Mini Project

Introduction:

In the contemporary world there is a huge need for High Tech based vehicles which includes Automation of Cars. In Automation of cars, Machine Learning plays a vital role for controlling the car functions based on what objects it sees around. The examples could range from stopping at a signal or accelerating on high-speed roads to using different speeds during weathers, traffics etc.

In this Project, we try to use image datasets taken from Cars during different distinct weather conditions and using the power of Latent Space interpolations to generate new images of mixed weather conditions which can serve as a great data for training autonomous cars for such mixed weather conditions.

Dataset:

The dataset consists of 490 Images out of which 245 images are taken cloudy days and the rest 245 images are taken on rainy days. The images are taken at exact same location. So there 245 places where both a rainy and a cloudy image is taken. The Image is in general with pixel size of the of 1024 pixels high and 2048 pixels wide.

The task at hand is to generate new images which are interpolations between these 2 distinct weather conditions

Sample Images:

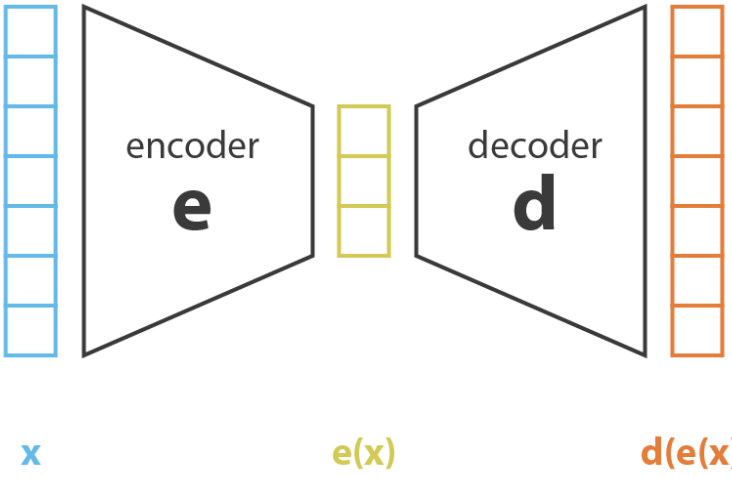
Cloudy

Rainy

Variational Auto Encoder Theory:

In general, the images are of greater sizes, however when we use Neural networks we can try to reduce the image size while retaining most of its features. This type of compression that we use here is called Latent Space Encoding.

The structure of a typical **Autoencoder** is as follows:



- Input data in some dimension 'N' is passed through an encoder function.
- The encoder function/network takes in 'N' and encodes it to a much smaller dimension 'n'
- Later the decoder function/network takes the output of encoder network and re converts it back into 'N' dimension.

So, the accuracy of such a network is based on how well the 'n' dimension input contains the information of the larger 'N' dimensional system.

Latent Space Interpolation:

The 'n' dimension encoded space is known as the Latent Space.

Interpolation refers to interpolating between two given inputs 'X_1' and 'X_2'.

Instead of interpolating right between the direct inputs 'X_1' and 'X_2', we use the latent space encoded 'n' dimensional values for these inputs and interpolate between them.

For inputs say X_1 and X_2 , let the encoder give outputs : $e(X_1)=x_1$ and $e(X_2)=x_2$, where x_1 and x_2 are of 'n' dimensions.

Suppose we need 'k' latent space images then; we pass the following input to decoder based on linear interpolation between x_1 and x_2.

i^{th} image between x_1 and x_2 is given as :

$$x_{1-2,i} = (1 - \frac{i}{k})x_1 + \left(\frac{i}{k}\right)x_2$$

Where the 0^{th} image is x_1 itself and the k_{th} image is x_2 itself.

On Passing through decoder, we get an 'N' dimensional vector of $X_{1\text{-}2,i}$ for i^{th} image between X_1 and X_2 inputs.

These are the typical steps used in a normal AutoEncoder.

A **Variational Autoencoder** can be defined as being an autoencoder while training is regularized to avoid overfitting and ensure that the latent space has good properties that enable decoding in a fruitful way.

Instead of encoding an input as a single point as we were doing in Autoencoders, here we encode it as a distribution over latent space.

So, the structure in Variational AutoEncoder change as:

- Input data in some dimension 'N' is passed through an encoder function.
- The encoder function/network takes in 'N' and encodes it as a distribution over some 'n' dimensional latent space.
- Next a random sample is chosen out of this distribution and is passed into the decoder network.
- The Decoder network re converts it back into 'N' dimension.

Since this is a sample from an encoded distribution for an image rather than a single point, this reduces the risk of overfitting a particular image to a single point.

The Code Explanation:

Required Libraries are imported and Images is read from google drive and reshaped to 256x512 size due to Colab Crash issues on higher dimensions.

Input shape is (490 images, 256, 512, 3), 3 indicates RGB images

Loss Function for VAE:

Loss function involves a binary cross entropy function which computes the loss between the original and produced image and it contains the KL divergence loss which stores the information of how much our Variational distribution differs from the gaussian distribution.

- Variational Encoder consists of 2 parts Encoder and Decoder.
- Initially there are 2 Encoder Layers and a Flattening which is passed to learn mu and sigma.
- Latent space is found out from sampling from mu and sigma.
- Later Decoder is used to retrieve the original dimensions from the latent space.

This whole model is compiled using Adam Optimizer and the loss function as mentioned earlier.

Sample Outputs:

Rainy to Cloudy:

 to 

Cloudy to Rainy:

 to 

References:

- https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73
- https://becominghuman.ai/using-variational-autoencoder-vae-to-generate-new-images-14328877e88d
- https://www.youtube.com/watch?v=rZufA635dq4
- https://towardsdatascience.com/reparameterization-trick-126062cfd3c3
- https://cityscapes-dataset.com/