Sports car image generator project by reyan javed

**Abstract:**

This project presents the development of a variational autoencoder (VAE) model designed to generate images of sports cars from a given dataset. The VAE architecture comprises encoder and decoder networks, enabling the model to learn condensed representations of input images in a latent space. By training on a curated dataset, the VAE learns to capture essential features and characteristics of sports cars. Evaluation metrics such as image quality, diversity, and similarity to the training dataset demonstrate the model's effectiveness. The generated images should ideally closely resemble sports cars, capturing details and variations present in the training data.

**Introduction:**

­This report will go through the process of developing and testing of the application as well as going through any changes or issues that were made throughout development.

I had chosen this idea for my project from my interest in motorsports and sports cars. Due to the diverse nature of these vehicles, I came across the idea of making a novel system for generating images of sports cars. The system could be used by other enthusiasts to create images of cars with the user’s preferred aesthetics. Though this project with some further development and refinement could hold practical applications. Images generated from the system could serve as a source of inspiration for manufacturers offering fresh ideas and concepts as well as video game developers who may wish to create their own vehicles rather than obtaining licenses to use real-world cars in their games.

In the following sections of this report, I will talk through the various stages of development, from planning to implementation, providing an overview of the methodologies used and challenges. The report aims to provide insight into developing an image generation system.

**Background, methodology and design:**

In the initial stages of designing the program I had set out the scope of the project, which was creating a system that could generate images of sports cars. I then collected a dataset of images of sports cars, to do this I had used images of sports cars I had taken previously as well as took additional photos of other real world sports cars inside of popular video games using the games built-in photography mode, this allowed me to collect images of different types of sports car classes (classics, sports cars, super cars, track cars) at different angles to ensure that there was diversity in the dataset.

Below are some examples of photos I had taken and used in my dataset:

The back of a blue sports car

Description automatically generatedA red sports car on a road

Description automatically generated

I then looked at the different ways I could generate images using the dataset I had looked Generative Adversarial Network (GAN), which is the process of creating images from random noise, the goal is to use the generator to fool a discriminator that tries to distinguish real images from fake ones. They both use adversarial training to improve their performances and the goal is for the generator to improve iteratively until its able to produce realistic images indistinguishable from the real ones. Resulting in high quality data generation which would be ideal in this use case. However, this method would have required significant computational resources and would be difficult to train due to GANs requiring large data sets to produce high quality outputs.

I had also looked at using Variational Autoencoders (VAEs) to use for the project. VAEs are a neural network model that use compressed representations of input data and use it to generate new data samples. As VAE use generative modelling, they can generate new data samples which enables the creation of realistic and diverse data similar to those in the training dataset. However, VAEs can struggle to capture complex data distributions with good accuracy which can lead to the generation of unrealistic/blurry samples and are sensitive to hyperparameters thus requiring careful tuning which can be time consuming and require extensive experimentation.

After considering both GANs and VAEs for my project, I opted for the latter. While GANs are able produce high quality data generation, their training demands significant computational resources and very large datasets, which didn’t seem feasible for my project. VAEs on the other hand seemed like a more interpretable approach and therefore I had decided to use the VAE model.

Early development:

Before starting development, I explored how I could implement my system. In my modules throughout my studies, I had used the TensorFlow library which is a deep learning framework for some projects in other modules, and as I was familiar with it, it would be a good way to implement my system.

I had decided to code the program in python with this library and had set my code up to use the Keras API within TensorFlow to start coding the neural network.

To start I created the encoder network, which would act as a data compressor. This encoder analyses the input images and then converts them into a condensed representation called the latent space. This is done through convolutional layers that extract essential details from the images that we are using to train the VAE. There is also the decoder network which is the counterpart to the encoder network. The decoder network functions to generate images using the representations in the latent space. This is done by expanding the compressed data into a shape that represents the original image and is then upsampled using the convolutional layers.

These two networks will serve as the foundation for the image generation system.

Once the encoder and decoder networks were created, the next step was to integrate them to function as a VAE to do this I made a class that would bring together the encoder and decoder this would be the autoencoder. This class would take input images convert them into latent representations using the encoder and then reconstruct them using the decoder class. This process would allow the model to learn a compressed representation of images that we previously input into the program.

Continued development:

After implementation of the autoencoder and setting up how the program loads images.

I added a function that would split input images from the directory into training and validation sets. This is so that I can evaluate the performance of the model on data that it hasn’t seen during training. Not only that, but it would also be useful to tune hyperparameters such as the batch size and learning rate. Doing so would aid in my development of the program and would allow me to make any fine changes if needed.

Training the program:

To train the program, I had initially created a loop that would iterate for a given number of epochs and would provide it with a bunch of augmented images and would use a validation set to assess the model. I would then use an early stopping function to stop the program if its performance did not improve after a certain number of iterations.

The initial results of the program were quite disappointing there was some detail, not enough to make out anything resembling a car of any sort and the images had lacked colour.

A screenshot of a computer

Description automatically generated

* Initial output images

Improving the program:

Now that I had a starting point, I began looking into ways that I could further improve my program.

I had adjusted the number of layers inside of the encoder and decoders. In theory this would help improve feature extraction and reconstruction. I had also tried different values for learning rates, and batch sizes in an attempt to optimise the training process.

After trying these methods the program had improved further and the images had some more detail.

A screenshot of a computer

Description automatically generated

* Images produced after some tweaks.

Whilst there was some improvement, there was still a lack of colour in these images and the images were too blurry for my liking.

After continuing to experiment with how the program was trained I by trying different methods of architecture optimisations and tuning. The program was able to generate images with colour and to further improve its performance I had used images from Unsplash.com to add copyright free images of sports cars in the real world. I did this as it was a faster way to source images, and I had used these images to see how much better my program would perform with a larger dataset. Though given the time and ability to take more photos of cars I would have liked to create a dataset that was my own.

In doing so, the program began to memorise images rather than trying to create new ones.

A screenshot of a computer

Description automatically generated

* The program had outputted a compressed version of an image inside the dataset twice.

Results and conclusion:

After making some more changes and going back and forth with different values I had changed how the program was trained, by first feeding it images and slowly adding more nosie to them alongside clean images and then asking it to reconstruct an image this would allow the program to learn to denoise these images progressively and add some more detail to the final images produced.

Here were the images that were produced in the end:

A screenshot of a computer screen

Description automatically generated

Although these images are far from what I had initially wanted to produce. Given the limited time and computing resources available to me I was unable to refine my system to produce an image that would be a clear image of a car. However, the images that were produced in the end did show shapes that resembled cars and at different angles which was cool.

Working on this project has taught me more about the possibilities of what is achievable through computational creativity and this project had allowed me to combine two things that I am dearly passionate about being motorsports and computers.

If possible, I would have liked to further develop my program by making the layers more complex and possibly adding more advanced learning methods. However, as I struggled to implement Nvidias cuda development tools into the project these images were produced on my cpu alone which was very slow and took some time to produce. By using these tools it may have allowed me to produce much better results.

Despite these challenges the project has shown progress and potential for further development and I would like to continue to develop further to see how far it can go.

Thank you for reading my report.