Chapter-09 Generative Adversarial Networks & Applications of Deep learning

9.1 What is a Variational Autoencoder:-

Variational Autoencoders also generate a latent representation & then use this representation to generate new samples (i.e images)

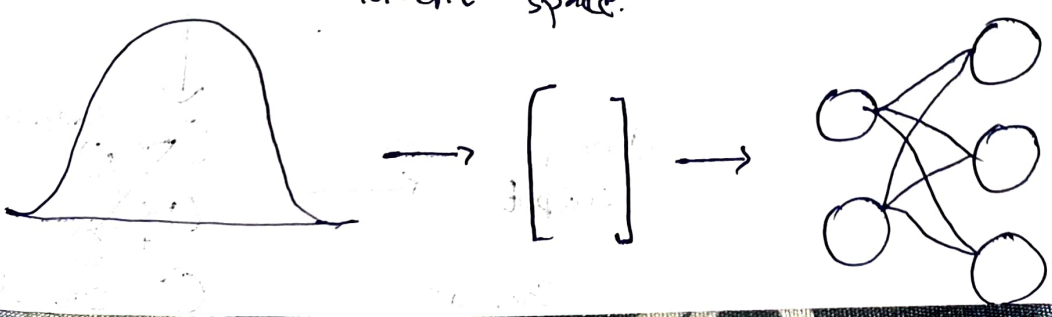
Important features of variational autoencoders:-

- Data are assumed to be represented by a set of normally distributed latent factors
- The encoder generates parameters of these distributions namely μ & σ
- Images can be generated by sampling from these distributions

Goal of VAE :- generate image using the decoder

- latent vector: Each element drawn from a normal distribution
- Parameters learned by the decoder

Secondary goal:- have similar images be close together in latent space.



Working of VAE:

Variational autoencoders assume that the latent distributions is normally distributed, then learn to generate images from this distribution.

Step 1 :- Image fed through encoder network

Step 2 : Learning the parameters from normal dist. (μ, σ)

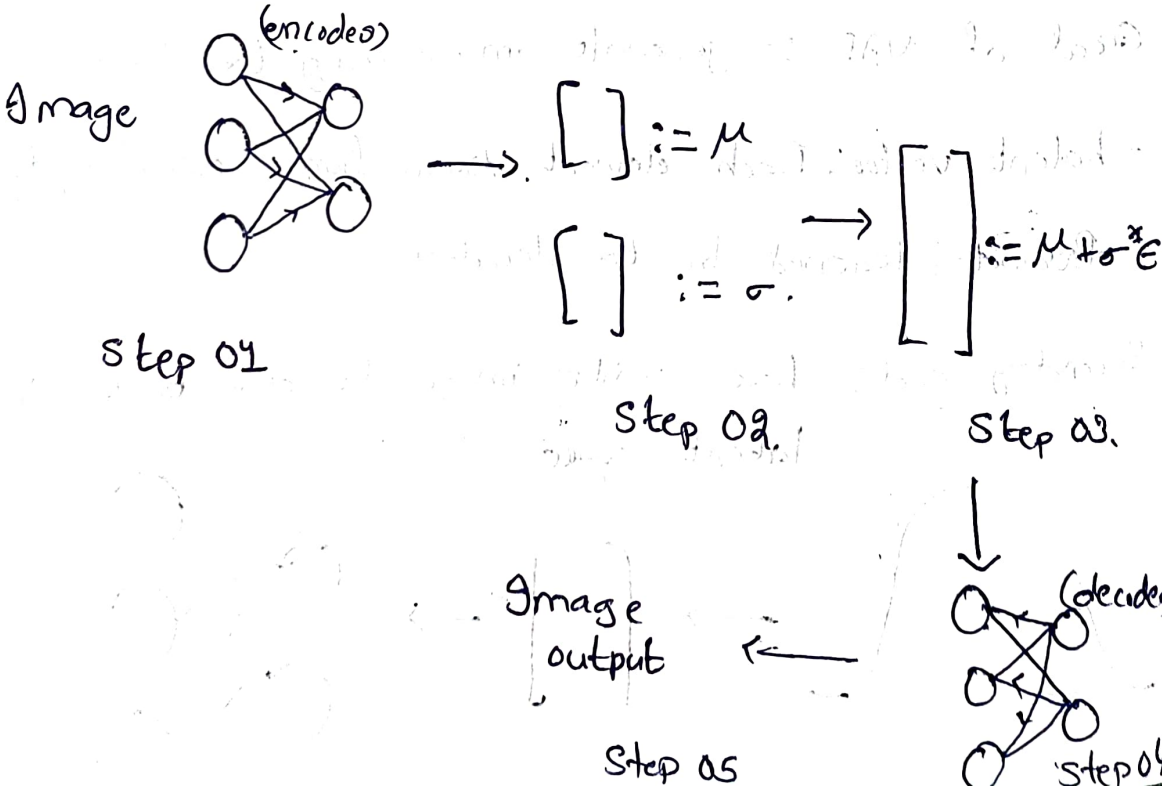
Step 3 : These are combined into one vector.

- Random Noise $\epsilon \sim N(0,1)$ added.

combined vector = $\mu + \sigma * \epsilon$

Step 4 :- This vector is fed through the decoder.

Step 5 : The reconstructed image is generated using the decoder network.



Loss Function of VAE:-

We want to VAE to learn to reconstruct the original image.

- The VAE should do so from the space of vector drown from a standard normal distribution

The 2 components of the loss will be:

- A penalty for not reconstructing the image correctly
- A penalty for generating vectors of parameters μ & σ that are different than 0 & 1 respectively, (the parameters of standard normal distribution)

VAE have a loss function with 2 components:-

1) The pixel-wise different between the reconstructed image & the original image. Many functions (like MSE, etc) can be used

2) The difference between the vectors produced by the encoder & the parameters of the standard normal distribution.

* Specific loss used for this part of VAE is the "KL Divergence" between the generated data & the normal distribution

→

$$\mu = \begin{bmatrix} 0.7 \\ -0.6 \end{bmatrix} \longleftrightarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

→ KL Divergence used to compare these vectors

→

σ vector interpreted as $\log(\sigma)$ since variance can't be negative $\left\{ \log(\sigma) = \begin{bmatrix} 1.0 \\ 0.6 \end{bmatrix} \longleftrightarrow \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right\}$

$\log(\sigma)$ compared to 0 since $\log(1) = 0$.

→ KL Divergence formula for Normal distribution :-

$$KL = \frac{1}{2} \times \left(\underbrace{e^{\log(\sigma)} - (\log(\sigma) + 1)}_{\text{penalises loss of } \log(\sigma)} + \underbrace{(\mu^2)}_{\text{penalises loss of } \mu} \right)$$

This penalizes $\log(\sigma)$ & μ respectively for being different from zero (0)

* Loss for $\log(\sigma)$ use the fact that :- $e^x + (x+1)$ is minimised at $x = 0$

Note on KL Diversion :-

- It is not technically necessary to include it in VAE loss function
- It helps generate a latent space where visually similar images are close in the latent space

VAE : Results

On the right:

- VAE trained with 2 dimensions in the μ & σ vectors
- As the 2 dimensions of the μ vector vary from -15 to 15, we can plot the digits the trained decoder generates
- Close numbers in the latent space generates similar images

9.3 Generative Adversarial Networks (GAN)

* Unsupervised learning & Powerful Generative Model.

Review: Adversarial Examples:

Neural network's differentiability makes them vulnerable to adversarial examples:-

Image → Forward passing NN. → L = Loss

Adversarial examples: generated by repeatedly adding a function of $\frac{\partial L}{\partial X}$ to the input image

Image → Backward passing NN → L = Loss

$\frac{\partial L}{\partial X}$

→ Take a training set & the focus on adjusting our original images in that backward pass in relation to each one of our gradients.

GAN's Origin Story:

The invention of GAN was connected to neural network's vulnerability to adversarial examples

- Researches were going to own a speech synthesis content, to see which NN could generate the most realistic speech
- A neural network - "the discriminator" - would judge whether the speech was real or not
- They decided not to own the contest, bcz they realised people would generate speech to fool this particular network, rather than generating realistic speech
- The researchers realized they could solve this by having the discriminator continually improve at distinguish between real & fake speech
- They could do this by feeding it real speech beside fake speed.
- By feeding the gradient of resulting discriminator with respect to the input $\frac{\partial L}{\partial x}$, back to a NN generating the speech, they realized they could train a network to generate a very realistic speech.
- As both networks try to beat the other, they both continously improve.

GAN Defination:

GANs are a way of training a NN simultaneously

- One of the neural networks - "the generator" - learns to map random noise to images indistinguishable from those in some training set

Generator:

→ Conv1 → conv2 → conv3 → conv4 → G(x)

Discriminator:- → which are fake
→ which are from training set

G(x) → conv4 → conv3 → conv2 → conv1 → D(x)

Image indistinguishable from training set images

9.4 How GANs Work?

Training procedure of GAN:-

Step 1 :- Randomly initialize weights of generator & discriminator networks

Step 2 :- Randomly initialise noise vector & generate image using generator
(Generated img should be the same size of ~~train~~ training set images as it is later fed through the Discriminator network)

Step 3:- Predict probability generated image is real using discriminator.

(Output:- probability image is real)

Step 4:- Compute losses both assuming the image was fake: comparing $p^{x_G}_{real}$ to zero(0), & assuming it was real: comparing $p^{x_G}_{real}$ to 1(one):

Generator $\rightarrow X_G \rightarrow$ Discriminator $\rightarrow p^{x_G}_{real}$

$L_0^{x_G} = f(p^{x_G}_{real}, 0)$ $\quad L_1^{x_G} = f(p^{x_G}_{real}, 1)$

f = loss function.

Step 5:- Use $L_0^{x_G}$ to train the discriminator, we want to train the discriminator to output 0(zero) for generated (fake) images

Discriminator $\leftarrow L_0^{x_G}$

Step 6:- Compute $\frac{\partial L_1^{x_G}}{\partial x_G}$. based on $L_1^{x_G}$, the penalty for the discriminator outputting a probability different than one (1), but ~~not~~ do not use this to train the discriminator.

Step 7:- use $\frac{\partial L_1^{X_G}}{\partial X_G}$ to train the generator, we want it to generate images that the discriminator thinks are real. Here we continue to backpropagate through generator to update its weights.

Generator $\leftarrow \frac{\partial L_1^{X_G}}{\partial X_G}$

Step 8 :- use the discriminator to calculate the probability that a real image is real.

Generator $\rightarrow X_R \rightarrow$ Discriminator $\rightarrow p_{real}^{X_R}$

Step 9 :- Compute $L_1^{X_R}$

Generator $\rightarrow X_R \rightarrow$ Discriminator $\rightarrow p_{real}^{X_R}$

$L_1^{X_R} = f(p_{real}^{X_R}, 1)$

Step 10:- use $L_1^{X_R}$ to train the discriminator by updating its weight

→ Repeat this procedure with new random noise from generator each time. Continue until image from generator looks real.

- values ~~from~~ of losses from discriminator & generator may still be fluctuating when generator is producing realistic images, so can't use these alone to determine when to stop training

- These are ways of quantifying image quality (eg: Inception score)

9.5 Issues with Training GANs

→ Training GANs effectively is highly dependent on both generator & discriminator learning at the same rate

→ Ability of 2 networks to learn is affected by:

- Network architecture
- Loss functions
- Learning rates
- Optimization Techniques

→ GANs are more sensitive than traditional neural networks to choice on these dimensions

What should you do to train a GAN?

compared with building a neural networks for a supervised learning problem such as image classification or text generation, it is more important to read original papers & examine code on GitHub to see how researchers train their GANs.

Famous examples of GAN include:-

- Deepfakes → Replacing your face with any other Celebrity.
- Age Interpolation - changing one's age
- Text to Image
- Lip sync

9.6 Additional Topics in Deep Learning

Hardware for AIs: GPUs

GPUs have become popular DL workhorse

- Feature thousands of small, simple cores specialized for numeric parallel computations
- Many transistors dedicated to computation
- GPUs excel at repeated similar instructions in parallel
- Optimized for parallel data throughout computation
- Major neural networks breakthrough since 2012 have been powered by GPU computations
- Performance has since increased >5x
- Once the data is in GPU memory the bottleneck are small
- NVIDIA® is highly popular in GPUs
- GPUs are great at parallelization.

CPU vs GPU :-

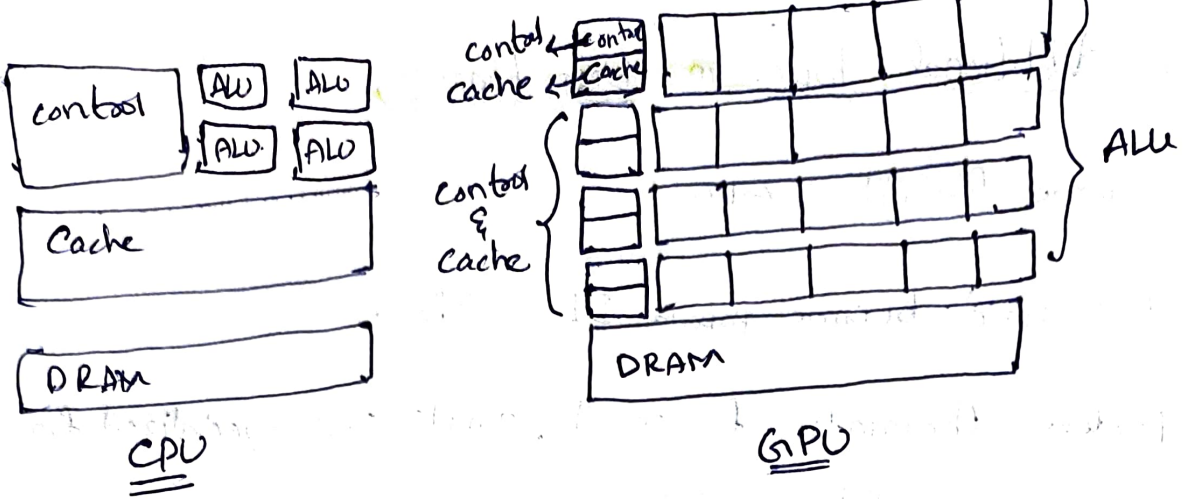
⇒ CPU :- Dozens of cores GPU : Thousands of cores

*ALU :- Arithmetic Logical Units → which performs arithmetic operations

*control :- Decode instructions into commands & calls on the ALU for any operations

*Cache :- serves as high speed memory, where instructions can be copied & retrieved.

*DRAM :- for more, longer Term memory.



* GPU are specialised processors More ALU & seperate control & cache for ALUs
* CPUs however are still main computation engines on most computers

Difference:-

- CPU have dozen of core, GPU have thousands of less powerful cores
- CPU have fewer ALU & lower compute density than GPU
- CPU have lower latency & have larger cache memory, compared to GPU's
- GPU are designed for parallel tasks
- GPU perform well for a single instruction performed over a large amount of data
- GPU have additional overhead when copying data from main memory
- GPU can be better when a large number of memory swaps are needed.
- GPU are poor for task that can't be parallelized & for heavy processing on fewer data streams.
- CPU excels at serial tasks & are easy to program
- Popular programming lang, compiles, to be run on CPU by default.

Locally - Interpretable Model - Agnostic Explanations :-

DL models are difficult to interpret :-

- Many parameters, Complex Network

One approach is to generate Locally Interpretable Model-Agnostic Explanation (LIME):

- LIME treat the model as a black box, & focus instead on the sensitivity of outputs to small changes in input
- Analogous to feature importance, LIME summarize the sensitivity of Regression or Classification outcomes to each variable
- Nonlinears & variable that cannot be perturbed present challenges to approach