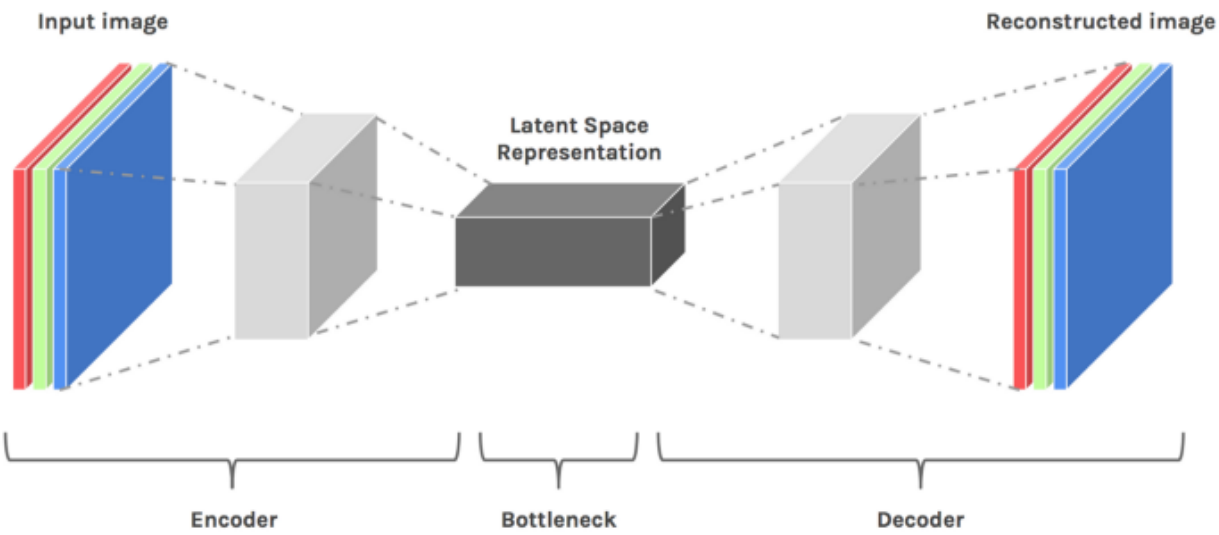
MINI PROJECT

SCENE RECONSTRUCTION IN AUTONOMOUS VEHICLE

Name : Hemanth ,ES1919BTECH11003 | Naveen, CS19BTECH11009



Introduction

Image generation is an active area of research, and the latent space is an essential concept that must be understood. **l**atent space representations are used to transform more complex forms of raw data (i.e. images, video), into simpler representations which are ‘more convenient to process’ and analyze that is we do dimension reduction.Each time the model learns through a data point, the dimensionality of the image is first **reduced** before it is ultimately increased.When the dimensionality is reduced, we consider this a form of lossy compression.Because the model is required to then reconstruct the compressed data (see Decoder), it must learn to store all relevant information and disregard the noise. This is the value of compression- it allows us to get rid of any extraneous information, and only focus on the most important features.**This compressed state is the Latent Space Representation of Data**The model learns the data features and simplifies its representation to make it easier to analyze. This process is termed as **Representation Learning**.

Variational AutoEncoder:

In the last few years, deep learning based generative models have gained more and more interest due to (and implying) some amazing improvements in the field. Relying on huge amount of data, well-designed networks architectures and smart training techniques, deep generative models have shown an incredible ability to produce highly realistic pieces of content of various kind, such as images, texts and sounds. Among these deep generative models, two major families stand out and deserve a special attention: Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs).

VAE is autoencoder whose encoding distrib is regularised while training so that the latent space has good properties(clarity in image) after reconstruction. Variational comes from close relation there b/w regularisation and variational interference method in stat.

VAE is an probabilistic extension of auto encoder. Vae has below extra features

1. Probabilistic encoder ($q_\phi(\mathbf{z}|\mathbf{x})$) and decoder($p_\theta(\mathbf{x}|\mathbf{z})$.
2. Prob disturb of latent space representation.i.e, the bottleneck layer of the auto encoder

A variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting

and ensure that the latent space has good properties that enable generative process.

what we wanna do here is to generate new images using VAE (Variational Autoencode)

Approach and Methodology:

Initial Work:

- We initially played around with MINIST data set and then tried to implement the similar kind of concept to the images given in the data set.
- Raw idea how to get an interpolated image of two numbers(MNIST)
  As usual represent both the numbers in the form of 28*28 matrix and the try to operate(some mathematical operation on each vertex of both the matrix) between these two matrix and obtain a new one. And then try to get the image of this new matrix.

  Step1: We take 2 numbers and generate the encoding by passing them through the encoder. The encoding are flat vector of length+.

  Step 2: generate interpolate b/w these encoding .

  Step3: generate the images through this interpolation by passing through decoders.

The above interpolation is done using keras and tensor.js Following codes for building model,image generator , training the model interpolating the encoding , decoding the interpolation are available on MNIST interpolation.

From this we can obtain the interpolated images of the two numbers (28*28) .

Extension of initial work:

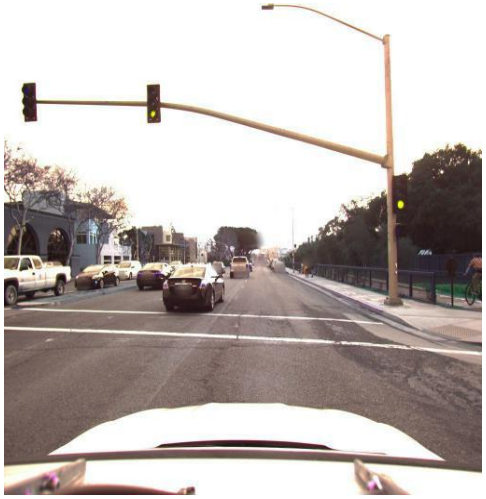
Method1:

- First We change the images in the given dataset into an array of data points and then we encode using Variational Auto Encoders and created the latent space vector interpolation.
- Initially, we have fitted the model with the same input image as the target image which is present in the given data set.
-
- later, we took two images from the data set and tried plotting the interpolated images, and also tried to know how the picture quality varies when we change the dimensions of the image, epochs

- Code for the above attempt is in the zip file named as **Image Transition without Training.ipynb**

- After observing the output for few pairs of images we got to know that in the interpolated resultant image one of the parent image was dominating in all the sequence.

Example:1



Output Image

This method really gave the bad results because we have not trained the model with enough dataset.

Method2:

- Considering all the images which are in dataset , We have taken all the images of the given data set into one folder and then we have trained the model using all these images which is similar to what we have done with the MNIST data set and tried plotting the latent space representation of all the images. We have changed the images in the given dataset into an array of data points and then we have encoded use Variational Autoencoders.

- We also started altering the dimensions from 28x28 to 256x256 and also tried to notice the changes that are being occurred in the output series of transitions.

- But if we are increasing more than 256 the kernel cell is being crashed sometimes and sometimes only a black image is being displayed in place of transition images.
  But after including the learning rate has been resolved also we got a better loss after including this lr.

- Changing the value of epochs so as to see the changes in the picture clarity and distribution.

- Code for the above method is included in the zip file with name **Image Transition with Training.ipynb**

Some Explanation of snippets in code:

The idea is mapping original image to latent space (done by encoder) and reconstructing back values in latent space into its original dimension (done by decoder).

Image dataset preprocessing

After running the code below all images are going to be stored in *X_train* and *X_test*, while the ground truths (or labels) are stored in both *y* arrays.

Now, if you want to view how the image looks, we can display photos from the X train array at indexes.

The first step is to normalise the values that indicate each pixel's brightness so that they fall within the range of 0 to 1 rather than 0 to 512. It's as simple as dividing all elements in the array by 512

Constructing encoder

Now we like to define some variables before building the encoder component of VAE so that we may reuse this architecture for various purposes without having to change many things in the neural net. We'll also describe the input shape for the first layer, which will take its values directly from the picture data shape. In addition, our latent space will have two dimensions, allowing us to depict the image distribution in a normal scatter plot.

Constructing decoder

The decoder works in a similar way to the encoder. Instead of starting with (1024, 1024, 1) as the input shape and producing a value in the shape of (2,), we'll use (2,) as the input shape and produce an image in the shape of (2,).

It's worth noting that the decoder's convolution layer is Conv2DTranspose, which operates by inversely transforming the conventional Conv2D layer.

Connecting the encoder and decoder

Remember that we haven't built both an encoder and a decoder yet. We may achieve this by just passing the input and output layers to Model ().
However, note that the encoder and decoder parts are still not connected. In order to build the whole VAE, we must connect the two. The output of the vae model is the decoder's output, which gets its input from the encoder's output."
By the time we get to this stage, we've already created three models that can be trained simply by using the fit() method on the linking model (vae). The summary() method can be used to see the information of each model.

Results:

Following are the outputs of the final code:

Output Image:

References:

1. https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d
2. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73
3. https://medium.com/@noufalsamsudin/latent-space-interpolation-of-images-using-keras-and-tensorflow-js-7e35bec01c5a
4. https://openaccess.thecvf.com/content_CVPR_2019/papers/Chen_Homomorphic_Latent_Space_Interpolation_for_Unpaired_Image-To-Image_Translation_CVPR_2019_paper.pdf
5.