IMAGE GENERATION & VIDEO GENERATION FROM TEXT PROMPT USING CLIP & TRANSFORMER

WE ARE GOING TO WORK WITH MULTIMODEL ARCHITECTURE CALLED CLIP THAT IS GOING TO LINK TEXT VISUAL ELEMENT & WE ARE GOING TO COMBINE WITH GENERATIVE MODEL WITH TRANSFORMER SO THAT WE CAN TAKE TEXT PROMPT AND GENERATE VISUAL PROMPT AND EVEN VIDEOS CAN MAKE WITH SEQUENCE. THIS IS CUTTING EDGE GENERATIVE AI WORKSHOP.

WHAT IS CLIP ARCHITECTURE? → **!git clone https://github.com/openai/CLIP**

- ,CLIP ARCHITECTURE BY OPEN AI FROM THE PAPER CALLED LEARNING TRANSFERABLE VISUAL MODEL FROM NATURAL LANGUAGE SUPERVISION . THEY CREATED THIS GENERATIVE ARCHITECTURE TO CONNECT TEXT & IMAGE . THIS ARCHITECTURETRAINED WITH LOT OF IMAGES & LOT OF TEXT TO PREDICT THE IMAGES USING TEXT ENCODER & IMAGE ENCODER

(LEARNING TRANSFERABLE VISUAL MODELS FROM NATURAL LANGUAGE SUPERVISION) Refer to the pdf for more information.

TAMING-TRANSFORMER ARCHITECTURE →

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

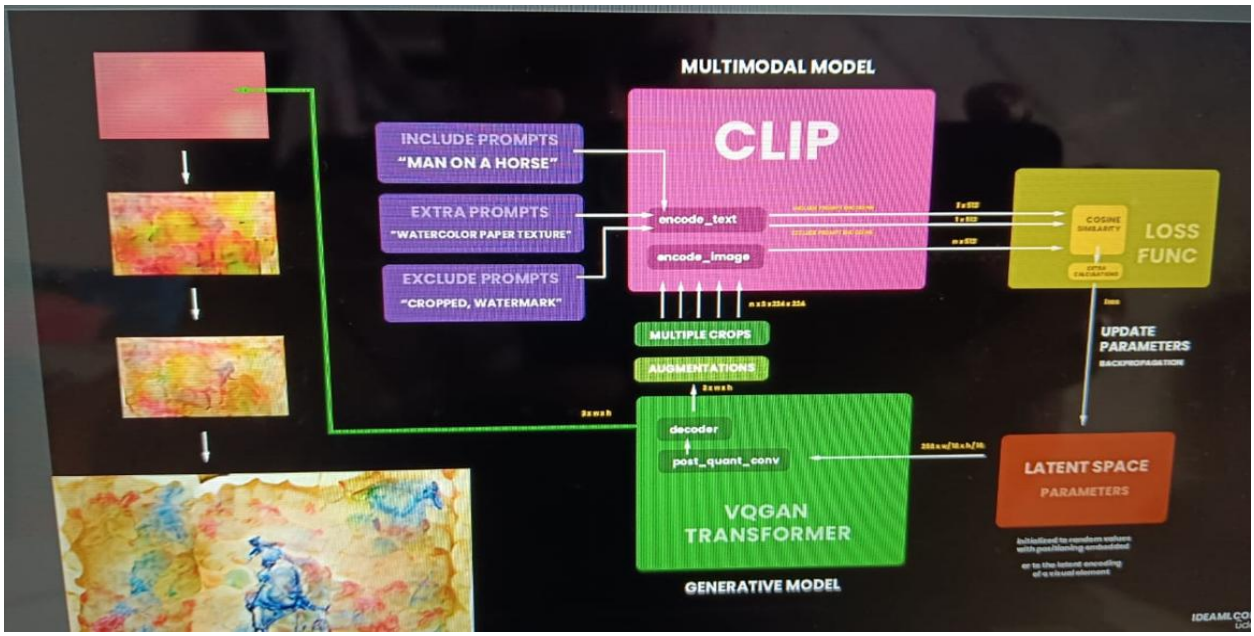
TAMING TRANSFORMERS FOR HIGH-RESOLUTION IMAGE SYNTHESIS (AKA – VQGAN)

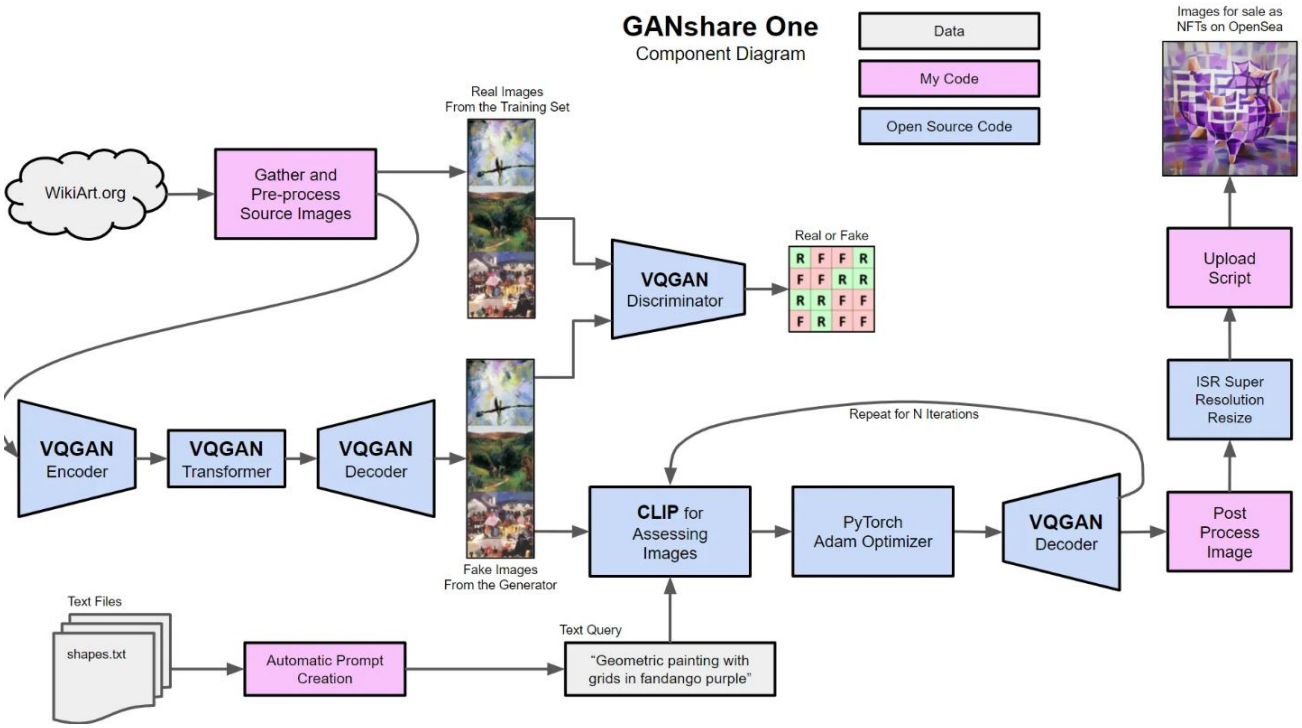
https://compvis.github.io/taming-transformers

WE TRY TO ADD ANOTHER ARCHITECTURE VQGAN. WHAT **IS VQGAN? (VECTOR QUANTIZED GENERATIVE ADVERSIAL NETWORK)**

Image generation: VQGAN generates images using a GAN architecture, which learns to generate images from random noise.

Quantization: VQGAN applies vector quantization to the output of the GAN, which divides the continuous-valued image into a fixed number of discrete codebook vectors.





PRACTICLE STEPS →

1- GOOGLE COLAB →

We will work this project in Google Colab (CPU it wont work because need more high configuration system)

2- DOWNLOAD 2 ARCHITECTURE→

Let's download 2 architecture from Github using !!!! Git clone (Clip & Vqgan)

3- INSTALLATION →

Install 2 architecture 2 folder created in the Colab

pip install --no-deps ftfy regex tqdm		**This command installs the specified packages (ftfy, regex, tqdm) installing without the dependencies.**
pip install ftfy regex tqdm		**If you want to install the packages along with their dependencies,**
!pip install omegaconf==2.0.0 pytorch-lightning==1.0.8		**install specific versions of the omegaconf and pytorch-lightning packages using the pip command.**
!pip uninstall torchtext --yes		**This command uninstalls the torchtext package, and the --yes flag automatically confirms the uninstallation without prompting for user confirmation.**
!pip install einops		**This command installs the einops library, which is commonly used in machine learning and deep learning workflows for reshaping and manipulating tensors. Once the installation is complete, you can import and use einops in your Python scripts or Jupyter notebook**
import PIL		**import the Python Imaging Library (PIL)**
Numpy, Pandas, Matplotlib		**Multidimension Array || DataFrame || Visualization**

import torch, os, imageio, pdb, math	torch	This is the PyTorch library, a popular deep learning library in Python. It provides tensor computations and supports building and training neural networks.
	os	The os module provides a way of interacting with the operating system. It allows you to perform various operating system-related tasks, such as reading or writing files and directories.
	imageio	The imageio library is used for reading and writing images in various formats. It is commonly used in computer vision tasks.
	pdb	The pdb module is the Python Debugger. It allows you to interactively debug your Python code, set breakpoints, and inspect variables during program
	math	The math module provides mathematical functions and constants. It includes functions like sin, cos, sqrt,
import yaml		The yaml library is commonly used for working with YAML (YAML Ain't Markup Language) files in Python. YAML is a human-readable data serialization format that is often used for configuration files, data exchange between languages with different data struct

4- CREATE HELPER FUNCTION() ➔

Lets create some helper function which will be helpful for us that shown image from tensor. Tensor is A tensor is a multi-dimensional array of numbers. It can have any number of dimensions (0-D, 1-D, 2-D, ..., N-D). then clone the tensor & we take the data multiply by 255 cuz pixel range is 0-255 .

Transpose(1,2,0) refers to specific permutations of the axes in 3d array or tensor. This operation is performed for array manipulation for Deep learning frame work numpy & py-torch.
Transpro(1,2,0) indicate that the original axes order should be rearranged so that or that the original axes order should be rearranged so that the original axis 1 becomes the new axis 0, the original axis 2 becomes the new axis 1, and the original axis 0 becomes the new axis 2.

data.clip(-1, 1) + 2 involves two operations: the clip method and addition.

1. data.clip(-1, 1):
   - The clip method is used to limit the values of the array (or data) to be within a specified range. In this case, it is restricting the values to be between -1 and 1. If any value in the original data array is less than -1, it is set to -1; if it is greater than 1, it is set to 1. Values within the range (-1, 1) remain unchanged.
2. + 2:
   - After applying the clip operation, the resulting array is then added to 2. This means that 2 is added to each element of the clipped array.

when training neural networks, the term **wd** likely stands for "weight decay." Weight decay is a regularization technique used to prevent overfitting in a model by penalizing large weights.

The parameter **wd = 0.1** implies that a weight decay term is added to the loss function during training, and the regularization strength is set to 0.1. The weight decay term is usually proportional to the sum of squares of all model parameters (weights).

noise_factor = 0.1 likely refers to the strength or intensity of the noise that is being added to the images.

When working with image data, introducing a controlled amount of noise can be a form of regularization, helping the model generalize better by making it less sensitive to small variations in the input data.

5- SET UP CLIP MODEL ➔

Clipmodel – It will convert text to image & image-text matching, zero-shot classification, etc.
_ underscore (returned by the clip.load function)
Clip.load -> This function is part of the CLIP library and is used to load pre-trained CLIP models. The argument 'ViT-B/32' specifies the specific variant of the CLIP model to load (in this case, ViT-B/32).
Jit =false -> Just-In-Time (JIT) compilation should not be used. JIT compilation is a technique that can improve the execution speed of the model, but in this case, it seems to be turned off.
Clipmodel_eval() == After calling clipmodel.eval(), the CLIP model is set to evaluation mode. If the model contains layers such as dropout or batch normalization, these layers will now work in evaluation mode.

6. SETUP TAMING TRANSFORMER ➔

%cd taming-transformers/ ➔ changing the working directory to the "taming-transformers" directory.

!mkdir -p models/vqgan_imagenet_f16_16384/checkpoints ➔ This is the path where the directory structure will be created. It includes nested directories.

!mkdir -p models/vqgan_imagenet_f16_16384/configs

- models: This is the main directory.
- vqgan_imagenet_f16_16384: This is a subdirectory within "models."
- checkpoints: This is another subdirectory within "vqgan_imagenet_f16_16384."
- `configs`: This is another subdirectory within "vqgan_imagenet_f16_16384."

The -p flag ensures that all these directories are created, even if "models" or "vqgan_imagenet_f16_16384" do not exist yet.

if len(os.listdir('models/vqgan_imagenet_f16_16384/checkpoints/')) == 0: ➔

The code snippet checks whether the directory 'models/vqgan_imagenet_f16_16384/checkpoints/' is empty by using the os.listdir() function to list the files in the directory and then checking the length of the list.

!wget 'https://heibox.uni-heidelberg.de/f/867b05fc8c4841768640/?dl=1' -O 'models/vqgan_imagenet_f16_16384/checkpoints/last.ckpt' ➔

- `!wget`: This is a shell command used to download files from the internet.
- `'https://heibox.uni-heidelberg.de/f/867b05fc8c4841768640/?dl=1'`: This is the URL from which the file is being downloaded. The `?dl=1` in the URL is often used with services like Dropbox to force a direct download.
- `-O 'models/vqgan_imagenet_f16_16384/checkpoints/last.ckpt'`: This specifies the output file name and path where the downloaded file will be saved. In this case, it's saving the file as `last.ckpt` in the `checkpoints` directory.

So, this command is downloading a file from the given URL and saving it as `last.ckpt` in the specified directory. After running this command, you should have the `last.ckpt` file in the `checkpoints` directory.

!wget 'https://heibox.uni-heidelberg.de/f/274fb24ed38341bfa753/?dl=1' -O 'models/vqgan_imagenet_f16_16384/configs/model.yaml' ➔

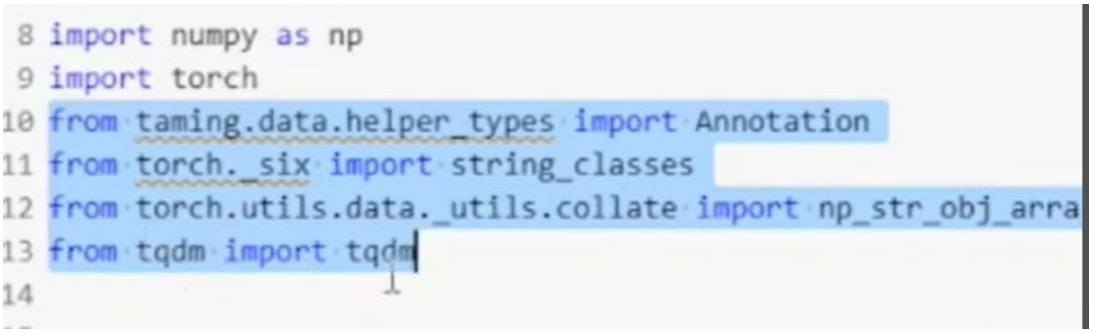
downloading a YAML configuration file from a different URL and saving it as model.yaml in the specified directory.

- `'https://heibox.uni-heidelberg.de/f/274fb24ed38341bfa753/?dl=1'`: This is the URL from which the file (in this case, a YAML configuration file) is being downloaded. The `?dl=1` in the URL is often used with services like Dropbox to force a direct download.
- `-O 'models/vqgan_imagenet_f16_16384/configs/model.yaml'`: This specifies the output file name and path where the downloaded file will be saved. In this case, it's saving the file as `model.yaml` in the `configs` directory.

After running this command, you should have the `model.yaml` file in the `configs` directory. This YAML file likely contains configuration settings for the VQ-VAE-2 model, specifying parameters such as model architecture, training hyperparameters, and other relevant settings.

7. PROCEED THE TAMING TRANSFORMER VQGAN ARCHITECTURE ➔

ERROR ➔ Module_not found (torch-six) || FIX ➔ click the link above above error message & remove the selected 4 line



8. Create the class to declare the values that we are going to optimize➔

In this code we are going to create class with parameter torch with module. We make this parameter as super class (paraent class) & we are going to initialize value with random values. Using this class it will produce the tansor of random values

9. Create few function to Encoding prompts➔

Write the function for using clip architecture to encoding text prompt.

pythonCopy code

```
normalize = torchvision.transforms.Normalize((0.48145466, 0.4578275, 0.40821073), (0.26862954, 0.26130258, 0.27577711))
```

- torchvision.transforms.Normalize: This is a transformation from the torchvision library that normalizes an input image.
- (mean_values): The first tuple contains the mean values for each channel (Red, Green, Blue). In this case, the mean values are (0.48145466, 0.4578275, 0.40821073).
- (std_dev_values): The second tuple contains the standard deviation values for each channel. In this case, the standard deviation values are (0.26862954, 0.26130258, 0.27577711).

We write the few functions to receive the text and then it will call the clip model & tokenize the text . text will converted to tokens and user get result of the token & then clone the result.

Whenever we create our prompt user need to specify 3 things -include, exclude, extra
Include-- I want to enter blue tree in the forest or elephants are dancing etc. like text prompt
Exclude – it should exclude the green color or anything user want.
Extra- some extrac color need to be applied for spaced to create generate images.

```
pythonCopy code
augTransform = torch.nn.Sequential( torchvision.transforms.RandomHorizontalFlip(), torchvision.transforms.RandomAffine(30, (.2, .2), fill=0) ).cuda()
```

- `torch.nn.Sequential`: This is a container module that sequentially applies a list of transformations or layers.
- `torchvision.transforms.RandomHorizontalFlip()`: This transformation randomly flips the input images horizontally. It's a common augmentation technique to increase the diversity of the training dataset.
- `torchvision.transforms.RandomAffine(30, (.2, .2), fill=0)`: This transformation applies a random affine transformation to the input images. The parameters are:
  - `30`: Maximum rotation angle in degrees.
  - `(.2, .2)`: Range of scaling factors.
  - `fill=0`: The fill value for pixels outside the image boundaries after the transformation (in this case, filling with zeros).
- `.cuda()`: This moves the entire sequence of transformations (augmentation pipeline) to the GPU. It's used for GPU acceleration during training.

Before generating any images we are going to create crops of our images, we will send the image when we encode the image from clip.
We try to initialize and optimize the parameter, lets make simple text to image generate using generator functions & it will convert simple text to image generate throught our transformer generated initial parameter.

10. CREATE CROPS➔

We need to create crops . when we encode the image with clip architecture you just no need to send only single image many number of crops of images with augumentation , rotation, translation, you will give different variation of the image & clip can understand the various essence of the image for better image.

What is mean by crating crops ➔

In the context of CLIP (Contrastive Language-Image Pre-training), "crops" typically refers to the image processing technique used during training. CLIP uses a combination of text and image inputs to learn a joint embedding space where semantically similar text and images are close to each other.

During training, images are divided or cropped into several patches or regions, and these regions are treated as separate inputs to the model. The model then learns to associate each cropped region with the corresponding textual description. This process is part of the contrastive learning objective, where the model is trained to bring positive pairs (correct image-text pairs) closer in the embedding space and push negative pairs (incorrect pairs) farther apart.

The use of crops in CLIP training is a form of data augmentation. By presenting different views of the same image to the model, it helps the model generalize better and learn robust representations. Different crops from the same image are considered positive samples, and crops from different images are considered negative samples during the training process.

The specific details of how crops are applied, the size of the crops, and other hyperparameters may vary depending on the implementation or version of the CLIP architecture. It's a common technique in vision-language pre-training to improve the model's ability to understand and relate images and text.

11. FUNCTION TO DISPLAY GENERATED IMAGES & CROPS➔

12. OPTIMIZE THE PARAMETER BEFORE TRAINING THE LOOP➔

Before training a model, it's crucial to optimize the hyperparameters and settings to ensure the training process is efficient and effective. Here are some common steps to optimize parameters before training:

1. Hyperparameter Tuning:

a. Learning Rate:

- Experiment with different learning rates. Learning rate too high can lead to divergence, while too low can result in slow convergence.

pythonCopy code

```
# Example learning rate schedule in PyTorch optimizer = optim.Adam(model.parameters(), lr=initial_lr)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.5)
```

b. Batch Size:

- Adjust the batch size to find a balance between faster training and memory requirements.

c. Number of Epochs:

- Choose an appropriate number of epochs. Training too few epochs may result in underfitting, while training too many may lead to overfitting.

d. Architecture-specific Hyperparameters:

- Tune hyperparameters specific to your model architecture, such as the number of layers, hidden units, etc.

2. Loss Function:

Choose or customize a loss function suitable for your task. For GANs, you typically have a generator loss and a discriminator loss. For VAEs, you have a reconstruction loss and regularization terms. The choice of loss function can have a significant impact on model performance.

3. Regularization:

Apply regularization techniques such as dropout, weight decay, or batch normalization to prevent overfitting.

pythonCopy code

```
# Example of weight decay (L2 regularization) in PyTorch optimizer = optim.Adam(model.parameters(), lr=initial_lr, weight_decay=1e-5)
```

4. Data Augmentation:

Apply data augmentation techniques if necessary, especially if your dataset is limited. Augmentations can include random rotations, flips, and translations.

5. Monitor Metrics:

Monitor relevant metrics during training to assess model performance. Common metrics include training loss, validation loss, accuracy, or other domain-specific metrics.

6. Pretrained Models:

Consider using pretrained models or transfer learning if applicable. Pretrained models can provide a good starting point and help in convergence, especially when dealing with limited datasets.

7. Hardware Acceleration:

Leverage hardware acceleration such as GPUs or TPUs if available. This can significantly speed up the training process.

8. Debugging:

Regularly inspect and debug your code. Check for any issues in data loading, preprocessing, or model architecture.

9. Cross-Validation:

If your dataset allows, consider using cross-validation to assess your model's generalization performance more robustly.

13. TRAINING THE LOOP & RUN THE TRAINING LOOP TO CREATE THE TEXT PROMPT TO IMAGE FOR SINGLE IMAGE & MULTIPLE IMAGE➔

14. INTERPOLATING BETWEEN POINTS IN THE LATENT SPACE➔

Interpolating between points in the latent space is a common technique in generative models, especially in the context of variational autoencoders (VAEs) and generative adversarial networks (GANs). The latent space is a lower-dimensional representation of the input data that captures meaningful features. By interpolating between points in this space, you can generate new samples that smoothly transition from one point to another.

Here's a general outline of the process:

1. **Latent Space:**
   - Train a generative model (such as a VAE or GAN) to learn a meaningful latent space representation of your data. This is often done by encoding real data into the latent space during training.
2. **Select Points:**
   - Choose two or more points in the latent space. These points could correspond to encoded representations of real data samples or randomly chosen points within the bounds of the learned latent space.
3. **Interpolation:**
   - Generate points along the linear interpolation path between the selected points in the latent space. Linear interpolation is a simple method where you generate points that evenly divide the line segment connecting the chosen points.
4. **Decode:**
   - Decode the interpolated points back into the data space using the decoder part of your generative model.
5. **Visualize:**
   - Visualize the generated samples at each interpolated point. This could be done for images, text, or any other data type your model is trained on.

15 . FINALLY WE CREATING VIDEO FROM TEXT PROMPT BY ADDING ALL THE PREVIOUS GENERATED IMAGE USING INTERPOLATIONS➔