Team - 22KTJDATV396127 **Datathon - Kshitij IIT KGP 2022** **Code Link** - https://github.com/oyetripathi/KTJ-Datathon

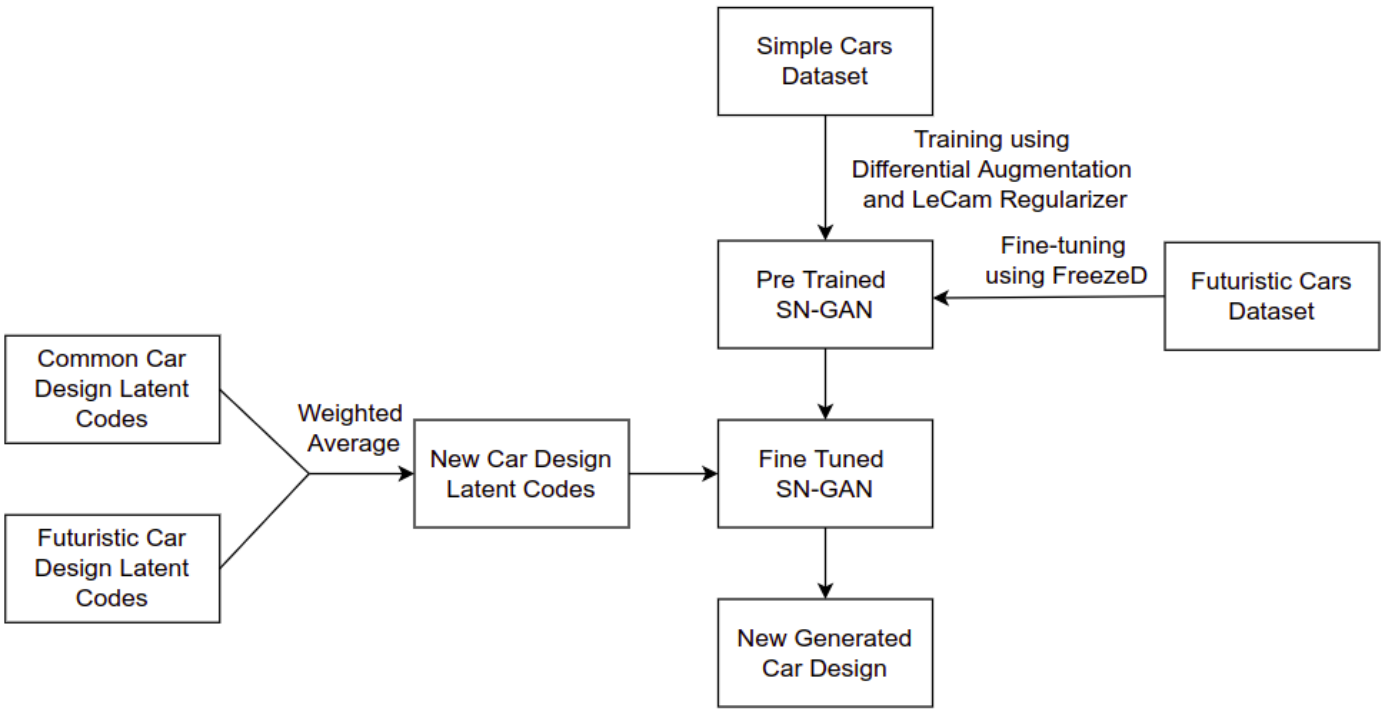
---

Summary Of Our Approach :

Our approach is based on the idea that upcoming new car designs will be similar to existing car designs in their basic structures so that designs don't lose their feasibility of implementation.

Firstly we have trained a SN-GAN on a simple car dataset with differential augmentation and LeCam regularizer. This is followed by fine-tuning of the trained GAN on a futuristic cars dataset using FreezeD technique. Further we have used Res-Net18 to extract latent codes of common and futuristic cars which are then combined using weighted average and fed into SN-GAN.

Weighted average gives us the flexibility to decide the intensity of futuristic flavor which can be added to existing designs to obtain novel ones.



Dataset used :

We use 2 datasets to train our models:

1. Simple Car Image Dataset: This dataset contains around 11,000 images of existing cars with a simple design, all the cars are facing left, front-left or front-right. This dataset was a subset of another dataset that was scraped from *The Car Connection* website and then further classified according to their view. The code to obtain this dataset was taken from following github repository: https://github.com/nicolas-gervais/predicting-car-price-from-scraped-data/tree/master/picture-scraper
2. Futuristic Concept Car Image Dataset: This dataset contains around 2000 images of concept cars with a futuristic design. All the images in this dataset were scraped from Google Images search results. The code for scraping was written completely by us. The dataset is present on our google drive: https://drive.google.com/file/d/1_98O7p_022WGyZJLeg1dkoh9yzhBPfy8/view?usp=sharing

SN-GAN :

For steady and diverse image generation, we needed a GAN which could effectively absorb the design features of existing car models and successfully reproduce them in subsequent generations. Popular GANS architectures like DC-GAN are plagued by their training instability and inaccurate density ratio estimation by the discriminators. Therefore for our-case we used SN-GAN, a type of General Adversarial Network which uses a weight normalization method called **spectral normalization to stabilize the training of discriminator networks**. In image generation tasks, **it has been verified that generated examples using SN-GAN are more diverse than the conventional weight normalization and achieve better or comparative inception scores relative to previous studies.**

Algorithm for the Spectral Normalization :

Algorithm 1 SGD with spectral normalization

- Initialize $\tilde{\boldsymbol{u}}_l \in \mathcal{R}^{d_l}$ for $l = 1, \ldots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer l:
  1. Apply power iteration method to a unnormalized weight W^l:

$$\tilde{\boldsymbol{v}}_l \leftarrow (W^l)^{\mathrm{T}} \tilde{\boldsymbol{u}}_l / \|(W^l)^{\mathrm{T}} \tilde{\boldsymbol{u}}_l\|_2 \tag{20}$$

$$\tilde{\boldsymbol{u}}_l \leftarrow W^l \tilde{\boldsymbol{v}}_l / \|W^l \tilde{\boldsymbol{v}}_l\|_2 \tag{21}$$

  2. Calculate $\bar{W}_{\mathrm{SN}}$ with the spectral norm:

$$\bar{W}^l_{\mathrm{SN}}(W^l) = W^l / \sigma(W^l), \text{ where } \sigma(W^l) = \tilde{\boldsymbol{u}}_l^{\mathrm{T}} W^l \tilde{\boldsymbol{v}}_l \tag{22}$$

  3. Update W^l with SGD on mini-batch dataset $\mathcal{D}_M$ with a learning rate α:

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}^l_{\mathrm{SN}}(W^l), \mathcal{D}_M) \tag{23}$$

Another advantage of SN-GAN is that it only has one hyperparameter to be tuned, and the algorithm does not require intensive tuning of the only hyper-parameter for satisfactory performance.
For discriminator and generator architecture, we have used Resnet architecture. We also used hinge loss for cost calculation.

We have also used LeCam Regularizer and Differential Augmentation technique to improve the generalization performance and stabilize the learning dynamics of GAN models under limited training data.

Fine Tuning :

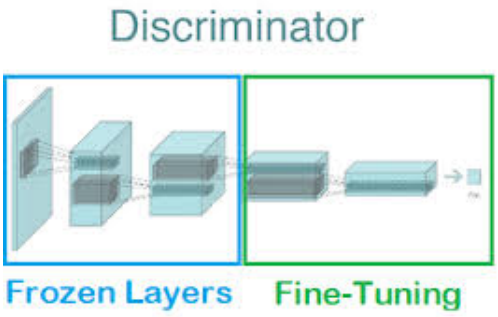
The goal for this work was to turn a generative model that had been trained on one distribution into a completely new distribution of images that didn't match any existing dataset. We approached this by taking the generator from a pre-trained generative adversarial network (GAN) trained on one dataset (here old car images) and then fine-tuned it with characteristics from another dataset (here super car images) using a classifier trained on data from both datasets.With this technique, we hoped to alter the generator such that it generates a new distribution of pictures that fuses visual information from both datasets, resulting in a distribution with novel characteristics, rather than merely modeling the distribution of images in the new dataset. We thought that by starting with a pre-trained model with excellent initial weights, we would be able to keep some properties of the original distribution, such as the spatial organization of the images, while also injecting some new traits from the other dataset.

Simple fine-tuning of GANs (both generator and discriminator) with frozen bottom layers of the discriminator, in particular, performs remarkably well. Intuitively, the discriminator's bottom layers learn general picture features, while the top layers learn to categorize the image as real or false depending on the extracted features. This dichotomous perspective of a feature extractor and a classifier (and freezing the feature extractor for fine-tuning) is not new; it has been commonly utilized for classifier training. We confirm that this viewpoint is likewise beneficial to GANs and establish a good baseline for GAN transfer learning.

FreezeD Technique:

Techniques like DropOut and Stochastic depth have already demonstrated how to efficiently train the networks without the need to train every layer. Freezing a layer, too, is a technique to accelerate neural network training by progressively freezing hidden layers.

We had the idea of FreezeD (our proposed baseline), FeezeD stands for “Freeze the Discriminator” which is a simple baseline for fine-tuning GANs. It's simply freezing the lower layers of the discriminator and only fine-tuning the upper layers performs surprisingly well. FreezeD splits the discriminator into a feature extractor and a classifier and then fine-tunes the classifier only. It’s proven to significantly outperform previous techniques used in GANs. The pre-trained part is frozen and only last layers are trained and how big the change is on the weights in a layer is governed by the learning rate.



By freezing those early layers and manipulating the last few layers. We were able to control the rendering style separately and thus generate images of a specific style. Then, the goal is to only fine-tune the classifier. What’s really surprising is that this super simple method outperforms most of the previous methods needing much more computation time using various architectures and datasets. We can see that it’s not always the most complex technique that performs bests

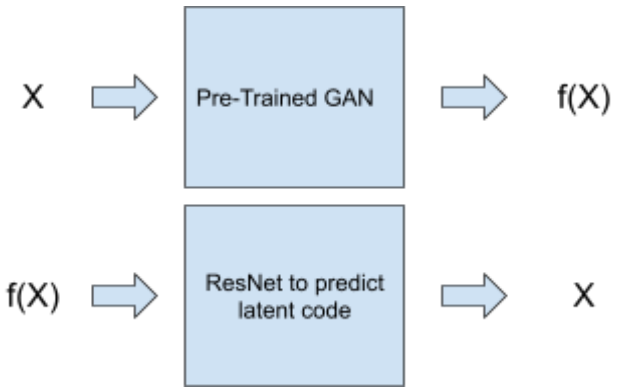
Combining latent codes to merge simple and futuristic cars:

Suppose a latent input w such that output of GAN for input w is $f(w)$, then we want w such that $f(w)$ is an image of a car that has a realistic design and a futuristic visual appeal. Let x be an input latent code for which output is an image of a simple existing car design, and y be an input latent code for which output is an image of a futuristic concept car. Our idea is to take a weighted average of *x* and *y* to produce a new latent code w which will have combined features of existing and futuristic cars, and $f(w)$ will be an image of such a car.

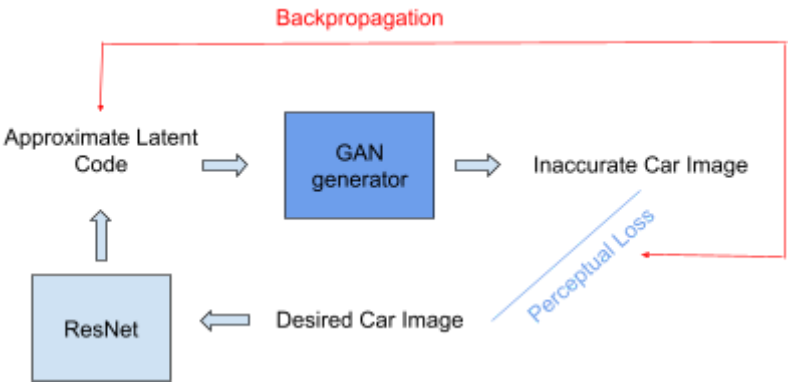
$$x: \textit{latent code of simple existing car} \quad f(x): \textit{image of simple existing car}$$
$$y: \textit{latent code of futuristic car} \quad f(y): \textit{image of futuristic car}$$
$$w = \alpha * x + (1 - \alpha) * y \quad (0 < \alpha < 1)$$
$$f(w): \textit{image of car with hybrid design of existing and futuristic car}$$

But first we need latent codes corresponding to realistic and futuristic cars while we only have images corresponding to these cars and not latent codes. So we train another ResNet model that can output the latent code given the desired output image. The training is done by producing a dataset using our trained GAN model, we input as many random latent codes as we want and store them along with their outputs ,then the inputs are used as outputs of our

ResNet model and vice versa. This idea has been extensively used in the context of Style GANs, where another model is trained to predict latent codes corresponding to face images.



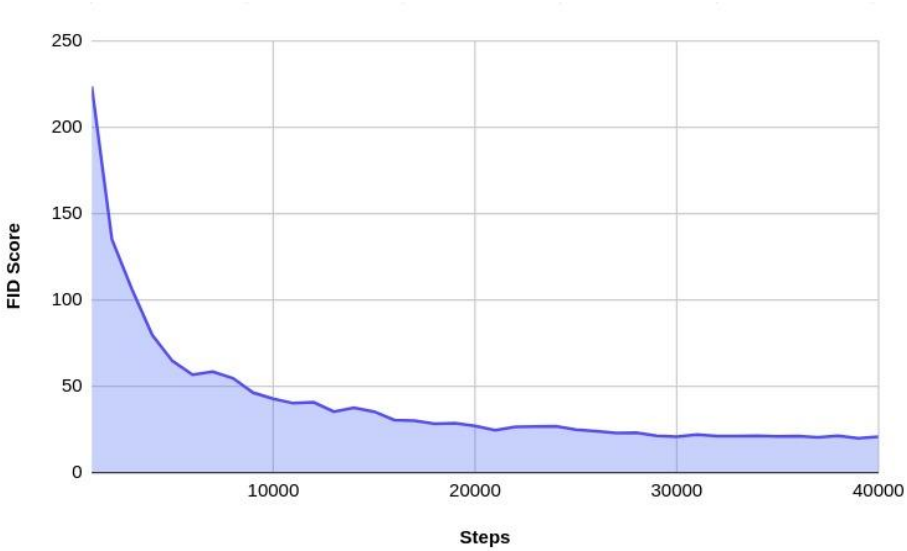
The output images of the latent codes predicted from such a ResNet are not accurate as these latent codes would only be an approximation of the real latent codes corresponding to the car images. To obtain the real latent codes corresponding to our desired car images, we pass the approximate latent codes into our GAN generator and obtain the inaccurate image and then take perceptual loss between this inaccurate image and desired image, we apply backpropagation on this perceptual loss with only latent code elements as trainable parameters.



This process of backpropagation works well, but is known to converge very slowly, to make it faster we may go a step further and take L2 loss between VGG16 (pre-trained) encodings of inaccurate and desired images instead of perceptual loss on images. These latent codes can be manipulated to “tweek” certain design features of the output car such as headlights, bumper, doors etc.

Training Process :

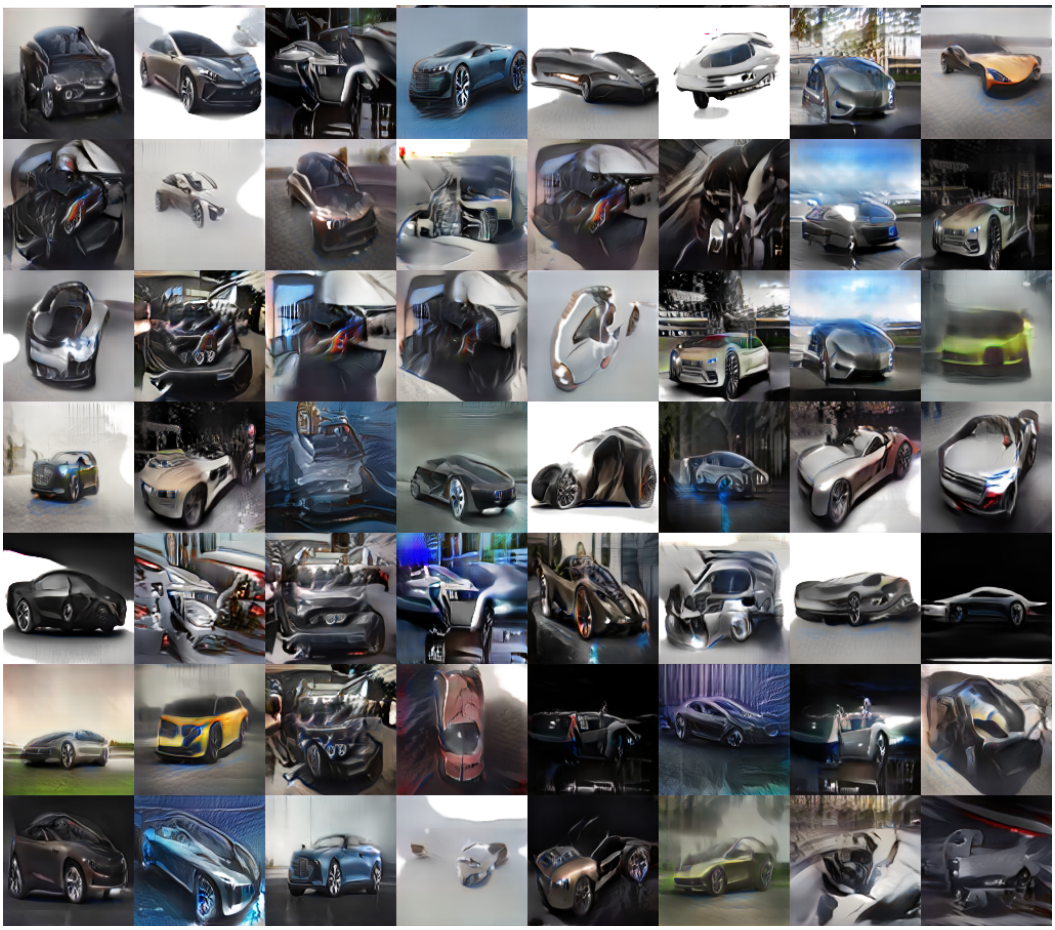
We only have images corresponding to both, realistic and futuristic cars. So to get the latent codes, we use the ResNet18 architecture trained on our realistic car datasets. We use 3 major class of our realistic cars datasets, i.e. Front Left, Front Right and Left. Total of which contains around 17,000 images. We use the ResNet18 model of resolution 128x128 pixels with Batch Size of 64 and 7 epoches. We used “ReLU” and “Adam” as activation functions and optimizer respectively.

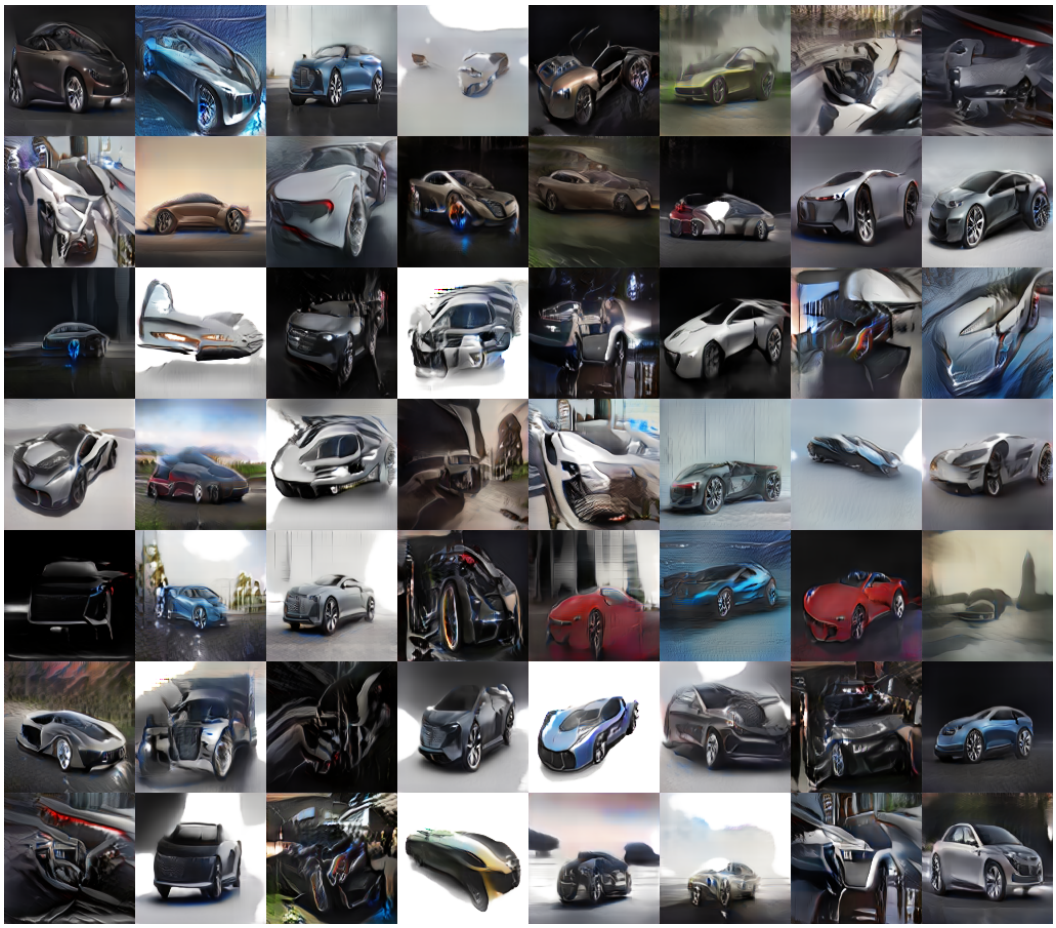


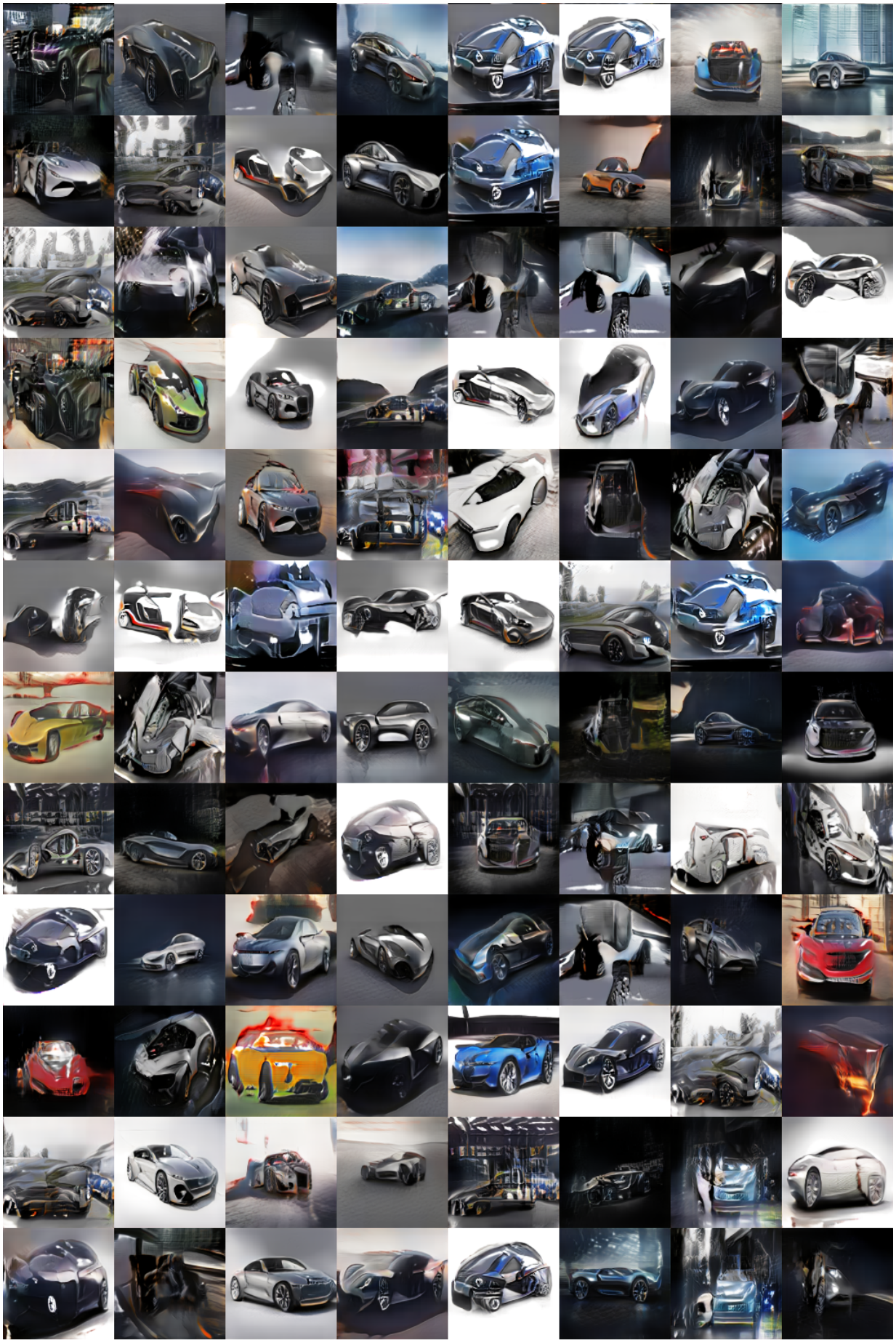
Here use the same ResNet model architecture of 128x128 pixels resolution with Batch Size of 128 and 35 epoches. We used “ReLU” and “Adam” as activation functions and optimizer respectively. Discriminator and generator learning rates are 0.0002 and 0.0005 respectively.
This is followed by special fine-tuning of the trained GAN on a customized dataset of futuristic concept cars. We used FreezeD, a simple baseline for fine-tuning GANs, instead of DropOUUT and Stochastic depth methods. We freeze the early layers of the discriminator and fine-tune the GAN based only on the last few layers of the discriminator.

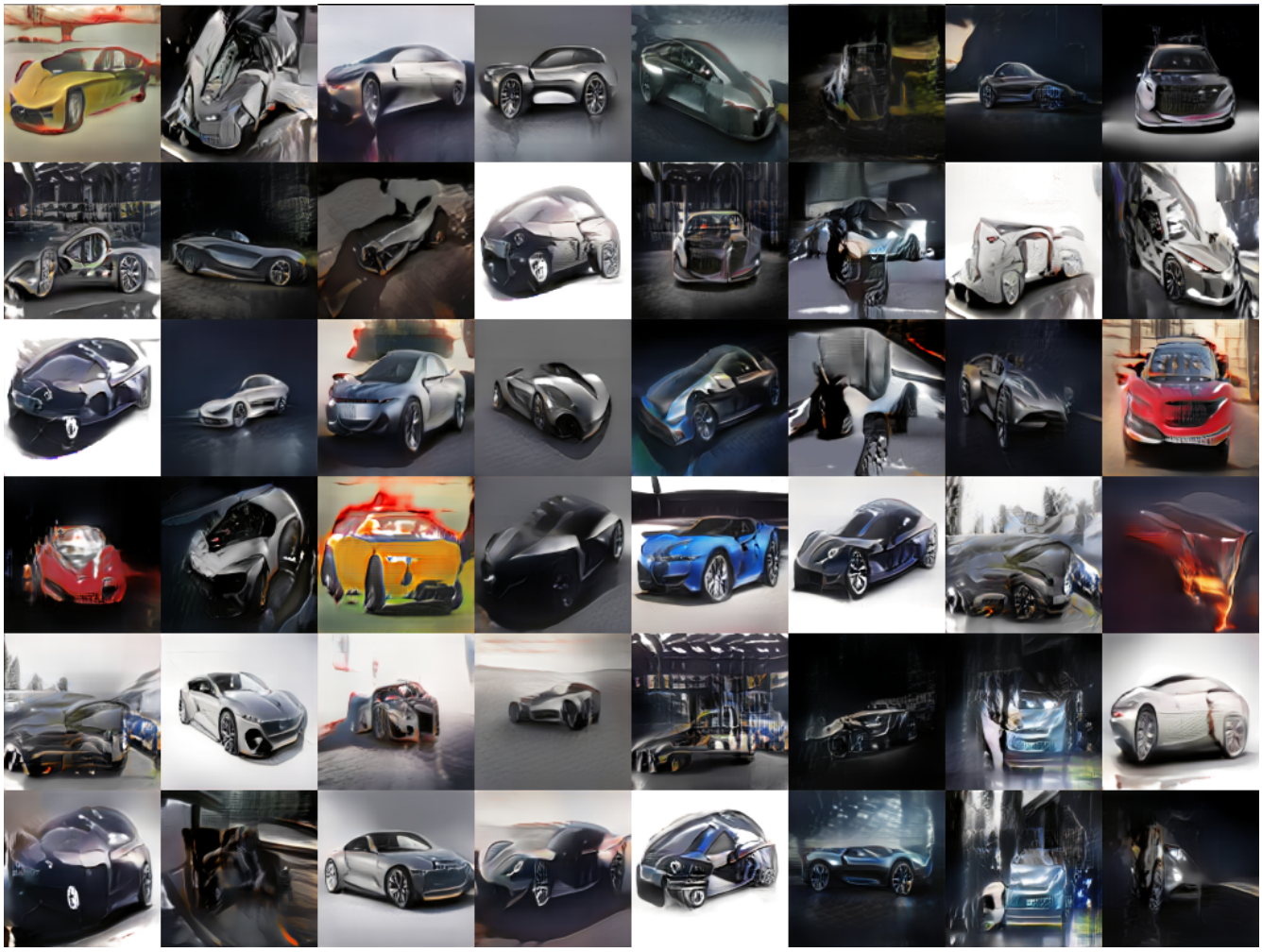
Results of Initial Prototype :

Fine Tuned : (Fig 1 and 2)









Fine Tuned with Combined Latent Codes : (Fig 3)

Future Modifications:
We are considering:

1. **StyleGAN instead of SNGAN**: Our original idea was to use StyleGANs which are better suited for the purpose of combining styles of two cars, but due to time and computational constraints, we settled for SNGAN. StyleGAN has already been used in the context of merging facial features of two different people which is very similar to our approach, hence StyleGAN will definitely improve our results.

2. **Time Conditional GAN**: Our results are largely dependent on futuristic concept car images we use to train our model, but these images are very limited. To produce futuristic car images, we can use the idea of conditional GANs. We will classify the car images according to their production years and then feed their production years along with random vectors to the GAN. The model shall learn the difference between cars produced in different years. For eg. If the generator learns the difference between cars produced in 2010 and 2020, it can use this difference to predict what cars will look like in 2030.

3. **Segmentation Mask for Cars**: The images of cars usually contain a lot of unnecessary background noise which affects the distribution of our data. To tackle this problem we can use Segmentation techniques like R-CNN to produce the mask of the region containing the body of the car. Using this mask we can produce a dataset where every car has the same background and alignment.

References :

SN-GAN : https://arxiv.org/pdf/1802.05957.pdf
Differential Augmentation : https://arxiv.org/abs/2002.10964
LeCam Regularizer : https://arxiv.org/abs/2104.03310
FreezeD Technique : https://arxiv.org/abs/2002.10964