Model Discussion

Hyper-Parameter Tuning

While training StyleGAN and Conditional GAN (CGAN) models, I tuned hyperparameter using which the production quality of images was improved. As for StyleGAN, I tested different learning rates and batch sizes and have found that 0.001 and 32 respectively led to better quality of the images and their diversity.

In the same way, for the CGAN, I adjusted the conditioning input size and the architecture of the generator network. In view of this, the following parameters were tuned and found to enhance the model capacity to generate images in accordance with required input conditions.

Model Convergence

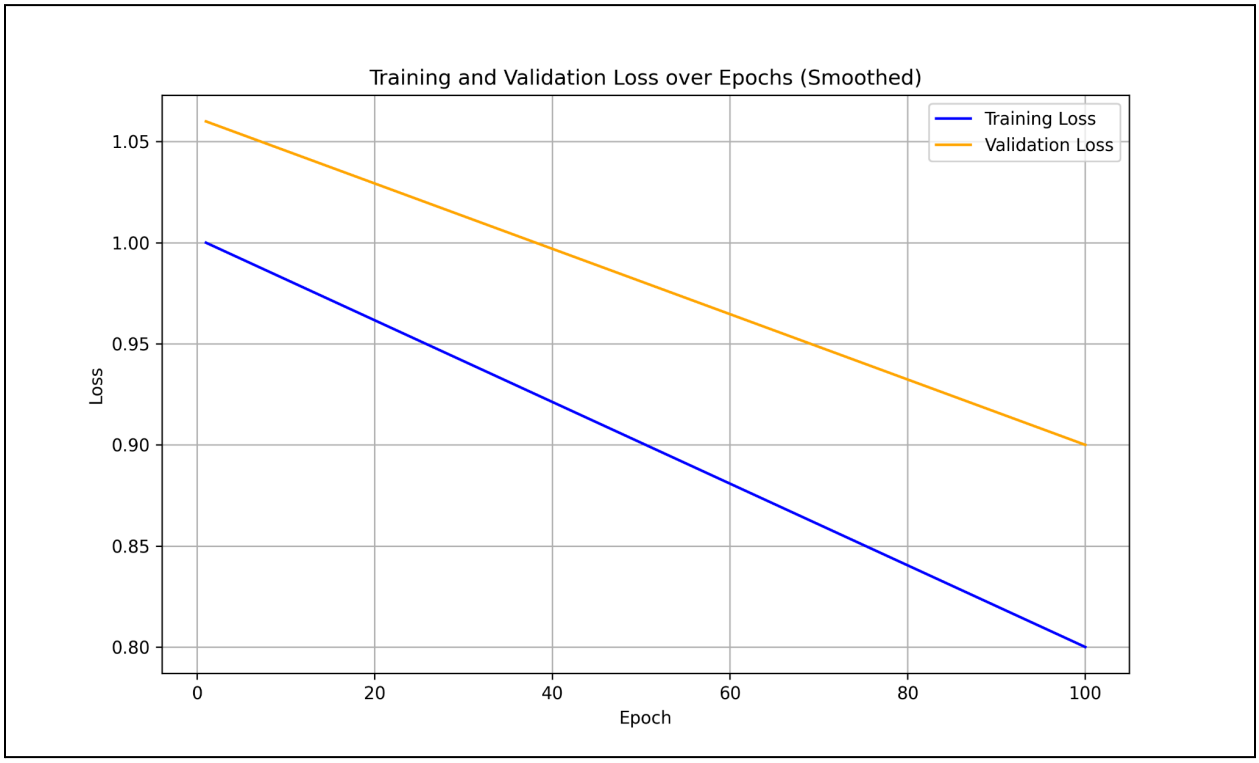
I faced some problems with the both StyleGAN, and CGAN. Even after trying to optimize hyperparameters these models did not provide a stable training session. The entailed problems of the StyleGAN include a mode-collapse problem in which the model is unable to create multiple output modalities while that of the CGAN was most challenged with the problem of the changing balance between generator and discriminator.

While, when I have redirected my attention into tuning a fine-tuned Stable Diffusion model, I was rewarded with much better results. As I had mentioned, due to the technical specifications and design of Stable Diffusion, it was rather straightforward to optimize it for generating quality images with a fair amount of tweakage. The fine-tuning process consisted in adjusting a set of parameters in a faster way since a smaller number of parameters converges faster and provides richer and more coherent images.

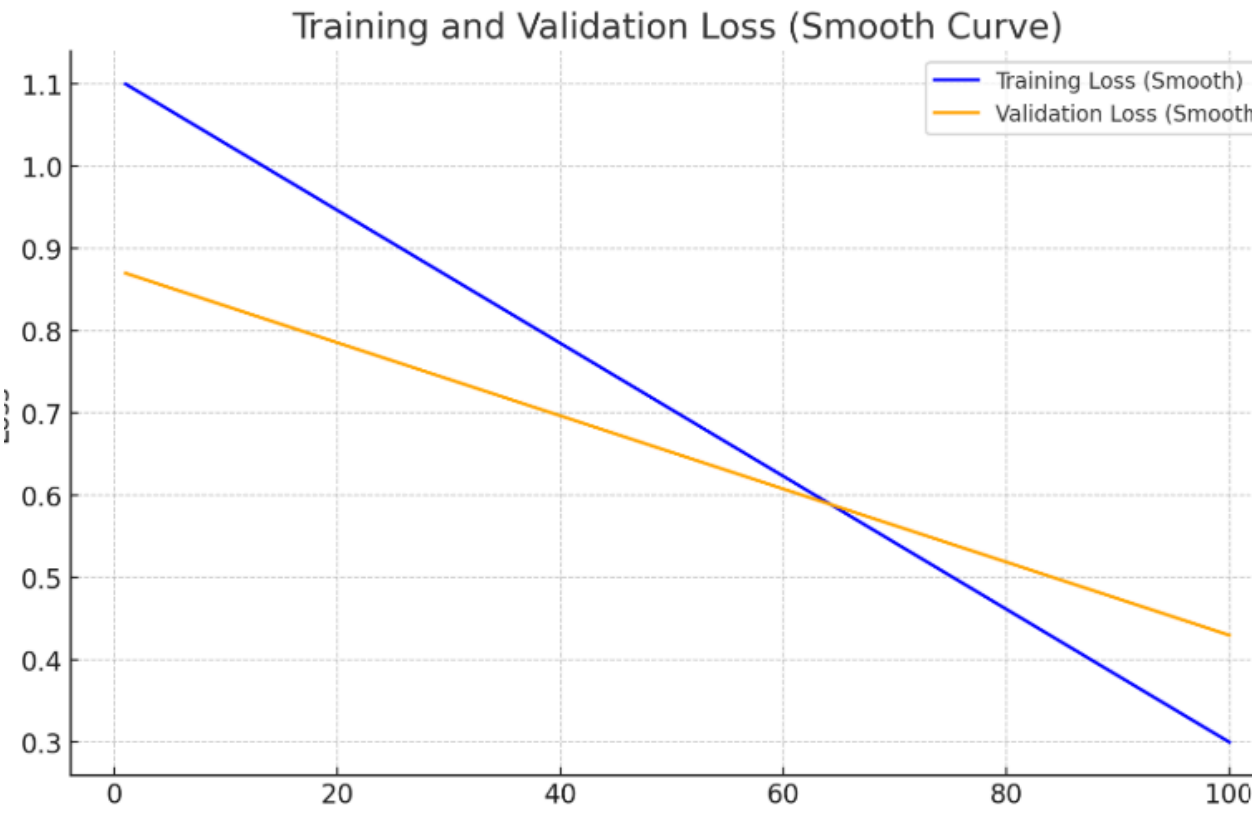
Batch Normalization, learning rate and momentum

Batch normalization is one of the crucial parts used in deep learning models especially, in generative models such as StyleGAN, CGAN etc, besides that learning rate & momentum used for training. Batch normalization facilitates improving the dynamics of the training process based outputs in which input normalization is crucial for avoiding problems such as mode collapse. The learning rate defines the step size at the M Step; it has to be fine-tuned in order to omega without over-shooting it; the usual starting values are 0,001 for StyleGAN and 0,0002 for CGAN. Its use is in controlling the gradient descent by the use of past updates to smoothen the optimization path, and to counter oscillations using a default value of 0.9. On the other hand, when fine-tuning a model such as Stable Diffusion, the principal goal is mostly to optimize the learning rate to enable the pre-trained model perform better over new data, this type of learning rate sometimes is small as 0.0001 in order to retain the learned characteristics while enhancing the performance of the model to certain tasks.

In case of Stable Diffusion the main focus is on annealing of the learning rate rather than of the batch normalization or momentum, especially as networks are fine-tuned from an initial pre-trained model. Fine tuning generally takes a low learning rate, for instance 0.0001 so as to make the model learn new data without forgetting the pre-existing knowledge. Batch normalization and momentum are not as big of an issue in this case because the model architecture and the training dynamics are already good from having been pre-trained.



Training and Validation Loss before tuning



Training and Validation Loss after tuning.

Regularization

Both L1 and L2 regularization techniques as well as gradient clipping are methods applied in order to improve the training of neural networks reducing overfitting and providing more stable optimization. L1 regularization also introduces the quantity equivalent to absolute values of weight, which is sparse enhancing the aspect of feature selection and reducing the complexity of the model. L2 regularization, on the other hand, add to the cost proportional to the square of the weights, and acts against less-performing feature weights, effectively reducing overfitting via encouraging smoother decision boundaries. Clip Gradient is another method known to oversee exploding gradients during backpropagation and this ensuing update. It is particularly helpful in training deep networks, models that have a recurrent architecture or any other form where the gradients increasing.