

Amazon AWS

SageMaker: Creating datasets, data pre-process, model training, tuning, inference, post-process, deploy.

SageMaker Clarify: Detect Bias, learn/understand model prediction.

Amazon SageMaker Pipelines: Automated ML workflow.

SageMaker Studio: Uses jupyter notebook.

Real-Time inference

Serverless inference

In/Out Module, Network architecture, loss

Things to do -> Deploy style transfer model to amazon sagemaker.

Diffusion Model (Likelihood models)- μ/σ

-> Unlike other generative models (VAE, GAN), takes an input of an original image adding the noise till the point it is independent to an original image(noise) .

First, adding the slight Gaussian noise, then training by reversing back from noise to data(denoising), removing Gaussian noise.

Why use Diffusion model -> for the variety (Unlike GAN where it produces good quality image samples, but not being able to generate various out)

Forward Process: $q \rightarrow$ where the noise is gradually added. $p \rightarrow$ where the noise is removed. The network is training p watching q and removing the noise.

p learns the denoising process from q : two distribution's condition and target variable are opposite to each other. Minimize the p 's distribution of X_t and $X_0 \rightarrow q$ step processed X_t 's distribution KL divergence. Latent's $P_{\theta}(X_t)$ get close to $Q(X_t|X_0)$, in the forward process the variances B_t is learnable by reparameterization, but fixed to constant. Since $q(X_t|X_0) \sim N(0, I)$, therefore, unlike VAE, when obtaining noise X_t , it

does not have to follow particular distribution, hence sampling is from Gaussian distribution. Reverse process and diffusion process's KL-Divergence is low. Training technique: matching terms for condition and target variable. When declaring the model with neural net probabilistic model: $\text{Mean}(\theta)\text{Std}(\theta)$ is inference from the output.

TF Lite

- Light version of TensorFlow
 - For mobile devices/IOT
 - Cloud Service

.Uses Markov chain

Denoising Diffusion Probabilistic models(AI colab notebook for img generation)

:The model is trained to reverse a gradual noising process, making the model possible to generate samples from the learned data distributions starting from an arbitrary noise.

Stylizing -> Surface Layer -> Texture, line, Deep Layer -> Face, eyes (higher-level features).

Deep Dream -> Utilizing Maximization by Optimization target using layer instead of feature map->



CLIP+ Diffusion Model

Gan -> Implicit Models (Harder to evaluate)

= random noise input -> generate something that is trained for (ie: images of cat, dog)

-> Requires lot of datasets and tuning

Auto-encoders -> Map input to low dimensions (features -> low dim latent space) using encoder, then decoder tries to rebuild the original input from a compressed latent space.

** -> X Generative since it only reconstructs. Loss -> Reconstruction loss

VAE (Likelihood (explicit) models)

-> Auto Encoder with generation, learns a continuous latent representation. The encoder

-> obtain mean and standard deviation of an input. Unlike the compressing like

auto-encoder, VAE learns the characteristics of an input (mean, standard deviation),

then models a probability distribution, learning likelihood of an attribute (ie, eye size,),

then generating image by altering the attributes to generate a new image -> shifting the average and scaling standard deviation with random constant value from a normal

distribution. (differ from the original)

*(EfficientNetV2)> Things to look for

Most models: When inputting image, extracting the features with CNN, with pre-determined # of classes, classify with index.

CLIP: Image + NL -> Multi-modal

Contrastive Language-Image Pretraining (CLIP)

- Consider it as perceptor
- Does not require label, pairing the images and texts
- Has image encoder and Text encoder
- -> Likelihood of images match the texts
- Joint representation in between text, images. (image, Caption) sets.
- With that, could calculate the similarities
 - Could utilize Text as guidance,
- Scoring System
- Not a generative model, represent text, images
- Zero-Shot Learning (Positive/Negative)

- W/o trained datasets, flexible pattern recognition, progressed from transfer learning
-
- Transformer model(networks):
 - Does not require labels-> learn with patterns.
 - Encoder/Decoder with the positional encoders tagging data in and out of the network. -> Attention units follows tags than compute the relations
 - Input of image or texts then it gives embedding. Could be used to identify or generates the image

Embedding: Extracting features from an input, converting into a vector.

Image Generation: Visual auto-regressive models: PixelCNN, PixelRNN, Image Transformer, diffusion model, DDPM/Guided Diffusion, synthesized in a Pixel-by-Pixel. This becomes a problem due to the inference and optimization that requires high-computational costs. Therefore, they combine the strengths of different methods. Two-stage generation: DALL-E, NUWA, VQ-Diffusion, and Latent Diffusion learn an encoder-decoder architecture, like VQ-GAN/VQ-VAE, compress image to latent representation/reconstruct it back to image, in second stage, AR based model like DALL-E and NUWA sequentially predict image token depends on the condition, while diffusion model like VQ-Diffusion, and Latent diffusion predict it w/ a gradual denoising process.

CLIP + Image Generator(VQ-GAN, Guided-Diffusion, Diffusion)

Cnn-> inductive bias where the model has local bias.

Transformer -> less than cnn

- Hence, more datas to find the patterns and train -> Leads to more robust

VQ-VAE -> Discrete latent representations

Prior -> probability distribution representing knowledge of data before observation

Posterior -> Conditional probability distribution representing what parameters are likely after observing

*Discrete -> one particular/not continuous/

- Encoder output goes to
- Model presumes posterior and prior follows categorical distribution
- Make an embedding space as table, and also train embedding space
- Discrete codebook(VQ-VAE)
- Codebook -> List of vectors associated w/ index (to quantize the latent representation)
- Output from encoder -> compared to vectors in codebook, vector closest in euclidean distance -> decoder
- Vector Quantization used
 - Could interpret as dictionary, latent space -> discrete representation
 - Continuous feature with pre-determined amount map to trainable dictionary feature
 - Mapping the vectors/Classification
 - Why?
 - Discrete Representations: more natural fit for many domains
 - Good gen and latent features
 - Prevent posterior collapse (learned latent space becomes useless)
- Codebook vectors -> learned with gradient descent
- Losses -> Reconstruction loss, Codebook alignment loss, stop gradient
- Encoder/Decoder shared the codebook

VQ-GAN: Taming Transformers for High-Res Image Synthesis

=> Starts with the idea that transformers learn convolutional structures -> is it necessary to re-learn everything we know about the local structure/learn everything we know in regards to the local structure/regularity of images from ground each time we train a

vision model, or could we encode inductive image biases while retaining the flexibility of transformers.

:Convolutional approach to learn a codebook of context-rich visual parts and learn a model of their global compositions.

:Long-range interactions within these compositions -> require an expressive transform architecture to model distribution over their constituent visual parts.

:Adversarial approach -> dictionary of local parts captures important local structure to alleviate the need for modeling low-level statistics with the transformer architecture. Allowing transformers to concentrate on their unique strength.

-> Merging convolutional and transformer architectures to model the compositional nature of our visual world

-> VQ-VAE based (VAE -> generation segment takes some representation of a probability distribution, -> latent space, GAN -> random noise)

- CNN with Transformers

:Training VQGAN -> Image -> Encode -> obtain latent representation -> latent vectors to codebook Vector -> Discrete Representation -> feed decoder to reconstruct the image
Two Stages

1-> learn to train VQ VAE model, create the latent vectors

2-> Freeze encoder, decoder, code book and train the transformer across the discrete latent space

Uses patch gan, Perceptual loss + Ganloss

Train the VQ-VAE model, Instead of pixel-CNN, uses Transformers(GPT2 model for this case)

-> Similar to VQ-VAE, they swapped pixel cnn and they are not sequence modeling, they are using reconstruction loss: perceptual loss and adversarial loss.

- CNN Encoder
- CNN Decoder
- Quantization -> Instead of just latent spaces, use quantizer

Neural Style Transfer: A Critical Review

- Neural Style Transfer(NST)
 - Transform scenes, change, and modify the environment of a media using Neural Network.
 - Image Stylization based on a general model
- MS-COCO/Cityscape (X benchmark d-set for NST)
 - Two datasets that are mostly used in the experiments w/ in the papers reviewed
 - Used for object detection and recognition.
 - Also, be used as content photos for training various models(330K/25Kin size)
- Danbooru, Safebooru, Videovo.net, or generated according to requirements of the problem statement(1~2k images) -Style Datasets-
- GANs
 - Utilizes two networks, Generators, and Discriminators (Both FC-Layers)
 - Commonly uses uniform/Gaussian distribution to sample the z
 - Generate data based on previous learned patterns/regularities as the model finds these patterns
 - Generative Models -> Gets the details
 - Discriminative Model -> Calculates the likelihood of an image comes from training examples instead of the Generative Model
- Style Based Generator Adversarial Network

(Instead of generating the image from z , after going through mapping network, convert into the intermediate vector W , then generate the image)

 - Characteristics

- Unlike other GANs, since it is utilizing intermediate vector W , it is not limited to the fixed input distribution. When generating image from an input vector Z , it is hard to modify the visual attribute with an input vector due to its limitation of following input distribution, however, mapping network allows to break this limitation, and intermediate vector W does not have to follow the fixed distribution, allowing much versatile mapping(training data), and use the intermediate vector W to control visual attribute. ->

Disentanglement

- Used Progressive Gan setup, reconstructed generator architecture
 - PCGAN ->Characteristics:
 - Does not construct the full network at once, during the training, adding layers to generate higher resolutions
 - Progressively expanding the networks
- Traditionally-> Gen is provided with latent through the input of the first layer of the feed-forward net
 - New: At the same time, omitted altogether and started with a learned constant (Nonlinear mapping-> mapping Z)
- provided latent code z in the nonlinear mapping, and latent input space Z , $f: Z \rightarrow W$ generates w in W
- After mapping, learned similar transformations specializes w to styles: $y = (y_s, y_b)$,
- Normalization used (Admin) : distance, mean, and standard deviation of x_i then scale by multiplied it by $y_{s,i}$ and bias it by $y_{b,i}$
 - $W \rightarrow$ Uses Adain to stylize, adjust shape using Affine Transformation
 - After layers, the training gets unstabilized, so adding normalization to fix this and AdaIN does this
- AdaIN (Style Modules)
 - Layer to apply style

- Instance Normalization, normalize one image, Batch size = N, after normalization, add additional style information and modify features statistics to apply statistics: Adaptive Instance Normalization
- The generator is given direct noise input. Noise input is uncorrelated noise input generated by a single channel of images. Input noise is given to each layer of the synthesis layer -> Stochastic Variation
 - Hair, mustache, etc ~> Stochastic
- Noise image is transmitted on all feature maps, then the corresponding convolutional layer output is applied

FID: Fréchet inception distance (FID) is a metric used to assess the quality of images created by a generative model, like a generative adversarial network (GAN).

Conv-layer -> (Filters), (Linear/Matrix, but goes through activation function which is non-linear), parameters to learn (weights which you update), pooling -> just applying some function

FC-Layer(Dense) -> Linear Operation on layer's input vector

- Deep Convolutional Generative Adversarial Networks(DCGAN)
 - Concepts: With original GAN, utilize CNN to generate higher image generating performance.
 - Guidelines
 - Removing pooling layers, Uses strided convolution in Discriminator and Fractionally Strided Convolution(Transpose Convolution) in Generator.
 - Batch Normalization Layer
 - When data passes from the previous layer, keep the data in batch, calculate the mean and distribution, with that data using normalization equations to normalize the data.
 - Training Rates-> Hyper-parameters impacts lower-> stable training

- Somewhat prevents overfitting, but cannot replace drop-out, better to use it together
 - Higher performance on almost every model
 - Remove Dense Hidden Layers(FC-Layer)
 - Use ReLU on all Layers, for final layer: use Tanh
 - Discriminator -> All Layer using LeakyReLU
- Cycle Consistent Adversarial Networks (CycleGAN)

Pix2Pix: Change one pixel to different pixels. (Black -> Color, label->image, edge -> output)

:Could be somewhat similar to CGAN, but instead of conditioned vector and pair of image, learns from Conditioned picture and image pair to learn

- Image translation on unpaired data
 - Unlike other generative models, train unpaired datasets.
 - Uses Cycle Consistency Loss Function
 - 4 Networks, $G(X) = Y$, $F(Y) = X$, (Restoring $Y \rightarrow X$)
 - Generators use Resnet based architectures and a few encoder-decoder layers
 - $X \rightarrow Y$: Forward Consistency, $Y \rightarrow X$: Backward Consistency
 - Unlike StyleTransfer, learns multiple pictures, cumulatively training.
 - IE) Not just one art piece of Monet, learns Monet's overall artistic style
 - Good for changing picture's night/day, colors, and textures, not performing well when it comes to changing the look(dog->cat)
 - Generator model focus on changing the atmosphere
- Conditional Adversarial Networks for Style Transfer
- Image Style Transfer Using CNN
 - Reconstruct image from the feature map
 - Use VGG19(specific conv network designed for classification and localization)
 - Style Transfer
 - To measure the style image-> choose arbitrary layer L

- Correlation between activations across channels
 - What does each channel has correlations
- Goal: When given Input -> Content image, Style Image, maintain the look of the content image, applying the style to it.
 - High-level feature -> more abstract
 - Use one high-level feature from the content image and every level of style image using feature correlation to define a loss function.
 - Simple and complex texture to make the consistent texture
- Content Transfer
 - Image-Image's content difference: measure with content loss: $L_{content}$
 - High-level layer- a lot of abstract information, compare with feature-map
 - L'th layers content loss -> feature features difference Frobenius norm

Gram Matrix -> Result: for specific layer l , store the correlation in between the feature map and channel and find the important correlation. Ultimately, by using it, minimize the loss in between gram matrix, reduce the distribution difference, higher the correlation and find the similar style


- Style Transfer
 - Style loss, L_{style} : each feature map's Gram matrix, define difference in between Gram matrix with Frobenius norm
 - Gram matrix -> used to quantify image to image's similarities
 - Low -> High layer, measure loss and weights sum to define L_{style}
- $L_{tot} = \alpha L_c + \beta L_{style}$
- Alpha = Content Cost, Beta = Style Cost
 - When layer is too shallow -> pretty identical to content image
 - When layer is too deep -> image will be way too differ from content
 - When alpha -> 1, beta -> 0, image generated: similar to content's outline

- Lcontent -> Extract with High-level layer and using features from low-level layer.
 - When $\alpha \rightarrow 0$, $\beta \rightarrow 1$ image generated : similar to style image
- Image representations-> Derived from CNN for object recognition(High-level Image details)
- Using pre-trained network (ie:image-net), extract feature maps from Content image and Style image
- Goal: Result image's feature map is similar to the feature map of content image, and Style image: similar to the style, the output image's pixel is optimized.
- When β goes up, the output of an image fails to remain in its outline, gets closer to style image's texture
- Towards The Automatic Anime Characters Creation
 - When generating faces -> they get distorted on some features/get blurred
 - 3 Contributions
 - DRAGAN (Deep Regret Analytic GAN), with WGAN(Wasserstein Distance GAN), normalize and trimmed gradient penalty, created gradient penalty schemes, improve mode collapse(Can only produce a single type of output or small set of outputs, generated samples are very similar)
 - Least computation cost
 - Gen->16 ResBlocks, 3 feature upscaling blocks
 - Disc-> 11 ResBlocks, a dense layer as an attribute classifier
 - Clean anime facial d-set collected from Getchu
 - Train GAN from untagged images
 - Tags -> assigned using Illustration2Vec
 - Drawback
 - Can not handle super-resolution
- CartoonGAN
 - Real-word Scene to cartoon-style
 - Unpaired images for training

- Generator -> one flat Convolution block
 - Proceeded by two down-Convolution
 - Blocks -> compression/encoding of an input img
 - Content/Manifold part : eight residual block
 - Two up-conv blocks and conv layer -> create cartoon style output
- Discriminator
 - Flat layers
 - Preceeded by two strided Conv. blocks
 - Reduce resolution and encode features
 - Final Layers
 - Feature construction block w/ conv layers to obtain a classification
- Loss Func
 - Adversarial loss and content loss
 - W : weight that limited to the content retention amt from the input
- **Artsy-Gan**
 - Tries to fix current issue with style transfer such as CycleGan(Slow training, source of randomness limited to input images)
 - Three Ways to fix the problems
 - Perceptual loss(Define Feature reconstruction loss, style reconstruction loss, maintain image style and context) defined img instead of reconstructing to improve speed and quality
 - Chroma sub-sampling to process the images: improves inference/prediction speed and makes the model compact by size reduction
 - More diversity by adding noise to the Gen input, pair with the loss function to create a variety of details for the same image
 - Inputs
 - 3 Channel color image w/ noise added to each channel
 - Gen:
 - Three branches

- Same input but produces different output img channels that are converted back into RGB by a model at the end of the network
 - Disc:
 - Same as Cycle Gan, using 70x70 PatchGANs(Only penalizes structure at the scale of local image patches, using small patch sizes, computing with sliding window, less parameter, faster computation)
 - Loss fnc
 - Loss Gan + aLoss perceptual + bLoss diversity
 - Alpha, beta = control the significance of losses
 - Loss Lgan -> equalizing distribution of domains
 - $G(x,z)$ -> produced imaged from gen
 - Ldiversity
 - For improving diversity
 - Lperceptual
 - Overcome unconstrained problem by keeping the object and content in the output
- Depth Aware Style Transfer
 - Tries to fix the depth of the content picture has not been reproduced
 - Technique -> depth-aware AdaIN (DA-AdaIN),
 - Works with varied strength
 - Closer areas -> less stylized,
 - Distant(representing background) -> more stylistic feature
- Deep Photo Style Transfer
 - When style transfer, there are distortions/ spill over
 - Sol: Use semantic segmentation
 - Problem may raise when use Gram matrix, if the reference img and content img's number of elements are different.
 - Semantic segmentation: use common label(ie;river/water/fond) as merged

- Photorealism regularization: regulates the input image transformation so it could maintain the photorealistic traits.
- Loss-> L_m = penalize when the image has distortion. matting-> detach foreground and background.
- Perceptual Factor Control in NST
 - Extension to the existing methods -> spatial color, and scale control
 - Breaking down into these features
 - Identification of perceptual factors -> key to producing higher-quality images.
 - Spatial control -> Control which region of the style image is applied to each region of the content image
 - Methods: Guidance-based Gram Matrices
 - Each Image is provided with a spatial Guidance channel, indicating which area should be applied to what style
 - Spatially Guided Feature Map -> R regions, L layers
 - Stacking the Guidance matrices with the feature maps directly
 - More efficient than the Guidance-based Gram Matrices
 - Downside -> texture quality
 - Color Control
 - Luminance-only Transfer
 - Only transfer preserves the content colors, losing dependencies between luminance and colors
 - Color Histogram Matching
- Stability Improvements in Neural Style Transfer
 - Image Style Transfer,
 - Groups:
 - Optimization approach that solves particular optimization problem for the generated image(matching CNN output's statistic's features)
 - Results: Good, take time to develop

- Feed-Forward approaches
 - Results: Shorter time (Could be used for real-time synthesis), unstable readings
 - New Methods
 - Stabilizing Feed-Forward Style Transfer for video stylization
 - Using Recurrent network trained using temporal consistency loss
 - Using three loss: Content Style loss, Style Reconstruction loss, and Temporal Consistency Loss
 - Preserving Color in Neural Artistic Style Transfer
 - Color Histogram Matching
 - $S(\text{Style img}), C(\text{Input Image})$
 - S's Colors -> coordinate the C's color, producing S'(new style image that replaces S as an input to the NST algo)
 - Pixel : $Xs' \leftarrow AXs + b$, A: 3*3 matrix, B:3D vector $x_i = (R,G,B)^T$
 - Transformation -> chosen with mean and covariance of the RGB val in S' matches the content image (C)
 - A, b val in formula, conditioned mentioned above with C
- 
- - Solution for A, that matches condition
 - Cholesky decomposition
 - Luminance(Brightness)-only Style Transfer
 - Visual perception -> Influenced more with changes in luminance than color
 - Luminance channels L_s, L_c derived from style, content imgs. NST applied to them, results luminance image L_t

- Use YIQ color space, input pic color info merged with Lt to generate the resulting image

- GauGAN:
 - Conditional picture synthesis -> Create photo-realistic pictures molding on some input data
 - Original methods, -> Segmantic layout (directly) -> convolutional, standardization-> Activation(nonlinearity layers)
 - Problems:Wash away information in information semantic covers
 - Solution:Spatially adaptable standardization
 - Semantic input formats -> spatially flexible, (learned change and can reasonably multiply the semantic information all through the networks
 - Spade Gen(Spatially-Adapative Denormalization)
 - Trained module parameter has enough label layout information encoded, no need to input segmentation map in first layer at generator
 - Delete generator's encoder part
 - Could take random vector as an input for generator

Deep Learning Building Blocks

- **Connectivity Patterns**
 - Fully-Connected
 - Convolutional
 - Dilated
 - Recurrent
 - Skip / Residual
 - Random
- **Nonlinearity modules**
 - ReLU
 - Sigmoid
 - Tanh
 - GRU
 - LSTM
- **Loss**
 - Cross Entropy
 - Adversarial
 - Variational
 - Maximum Likelihood
 - L1 and L2
- **Optimizer**
 - SGD
 - Momentum
 - RMSProp
 - Adagrad
 - Adam
 - Second Order
- **Hyper Parameters**
 - Learning rate
 - Weight decay
 - Layer size
 - Batch size
 - Dropout rate
 - Weight initialization
 - Data augmentation
 - Gradient clipping
 - Momentum

Conversation Log(Usama):

Define,(Write the Paper)

Differences/Similarities

Encoder with Clip+Diffusion Model /VAE,

, What he left off with the code, with the model

String(Csv) Taking nouns (6000) +4000 more

Combination, (Color)

Ranking(Random Noun) -> 1 Was not good results

Style Galleries->

1 Model for image generation

1 For style Galleries (applied into image)

256 (Size), 250(steps),

256(size)_250(epoch)

Manual Project(Defining on top of that)

1024 size(breaking down the layers)

Original + Style on top.

Oracle

Claudia is doing experimentation -> Best Quality one to stylized

FastStyle -> Style Transfer

256*256(Less Time)

1024*1024 Image

Upscale

-> Smoothing out

1 Super Resolution ->

0.4 Repos ->

Images_generated (Linked it into Colab)

Image Align

Iterations -> Steps for styles (300~500 textures and colors-> Take a long Time),

Paths are set Hard-Coded

0.4 Repos

Art Generators (Gradient-> Clip_guidance_scale)

Play with the steps

Clip + Diffusion (6~7 Different Libraries/ Open sources) -> Free Library

Model -> Different Architecture

forward-> visualizing (Clip + diffusion)

Clip -> Input: Texts , Output: 512 *1 Vector Dimensions of vector/(Single Dimensional Vector with 512

Features, Texts with features, Multimodal Vector-> Space in between texts and images, the vector could be passed onto generative model to generate the image). CLIP -> Target model should look like, and passes -> Generative Model to create the Model.

Methods to reach (could be different/ diffusion , Gan and anything).

Ver 0.3 Repo

Clip -> Encode the Image (Texts to the image), Image of a dog and string of a cat, generates somewhat in between dog/cat.

Embedding into Diffusion Model -> Something from the starting Image(Prompts,)

Clipembed(512), Image(Image)---> Art Generator

Clip, generative Model ,

Diffusion Model 3Inputs ->Clip output, random noise, timestamps(time series 0~

Markov -> 10~20 Steps , removing some, noises, Scratching of the noises, (noise reduction model)

Output from the clip,

Then give Style.

VGG-19(Visual

Geometry Group-19) NN model

Neural Style -> VGG19s(Conv NN that is 19 layers deep,designed for classification and localization)

(Different blocks), blocks(like resnet),

4, 12, 24~ Generated Images -> out of that 50, outputs as image outputs

Layers of residual models for style/~ Stylized Image (Averaging, Maxing, adding, Layers)~ VGG

Overlaying the images

Style -> Style

Content_layers -> Diffusion Model

Scores for the different Images,

Content, Style Image, output Image

Predefined styles (6Images),

Image , Style Images and Join them together

Models -> VGG (That can indefinitely),

Overlayers ->

Limited Capacity, (Open Sources)

Artistic Style Transfer with Deep learning -> Only Single Style,

VGG19,

Art Generator-> (Image Generations)

Output of Clip,

CLip -> Text Transformer, (Embedding)

Diffusion -> Sequential (Parameters and blocks)

Set # of steps, and #time,

Image -> 10 different layers, in every run , 10,20,different steps ,

Next Epochs -> Feedback, back to step 1 ~ denoising,

Denoise to a certain extent,

Generating new images

Sequential learn to denoise to generate the image,

Random noise

The model knows at X0 the image

Vision Transformer -> Images,

Dalle2

->Not Open Source

Dalle1

->Open Source -> Modification

ruDalle

->Generated from the texts

Google Image net

3.2 -> 2 Gpu

3.4 -> 4 Gpu

Oracle Cloud to Generate the Image

Vdiffusion (500words)

Splits into in progress

Genertaed

(Text Strings to done)

Splits -> Inprogress, -> Generated

All the Logs are screwed, because(Run Time)

Noise Model, (ddpm~etc)

Clip Sample, Moving them out

New Launcher -> Environment (2things)

Datascience -> Projects -> create the notebooks(google collabs instances)

Pressing delete button (Gan-Model)

Oracle Cloud / Refine them,

0.4

CLip.py

-> Interpolations: Expand into 64(upsampling)

"Rn" -> Residual ->Image Parts

"Vit" ->Visual transformation-> Texts, (Base 32), Models: visual model

Downloading the different model : def _ download

Def Load: clip trained on the model, resnet: separate model/ download the pretain model

Available from clip

model.apply -> encode image

patch_image-> encode text /encode image

->Transformers, texts and images

Model.py

Attentionpool2d -> parent blocks that are called

->Model architectures:

Modifiedresnet:

Layers -> mresnet

Activation layers

ResidualAttentionBlock:

Multi-headattention block: particularly giving attention to specific location,

Attention-weights: softmax applied:attention to the specific location, focus

Precomputed factors/colors or features/

multi -head:

Attention is very small part of the images/

:Learning specific details (locations)

Class Transformer(nn,module)

-> Text:Encode

Image and texts: assign function for positional_ embedding

2d to 1d

Def stye: calling from clip.py

Def forward

Cosine -> cost function (weight)

Repos 0.2

clip

Clip.py

Resnet take one model

Output of clip is embedding 512

Representation of an image

1Dimensional of target

Diffuson

different model(pretty much same structures)

Pretty similar

trained

Repos 0.2

Sampling ->

Def sample(model-> models (diffiusion), x->noise, steps-> howmany

x-> image we got out of rom the clip

SNR -> Signal to Noise Ratio

Ddpm -> one noise function

Cfg_sample.py

-> diffusion configuration

Vist or Resnet

Clip output -> Diffusion Model

Cc12m_1 : diffusion model

Clip:

Joint space between image and texts

Clip -> Already Pretrained ,

Diffusion -> ALready pretrained

Sequential,

Clip -> Input of image or texts then it gives embedding. Could be used to identify or generates the image

Clip

-> One dog -> Visual transformer or resnet-> pick top 9

Images, labels, custom models

-> Since we are using vague nouns

Rudalle.ru

Moon Knight :

Search Moon / Search Knight

: pick top 5

Image and style

Bridge in between texts and images

CLIP: Image + texts-> End to End

Reading over the articles related to the CLIP/Diffusion Parts,

Differentiate the part in between the Clip anrd other VAE,

Diffusion Model -> Study

Style Transfer

-> Trying to do a different style Model

Shapes do not work well

VGT-19

Style Image

Image Generation

-> Trying to do more

Usama, Image, and Style (pair)

Stochastic(Randomly Determined)