
HTML("""
<video width=500 loop="true" autoplay="autoplay" controls muted>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Out[18]:

In [19]:

```
generate_and_display_image(
    text='cosmic lion',
    step_size=0.14,
    max_iterations=300,
    width=512,
    height=512,
    init_image='',
    init_weight=0.04,
    target_images='',
    cutn=64,
    cut_pow=1.0,
    video_file="example_2"
)
```

```
0it [00:00, ?it/s]
```

```
i: 0, loss: 0.915396, losses: 0, 0.915396
gpu: 61%, gpu-mem: 17%
i: 50, loss: 0.853648, losses: 0.0217808, 0.831867
gpu: 72%, gpu-mem: 17%
i: 100, loss: 0.707277, losses: 0.0244697, 0.682807
gpu: 100%, gpu-mem: 29%
i: 150, loss: 0.738751, losses: 0.022894, 0.715857
gpu: 74%, gpu-mem: 22%
i: 200, loss: 0.718252, losses: 0.0224477, 0.695804
gpu: 73%, gpu-mem: 14%
i: 250, loss: 0.676772, losses: 0.0222299, 0.654542
gpu: 100%, gpu-mem: 31%
i: 300, loss: 0.676761, losses: 0.0224752, 0.654286
gpu: 95%, gpu-mem: 25%
```

In [20]:

```
mp4 = open('example_2.mp4','rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
HTML("""
<video width=500 loop="true" autoplay="autoplay" controls muted>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Out[20]:

In [21]:

```
# Call the function with your specific parameters
generate_and_display_image(
    text='empty haunted house',
    step_size=0.13,
    max_iterations=700,
    width=512,
    height=512,
    init_image='',
    init_weight=0.0,
    target_images='',
    cutn=64,
    cut_pow=1.0,
    video_file="example_3"
)
```

```
0it [00:00, ?it/s]
```

```
i: 0, loss: 0.916162, losses: 0.916162
gpu: 66%, gpu-mem: 14%
i: 50, loss: 0.793436, losses: 0.793436
gpu: 82%, gpu-mem: 26%
i: 100, loss: 0.720804, losses: 0.720804
gpu: 94%, gpu-mem: 27%
i: 150, loss: 0.721519, losses: 0.721519
gpu: 72%, gpu-mem: 19%
i: 200, loss: 0.709913, losses: 0.709913
gpu: 73%, gpu-mem: 21%
i: 250, loss: 0.685404, losses: 0.685404
gpu: 95%, gpu-mem: 28%
i: 300, loss: 0.683722, losses: 0.683722
gpu: 95%, gpu-mem: 21%
i: 350, loss: 0.681055, losses: 0.681055
gpu: 95%, gpu-mem: 22%
i: 400, loss: 0.677296, losses: 0.677296
gpu: 93%, gpu-mem: 20%
i: 450, loss: 0.76217, losses: 0.76217
gpu: 72%, gpu-mem: 17%
i: 500, loss: 0.731028, losses: 0.731028
gpu: 73%, gpu-mem: 15%
i: 550, loss: 0.676941, losses: 0.676941
gpu: 95%, gpu-mem: 27%
i: 600, loss: 0.744732, losses: 0.744732
gpu: 91%, gpu-mem: 29%
i: 650, loss: 0.724525, losses: 0.724525
gpu: 73%, gpu-mem: 23%
i: 700, loss: 0.704674, losses: 0.704674
gpu: 55%, gpu-mem: 12%
```

In [22]:
```
mp4 = open('example_3.mp4','rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
```

```
HTML("""
<video width=500 loop="true" autoplay="autoplay" controls muted>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Out[22]:

In [23]:

```
generate_and_display_image(
    text='footballer shooting penalty',
    seed=27560,
    step_size=0.14,
    max_iterations=700,
    width=512,
    height=512,
    init_image='',
    init_weight=0.0,
    target_images='',
    cutn=3,
    cut_pow=1.0,
    video_file="example_4"
)
```

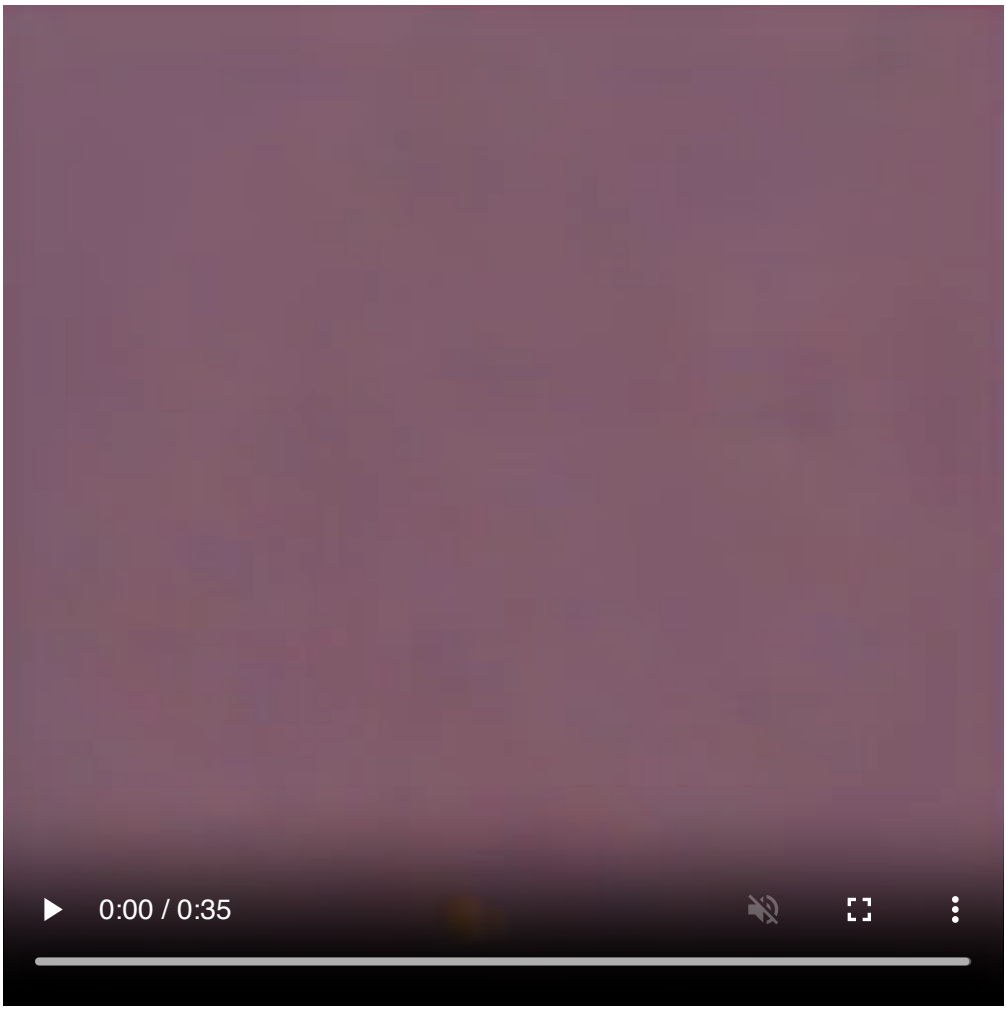
```
0it [00:00, ?it/s]
```

```
i: 0, loss: 0.910952, losses: 0.910952
gpu: 86%, gpu-mem: 27%
i: 50, loss: 0.908813, losses: 0.908813
gpu: 100%, gpu-mem: 26%
i: 100, loss: 0.794729, losses: 0.794729
gpu: 88%, gpu-mem: 28%
i: 150, loss: 0.851476, losses: 0.851476
gpu: 100%, gpu-mem: 30%
i: 200, loss: 0.752212, losses: 0.752212
gpu: 87%, gpu-mem: 28%
i: 250, loss: 0.762851, losses: 0.762851
gpu: 100%, gpu-mem: 29%
i: 300, loss: 0.742941, losses: 0.742941
gpu: 100%, gpu-mem: 27%
i: 350, loss: 0.7366, losses: 0.7366
gpu: 100%, gpu-mem: 29%
i: 400, loss: 0.761527, losses: 0.761527
gpu: 100%, gpu-mem: 27%
i: 450, loss: 0.73562, losses: 0.73562
gpu: 100%, gpu-mem: 31%
i: 500, loss: 0.734978, losses: 0.734978
gpu: 100%, gpu-mem: 27%
i: 550, loss: 0.717632, losses: 0.717632
gpu: 93%, gpu-mem: 30%
i: 600, loss: 0.707972, losses: 0.707972
gpu: 100%, gpu-mem: 28%
i: 650, loss: 0.665329, losses: 0.665329
gpu: 100%, gpu-mem: 29%
i: 700, loss: 0.716335, losses: 0.716335
gpu: 100%, gpu-mem: 27%
```

In [24]:
```
mp4 = open('example_4.mp4','rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
```

```
HTML("""
<video width=500 loop="true" autoplay="autoplay" controls muted>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Out[24]:



In [25]:

```
generate_and_display_image(
    text='Cyberpunk Dogs',
    seed=100,
    step_size=0.14,
    max_iterations=700,
    width=512,
    height=512,
    init_image='/kaggle/input/pexel-data/pexels-chevanon-photography-1108099
    init_weight=0.0,
    target_images='',
    cutn=32,
    cut_pow=1.0,
    video_file="example_5"
)
```

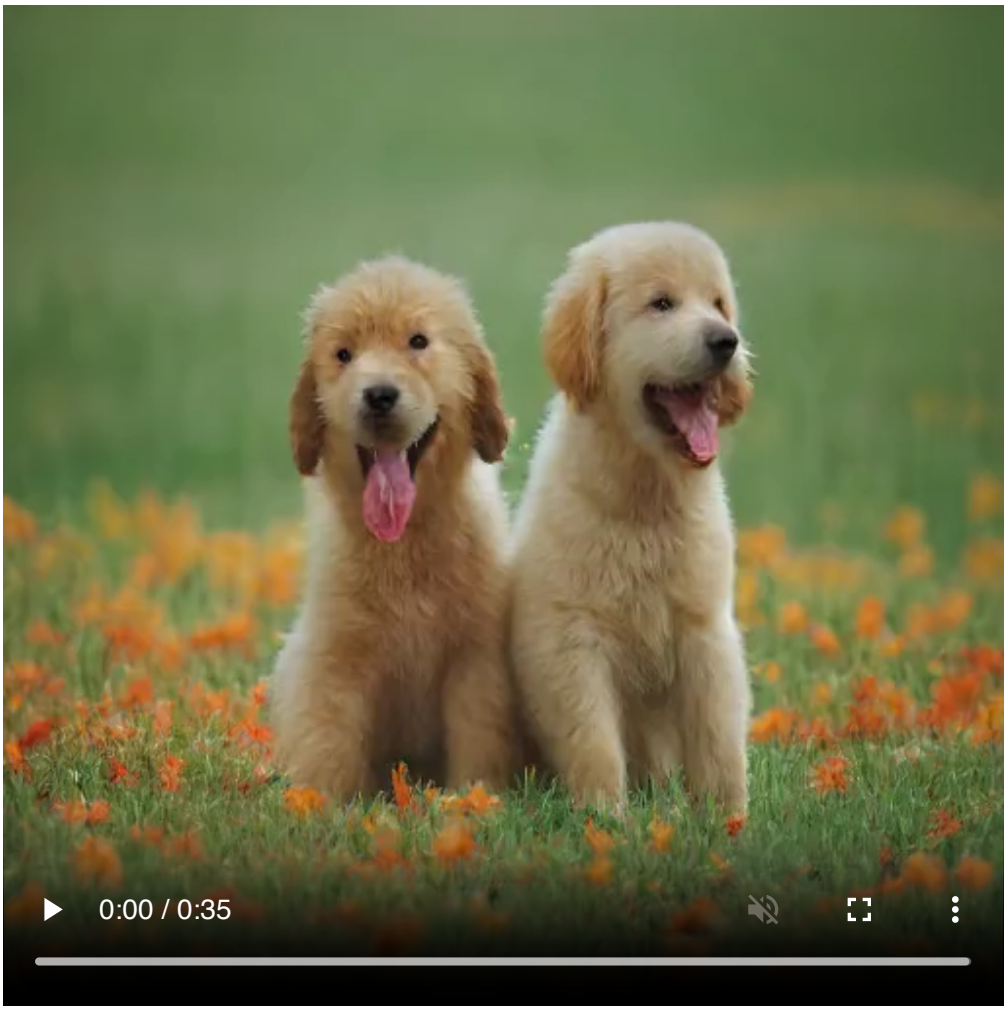
```
0it [00:00, ?it/s]
```

```
i: 0, loss: 0.89715, losses: 0.89715
gpu: 93%, gpu-mem: 28%
i: 50, loss: 0.719848, losses: 0.719848
gpu: 100%, gpu-mem: 31%
i: 100, loss: 0.680905, losses: 0.680905
gpu: 100%, gpu-mem: 29%
i: 150, loss: 0.741905, losses: 0.741905
gpu: 96%, gpu-mem: 31%
i: 200, loss: 0.636082, losses: 0.636082
gpu: 91%, gpu-mem: 25%
i: 250, loss: 0.701, losses: 0.701
gpu: 100%, gpu-mem: 31%
i: 300, loss: 0.633443, losses: 0.633443
gpu: 90%, gpu-mem: 25%
i: 350, loss: 0.701141, losses: 0.701141
gpu: 82%, gpu-mem: 21%
i: 400, loss: 0.663844, losses: 0.663844
gpu: 84%, gpu-mem: 23%
i: 450, loss: 0.624777, losses: 0.624777
gpu: 92%, gpu-mem: 29%
i: 500, loss: 0.636248, losses: 0.636248
gpu: 96%, gpu-mem: 30%
i: 550, loss: 0.758093, losses: 0.758093
gpu: 96%, gpu-mem: 30%
i: 600, loss: 0.6209, losses: 0.6209
gpu: 95%, gpu-mem: 26%
i: 650, loss: 0.717486, losses: 0.717486
gpu: 88%, gpu-mem: 29%
i: 700, loss: 0.614169, losses: 0.614169
gpu: 96%, gpu-mem: 30%
```

In [26]:
```
mp4 = open('example_5.mp4','rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
```

```
HTML("""
<video width=500 loop="true" autoplay="autoplay" controls muted>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Out[26]:



In [27]:

```
generate_and_display_image(
    text='island with birds flying above',
    seed=1632972250290523,
    step_size=0.14,
    max_iterations=700,
    width=512,
    height=512,
    init_image='',
    init_weight=0.0,
    target_images='',
    cutn=32,
    cut_pow=1.0,
    video_file="example_6"
)
```

```
0it [00:00, ?it/s]
```

```
i: 0, loss: 0.91905, losses: 0.91905
gpu: 91%, gpu-mem: 30%
i: 50, loss: 0.831787, losses: 0.831787
gpu: 100%, gpu-mem: 32%
i: 100, loss: 0.746477, losses: 0.746477
gpu: 100%, gpu-mem: 31%
i: 150, loss: 0.727265, losses: 0.727265
gpu: 84%, gpu-mem: 25%
i: 200, loss: 0.781486, losses: 0.781486
gpu: 77%, gpu-mem: 19%
i: 250, loss: 0.766035, losses: 0.766035
gpu: 82%, gpu-mem: 23%
i: 300, loss: 0.731921, losses: 0.731921
gpu: 100%, gpu-mem: 31%
i: 350, loss: 0.66021, losses: 0.66021
gpu: 97%, gpu-mem: 30%
i: 400, loss: 0.70778, losses: 0.70778
gpu: 84%, gpu-mem: 27%
i: 450, loss: 0.704431, losses: 0.704431
gpu: 88%, gpu-mem: 29%
i: 500, loss: 0.685408, losses: 0.685408
gpu: 84%, gpu-mem: 23%
i: 550, loss: 0.662847, losses: 0.662847
gpu: 100%, gpu-mem: 29%
i: 600, loss: 0.710032, losses: 0.710032
gpu: 83%, gpu-mem: 23%
i: 650, loss: 0.735147, losses: 0.735147
gpu: 89%, gpu-mem: 29%
i: 700, loss: 0.689392, losses: 0.689392
gpu: 81%, gpu-mem: 23%
```

In [28]:
```
mp4 = open('example_6.mp4','rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
```

```
HTML("""
<video width=500 loop="true" autoplay="autoplay" controls muted>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Out[28]:

In [29]:

```
generate_and_display_image(
    text='underwater fish fight',
    step_size=0.12,
    max_iterations=400,
    width=512,
    height=512,
    init_image='/kaggle/input/pexel-data/pexels-neha-pandey-2446439.jpg',
    init_weight=0.0,
    target_images='',
    cutn=64,
    cut_pow=1.0,
    video_file="example_7"
)
```

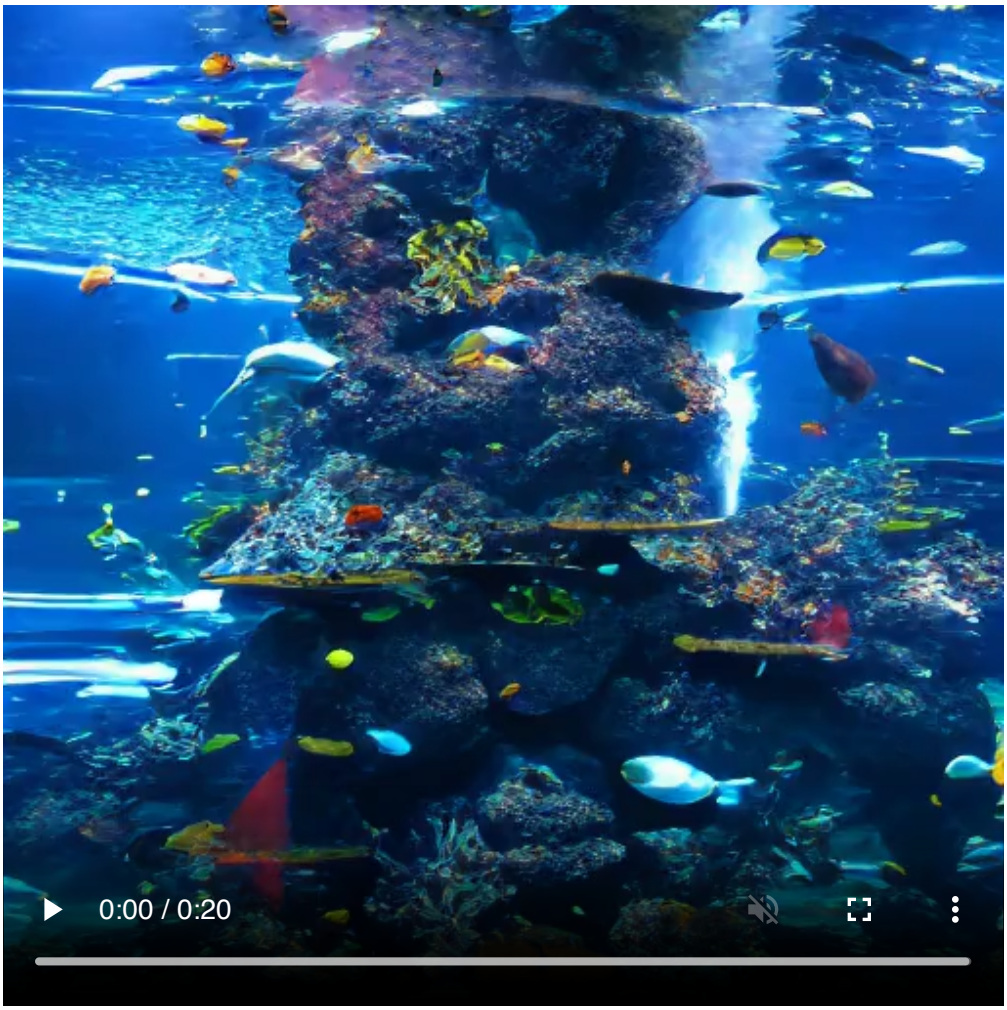
```
0it [00:00, ?it/s]
```

```
i: 0, loss: 0.835862, losses: 0.835862
gpu: 88%, gpu-mem: 28%
i: 50, loss: 0.753339, losses: 0.753339
gpu: 91%, gpu-mem: 29%
i: 100, loss: 0.708832, losses: 0.708832
gpu: 95%, gpu-mem: 28%
i: 150, loss: 0.73367, losses: 0.73367
gpu: 73%, gpu-mem: 19%
i: 200, loss: 0.716737, losses: 0.716737
gpu: 74%, gpu-mem: 19%
i: 250, loss: 0.685464, losses: 0.685464
gpu: 100%, gpu-mem: 31%
i: 300, loss: 0.684433, losses: 0.684433
gpu: 99%, gpu-mem: 29%
i: 350, loss: 0.684371, losses: 0.684371
gpu: 94%, gpu-mem: 24%
i: 400, loss: 0.679642, losses: 0.679642
gpu: 95%, gpu-mem: 26%
```

In [30]:

```
mp4 = open('example_7.mp4','rb').read()
data_url = "data:video/mp4;base64," + b64encode(mp4).decode()
HTML("""
<video width=500 loop="true" autoplay="autoplay" controls muted>
      <source src="%s" type="video/mp4">
</video>
""" % data_url)
```

Out[30]:



In [ ]: