Transformer-based models for protein structure generation

Abhijit Bhatnagar
abhijitb@cs.washington.edu

David Juergens
davidcj@uw.edu

Prashant Rangarajan
prashr@uw.edu

Individual contributions: Initial implementation of the baseline GAN was performed by all group members. D.J. optimized the baseline BCE training loop, implemented WGAN-GP, implemented the baseline-extended and ResNet architectures, derived and implemented the KL divergence loss metric, and set-up, executed, and analyzed the 89 production GPU experiments. A.B. performed hyperparameter search on the the baseline WGAN model, and implemented from scratch the image inpainting solution. He then executed 20 hyperparameter search runs on 64 x 64 and 128 x 128 images using training data to find optimized values and finally produced the inpainting results on test images. P.R. helped optimize the baseline model, devised/implemented a general architecture for running all the experiments and analysis, implemented the SAGAN model, implemented the different GAN training loops - WGAN, hinge etc. Report writing/editing was done equally by all three members.

1 Introduction

Proteins are large, complex molecules that are omnipresent in the living world. They carry out a plethora of functions in living organisms, such as catalysis of reactions, providing structure in cellular environments, transport of various bio-molecules or promoting the expression of genes. The backbone of any protein structure is a chain of amino acids, which are important organic compounds, 20 of which occur in nature (also known as residues). Modifying just one amino acid in a protein's sequence can affect the protein's overall structure and function. However, the *functional space* that naturally occurring proteins access is a minute fraction of all their possible chemical functionalities. This simply must be the case given that a protein with 100 amino acids has 20^{100} possible sequences, and there are approximately 10^{12} different proteins that exist naturally [10]. As such, the ability to generate brand new (de novo) proteins that are not naturally occurring promises access to exotic chemistries that can have wide ranging biomedical and industrial applications, giving us the ability to create next generation therapeutics, catalysts, and materials [10, 7, 15, 19].

One of the first steps toward harnessing this expanded functionality is the ability to accurately and efficiently sample valid protein structures from the distribution of all possible structures [23]. Moreover, it is especially desirable to sample from this distribution conditioned on some other functional or structural prior [23]. For example, one might desire a protein that contains a special *motif* (fold, shape or active site) in a certain portion, while not caring what the rest of the protein contains so long as it is stable and promotes the formation of the desired motif. The ultimate goal is to create designer proteins that perform the exact functionalities required for a given biomedical application with minimal human intervention or input. Traditionally, de novo design of proteins has been done via heuristics, which often leads to these models facing problems such as inefficient or poor feature engineering. Recent work has been done on using deep learning in order to learn the features automatically for protein generation. In this paper, we investigate if protein structure generation via distance/orientation map generation with generative adversarial networks (GANs) can be improved with the help of the transformer-based models [24].

2 Mathematical background of Generative Adversarial Network (GAN)

The main purpose of a GAN is to model or fit some unknown probability distribution, and then be able to generate samples from that distribution. A GAN consists of two networks - a generator G and a discriminator D which compete with each other in a minimax game until an equilibrium is reached between the two agents. After training is complete, the discriminator is discarded and the generator can be used to draw samples from the probability distribution being modeled.

During a single pass in the GAN training loop, there are two steps. One step trains the discriminator, and the other trains the generator. Typically, the discriminator step is performed first. In this step, samples from the true probability distribution $p(x)$ are drawn, as well as samples from the latent space $p(z)$ where high dimensional noise vectors are drawn from. The generator then produces *faked* images from the source noise. The goal of the discriminator is then to learn the probability that an input image is real. Consequently, its loss function is composed (via binary cross entropy, BCE) of 2 factors - how far from a zero probability it ranks the faked images, and how far from a perfect probability of 1 it ranks the true images.

After the discriminator receives a training step, it is then treated as a static *oracle* during the generator training step. Here, new noise z is sampled from the latent space and passed through the generator. The generators sample is then passed to the discriminator to receive a score on how good the discriminator thinks the image is. The further from 1.0 that $D(G(z))$ is, the higher the generator loss should be. BCE loss is then back-propagated to the generator's parameters. Formally, for a minibatch of noise $\{z_1, ..., z_n\}$ drawn from $p(z)$, and a minibatch of real images $\{x_1, ..., x_n\}$ drawn from $p(x)$, the loss for discriminator function D given generator function G may be written as: $-\frac{1}{n}\sum_{i=1}^{n}[\log(D(x_i)) + \log(1 - D(G(z_i)))]$. The loss for the generator function given discriminator D (and a new minibatch of noise) is written as: $-\frac{1}{n}\sum_{i=1}^{n}\log(1 - D(G(z_i))$. We wish to minimize both of these functions during training with respect to the generator parameters, θ_G, and discriminator parameters θ_D.

3 Related Work

3.1 Protein Generation Methods without Deep Learning

De novo protein generation typically consists of two steps - generation of protein backbones, and selection of amino acid side-chain types and conformations to stabilize the backbone conformation and to adopt specific three-dimensional active site geometries optimized for function [17]. Some approaches have been to take certain native protein backbones from existing databases of protein structures, and making small structural adjustments to them [13, 20]. One such recent approach is to use a superfamily of protein structures known as the NFT2 and run an enumerative algorithm to create a diverse range of structures from it [5]. Another approach is to assemble proteins using local structures. They use blueprints to extract secondary structures with the desired functionality, and then combine these structures into an overall protein structure [22]. The Rosetta modeling suite [8] is a benchmark method for designing protein structures. It makes use of a heuristic energy function, and samples native protein fragments to fold backbones - optimizing over amino acid types and orientations to obtain the most likely amino acid sequence for the required designed backbone. While these methods are useful to generate proteins, they are often inefficient or not versatile, often due to poor engineering design of the features.

3.2 GANs for Protein Generation

Anand et al. [1] have shown it is possible to model the distribution of naturally occurring protein structures using deep generative modeling. They were able to generate novel protein structures in a fast and scalable manner for use in protein design applications. In their approach, they used GANs to generate inter-residue alpha-carbon (CA) instance maps for proteins. The distance maps encode protein structures, which can be "folded" back into real protein structures. The model is shown to successfully generate novel protein structures, as well as predict missing sections of corrupted protein structures.

The generator and discriminator architectures used are based on convolutional neural network (CNN) layers. To the best of our knowledge, the approach of using a CNN-based GAN is a novel method of generating protein structures. Deep convolutional GANs have been shown to perform well at the image inpainting task [25], however the inductive biases built into CNN layers almost certainly limit their performance on the task of valid protein structure generation. Specifically, a critical assumption that CNNs make is that of information locality - i.e., the state of a pixel in an image is only affected by its immediate neighbors. This assumption is probably inaccurate when operating on 2D distance maps from proteins, as physical interactions in 3D Euclidean space can be between amino acids that are very long range in sequence space. Finally, GANs are known to encounter training instability and mode collapse issues while training for image generation. Also, very often they face other problems such as vanishing gradients, and low dimensional supports [3, 21].

3.3 Transformer-based Models for Image Generation

Image Transformer: Parmar et al. [18] use an Image Transformer to generate a higher-resolution images or sequences from the observed lower-resolution images. They modify the self-attention to increase the size of images that they can process, by attending primarily to local neighborhoods (inspired by CNNs). They also make use of a more tractable loss function for image generation. For image generation tasks, Parmar et al. show that transformers help address the issue of CNNs capturing only local information by making use of self-attention, while also being more efficient than RNN-based models, which have a large receptive field. However, since they use a more traditional transformer architecture, scalability is a concern, as the self-attention matrix has a quadratic complexity with respect to the input.

SAGAN: Zhang et al. introduced the Self-Attention GAN [26] i.e. SAGAN. It makes use of the self-attention mechanism introduced in [24], and similar to transformer models, can model long-range dependencies for image generation tasks. So, they address the common concerns of CNN based models of not capturing long range dependencies. The paper also introduces some additional techniques to stabilize the GAN training process, such as using spectral normalization for both the generator and the discriminator.

GANsformer: Hudson et al. introduced the GANsformer [11], which is a type of GAN where both the discriminator and generator networks are transformers. These transformers use a bipartite attention model instead of the traditional self-attention for a transformer. It addresses the typical scalability issues of transformers by improving on the efficiency over traditionally constructed transformers while also capturing long range dependencies. The authors are able to show that incorporation of transformers into an image generation task yields improvements over CNN-based methods. Apart from the GANsformer, there are also other models such as TransGAN [14] that attempt to generate GANs void of any form of convolution. Active research is being done to obtain superior models for image generation tasks using transformers.

3.4 Deep Learning Models for Protein Generation

Some work has been done making use of VAEs in order to design novel protein structures [9]. Other work has been done where deep network hallucination has been used for de novo protein synthesis [2]. Very little work has been done prior using self-attention mechanisms and models such as the transformer in protein generation. The Structural Transformer [12] makes use of a modified transformer and uses a graphical approach in order to model proteins and generate proteins de novo.

4 Methods

4.1 Data curation and preprocessing

We are using protein structure data from the Protein Data Bank (PDB) [6], which is a publicly available set of 170,000 structures of proteins. Each file contains the (x,y,z) coordinates of every atom in a protein as experimentally determined by methods such as X-ray crystallography, nuclear magnetic resonance, or cryo-electron microscopy. One of our group members already had access to copy of the entire PDB from early 2020, which had structures separated according to their sequence similarity. To curate our set of real distance maps, we wrote code to parse the coordinates out of 15000 of these files with high resolution (less than 3A). The coordinates of the proteins were parsed, and CA-CA distance maps were calculated for all proteins and split into train/test sets. Once a set of real distance maps was in-hand, it could be used during training by randomly sampling $C \times C$ maps

along the diagonal of real maps, where C is the crop size. Finally, before emitting the crops into a training step, values within the crop were scaled down by a constant factor of 100, as in [1].

4.2 GAN Implementations

In this section, we describe the various GAN architectures that were implemented. Note that in all architectures, the generator takes in multivariate Gaussian noise vector $z \sim \mathcal{N}(0,1) \in \mathcal{R}^{100}$ and output CA-CA distance maps $\in \mathcal{R}^{\text{crop}\times\text{crop}}$. Theoretically, a well-trained generator can then produce distance maps can then be used to construct protein backbones in 3D downstream.

4.2.1 GAN architectures

Baseline convolutional network

For our baseline, we implemented a version of the model described in Anand et al. [1]. The GAN in the model consists of deep convolutional layers (i.e. a DCGAN model) for both the generator and discriminator. For example in the case of 64×64 size images, both the discriminator and generator contain 5 blocks. A typical block consists of the following layers: *2D Convolution, Batch normalization, Leaky RELU activation*. We tried different training loops for the GAN, which are detailed in section 4.2.2.

Extended-baseline and ResNet
We wrote two different expansions of the baseline network. The first is called an *extended baseline*, and has a generator and discriminator with a single additional convolutional layer with 'same' padding added to operate on the output image of the generator, or the input image in the discriminator. This architecture was created to see how minor changes to the baseline affect performance. The second expansion is the **ResNet** expansion. Between every upsample or downsample in the generator or discriminator, ResNet includes a set of 4 dilated residual blocks, each with 2 *convolutions, LeakyRelu nonlinearities, batch normalizations, and one skip connection*. This architecture was meant to test if larger receptive field (via dilations and depth) could improve GAN performance.

Self-Attention Generative Adversarial Networks (SAGAN)
We adapt the SAGAN method used for image generation to protein generation. As mentioned earlier, the SAGAN makes use of self-attention layers that were introduced in the Transformer model [24] in order to capture long-range dependencies between components in the image. Say x represents the feature vector of the image. Then, we first map x to a new feature space using a linear transform as follows: $f(x) = W_f x$, $g(x) = W_g(x)$ where W_f and W_g are weights that will be learned by the model. And the main self-attention layer is as follows:

$$\beta_{ji} = \sigma(f(x_i) \cdot g(x_j))$$

where $\sigma(\cdot)$ is the sigmoid function, and β_{ji} represents how much the i^{th} position in the image influences or *attends* to the synthesized j^{th} position. Thus, this learned parameter can help capture any kind of long-range dependency. And the final output of the attention layer is

$$o = (W_o \beta_i) \cdot (W_h x_i)$$

where $\beta_i = [\beta_{ji}]_j$ and W_h and W_o are also learned weights. In the actual model, all these weights can implemented using 1×1 convolution layers. These self-attention layers are present in both the generator and the discriminator. Apart from the self-attention, [26] also has some other suggestions for training the GANs more stably. One such suggestion is to use spectral normalization on both the generator and discriminator. [16]. This helps prevent unusual gradients and also parameters from blowing up. Overall, the basic block of the Generator/Discriminator is as follows: *Self-Attention (using 2D convolutions), Spectral normalization, Leaky RELU activation.* Apart from that, the model also using different learning rates for the discriminator and the generator.

4.2.2 GAN training procedures (losses)

Binary Cross-Entropy (BCE): The BCE training loop is the standard loop that has been described in detail in section 2 i.e. we make use of a binary cross entropy loss in order to train the GAN, with the discriminator loss $L_D = -\frac{1}{n}\sum_{i=1}^{n}[\log(D(x_i)) + \log(1 - D(G(z_i)))]$ and generator loss $L_G = -\frac{1}{n}\sum_{i=1}^{n}\log(1 - D(G(z_i))$.

WGAN-GP: Wasserstein GAN + Gradient Penalty (WGAN-GP), is a modification of the usual training BCE loop of the GAN [4]. Unlike the traditional GAN that makes use of KL Divergence in order to find the similarity between the generator and discriminator distances, this method makes use of the Wasserstein aka Earth Mover's distance to do so. It is said to produce a smoother gradient as compared to the usual training loop. WGAN involves learning a 1-Lipschitz function f, which can be learned using a neural network. The GAN losses are as follows: $L_D = -\frac{1}{n}\sum_{i=1}^{n}[f(x_i) - f(G(z_i)))]$, and $L_G = \frac{1}{n}\sum_{i=1}^{n} f(G(z_i))$, both of which have to be minimized. In order to ensure that f is Lipschitz, we have to restrict the gradient in some manner. One way is to clip the weights, and the other way is to impose a gradient penalty. WGAN-GP involves adding a gradient penalty term to the loss function in order to maintain this Lipschitz property.

Hinge Loss: The Hinge loss is another variant of the the GAN training loop. It's similar to the original GAN training loop, except we slightly modify the loss functions to make it a hinge loss, as suggested in Zhang et al. [27]: $L_D = -\frac{1}{n}\sum_{i=1}^{n}[\min(0, -1 + D(x_i)) + \min(0, -1 - D(G(z_i)))]$ and $L_G = -\frac{1}{n}\sum_{i=1}^{n} D(G(z_i))$

4.3 Evaluation metrics

4.3.1 Qualitative analysis of training curves and distance maps

Before developing a method to quantify performance in GAN training sessions (see KL divergence below), much of our hyperparameter and model architecture search was performed by looking at GAN training curves and output CA-CA distance maps during training. Qualitatively, we sought training curves that weren't divergent, yet also didn't have the generator and discriminator "collapse" into a local minimum (as did often occur). In our distance maps, we looked for diversity of structure, distances similar to those in native distance maps, and realistic looking map patterns reminiscent of protein secondary structure.

4.3.2 Tracking KL divergence from a native distribution

Because the value of GAN training losses themselves often don't provide insight into the dynamics or quality of a training session, it would be beneficial to have an easily calculated metric to measure how far from the training distribution the generated data are. Such a metric would ideally allow a user to quickly judge the quality of GAN training sessions, and potentially even bias / regularize the generator toward natural data distributions. To do this in our study, we implemented a metric to measure the KL-divergence of the frequency distribution of generated CA-CA distances with some reference distribution of distances. To calculate the reference distance distribution, we sampled 1000 crops from different protein backbones in our training dataset. These crops were then flattened into vectors, and the CA-CA distances placed into 1 of 20 bins ranging from 0-100 Å, each of width 5 Å. Once all distances were placed into bins, bin counts were then normalized to sum to 1 and make a probability distribution of distances from 0-100 Å. Reference distributions were made for size 16, 64, and 128 crops and are depicted in figure 1.

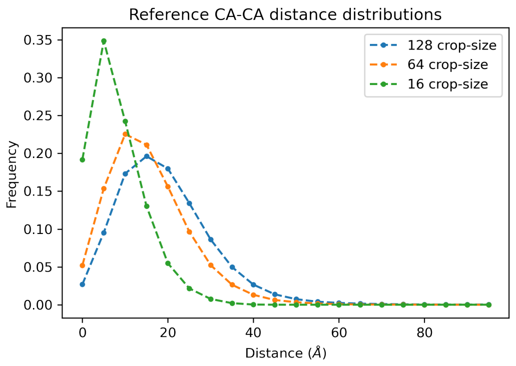


Figure 1: Reference CA-CA distance frequency distributions.

Given the frequency distribution of distances in a generated map (or a batch of maps), it is now possible to quantitatively probe the quality of said maps by calculating the KL divergence between the generated and reference distance distributions. Both the absolute value as well as the dynamics of this value can be used to interrogate training session quality for GANs. For example, the KL divergence between a few sampled crops of size 128 and their respective reference distribution is relatively low (Figure 2). In contrast, Figure 2 also shows that if a GAN experiences a collapse in which many generated distances are near 0 (as often happens) it results in a CA-CA distance frequency distribution with elevated KL divergence to the reference.

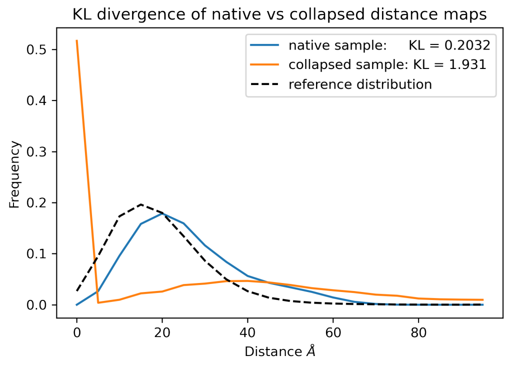


Figure 2: KL divergence loss calculation for a real and a (simulated) collapsed CA distance map.

4.4 Inpainting

Inpainting is the task of reconstructing missing regions in an image [25]. After training our models, we attempted to put them to a real world task of generating contextually correct missing portions of protein structures. This is an protein design problem where parts of the protein structures corresponding to desired functions are known, and the objective is to generate remaining parts. The intuition behind the inpainting task is that if the model has learnt the protein structures really well, then it should be able to distinguish between a valid and an invalid protein structure. In our generative adversarial networks, the generator generates a valid protein structure given a random noise vector, and the discriminator opines on whether the generated structure is valid or not. We use the 2 models to optimize the noise vector which finds an image which has the least amount of loss between the original image and the generated image. We use the method described in [1] with 2 models:(1) a 64 x 64 baseline convolutional network, and (2) a 128 x 128 baseline convolutional network. We use the test set to generate corrupted images by taking real images of protein distance maps and masking out a randomly chosen $\frac{1}{20}$ stripe of region symmetrically across the diagonal. We use the losses as described in [1]:

(1) **Context loss**: An l_1 reconstruction loss with higher weighting for pixels nearer to the masked region of the input. Given input x and binary mask M delineating the area to be inpainted, the weighting term W is found by convolving the mask complement M^c with a 2D identity filter of fixed size. For our experiments with 64-residue and 128-residue maps, we set the filter sizes to 9 by 9 and 15 by 15, respectively. x is the corrupted image generated from the test set by masking out regions. (2) **Prior loss**: Prior loss with respect to generated image used for inpainting. (3) **Discriminator loss**: Discriminator loss on the final inpainting solution. The losses and final objective are calculated as

$$\begin{aligned}
L_{context} &= ||(W * M^c) * (G(z) - x)||_1 \\
L_{prior} &= \log(1 - D(G(z)) \\
L_{disc} &= \log(1 - D(M * G(z) + M^c * x) \\
L_{total} &= L_{context} + \lambda L_{prior} + L_{disc}
\end{aligned}$$

where we seek to $\min_z L_{total}$ and we set weighting term $\lambda = 0.003$. We optimize z, the noise vector, with Adam (β_1 = 0.5, β_2 = 0.999) with learning rate 10^-5 for 50000 steps. We enforce that the generator output G(z) be positive and symmetric.

5 Results, Discussion, and Conclusions

Here, we walk the reader through the (most relevant) results from our numerous model training attempts. They are contained in figures 3 and 4, and subplots from these figures will be referred to as "plot X" or "experiment X", where X is the letter in the left part of the subplot title. Please further note that the four grey lines, in order of left to right, indicate the time points during training when the upper left, upper right, lower left, and lower right distance maps (respectively) were sampled using a fixed noise vector sent through the generator. We tried to choose interesting time points to look at. The first grey line (top left map) is always the initial output from the generator. Finally, in an effort to save space, the colorbars on the CA distance maps were omitted. Bright yellow pixels correspond to close (approaching zero) euclidean distance between atoms, and dark blue/purple pixels correspond to distances *at or beyond* 20 angstroms.

Results
Plots A and B in Figure 3 are the baseline model trained on crops of size 128 and 64, respectively. Things to notice qualitatively are the stable and competitive generator and discriminator losses, and the evolution of output maps to those with more realistic structure. Notice this also correlates with a decrease in KL divergence from their respective reference distance distributions. Further note the lowest *stable* KL divergence in the 64x64 model is around 0.5, before a sharp change in losses near the end of training. Here, there seems to be a mode collapse into a local minimum of BCE and KL divergence, but it manifests as a physically nonsensical map. Plot C depicts results from a baseline model trained using the WGAN-GP loss, which clearly yielded poor results. After numerous attempts to improve the performance of WGAN-GP training, it was discarded from further study. Notice here, however, a positive result (and a trend that continues) is the *absolute value* and/or *dynamics* of the KL divergence serving as an indicator of generated distance map quality.

Plots D and E depict results from the baseline extension models (both 128 crop size). Interestingly, adding just a single layer to both the generator and discriminator in otherwise identical training sessions ruins trajectory stability and performance. Notice the oscillation in KL divergence allows one to see hidden dynamics in the training sessions that are otherwise unrevealed by the generator and discriminator losses - again showing its usefulness. Notice further that the due to superior KL divergence and ending map quality in experiment E with respect to experiment D, there is partial support for the hypothesis that GANs with extra parameters may overfit to their training distribution more easily AND that added noise or dropout may be a valid method to be explored for preventing this tendency.

Plots F and G are the ResNet training sessions at crop size 128 with different dropouts. Performance is both quantitatively and qualitatively worse than the extended baseline, and further supports the over-fitting hypothesis.

Plots H and I depict the results from SAGAN training sessions using BCE and Hinge loss, respectively, at a crop size of 64x64. When, comparing plot H to B, we excitingly see that SAGAN BCE quantitatively outperforms the baseline BCE model in terms of matching native distance distributions by achieving a very stable KL loss falling considerably lower than that achieved by the baseline 64x64 model. Note in plot I that Hinge loss was very unstable, and resulted in a collapsed model very quickly. Hinge loss was no longer pursued after this result. Further note that although tempting due to the low KL in experiment A, plots A and H cannot be compared due to different crop size.

Figure 5 depict results of inpainting objective on baseline models trained on crops of size 128 and 64, respectively. In both the plots, the first 3 images show the original uncorrupted image, a mask which is symmetric around the diagonal and the corresponding corrupted image generated by combining the original image with the mask. The remaining images show the progression of optimizing the inpainting during the noise vector optimization session. We observed that the models did better on optimizing the 64 by 64 image than in optimizing the 128 by 128 image. The output image is produced by combining the area of the generated image corresponding to the mask with the original image. In Figure 5, it can be seen that the first output image has dispersed sections of green across the entire mask area, and it is relatively easy to identify that the mask. As the training progressed, the mask area begins to blend in well with the original area. This optimization is, however, not noticeable in Figure 6 on an image of size 128 by 128. We hypothesize it is due to the 128 x 128 model not learning the latent space very well and therefore not being able to navigate it.

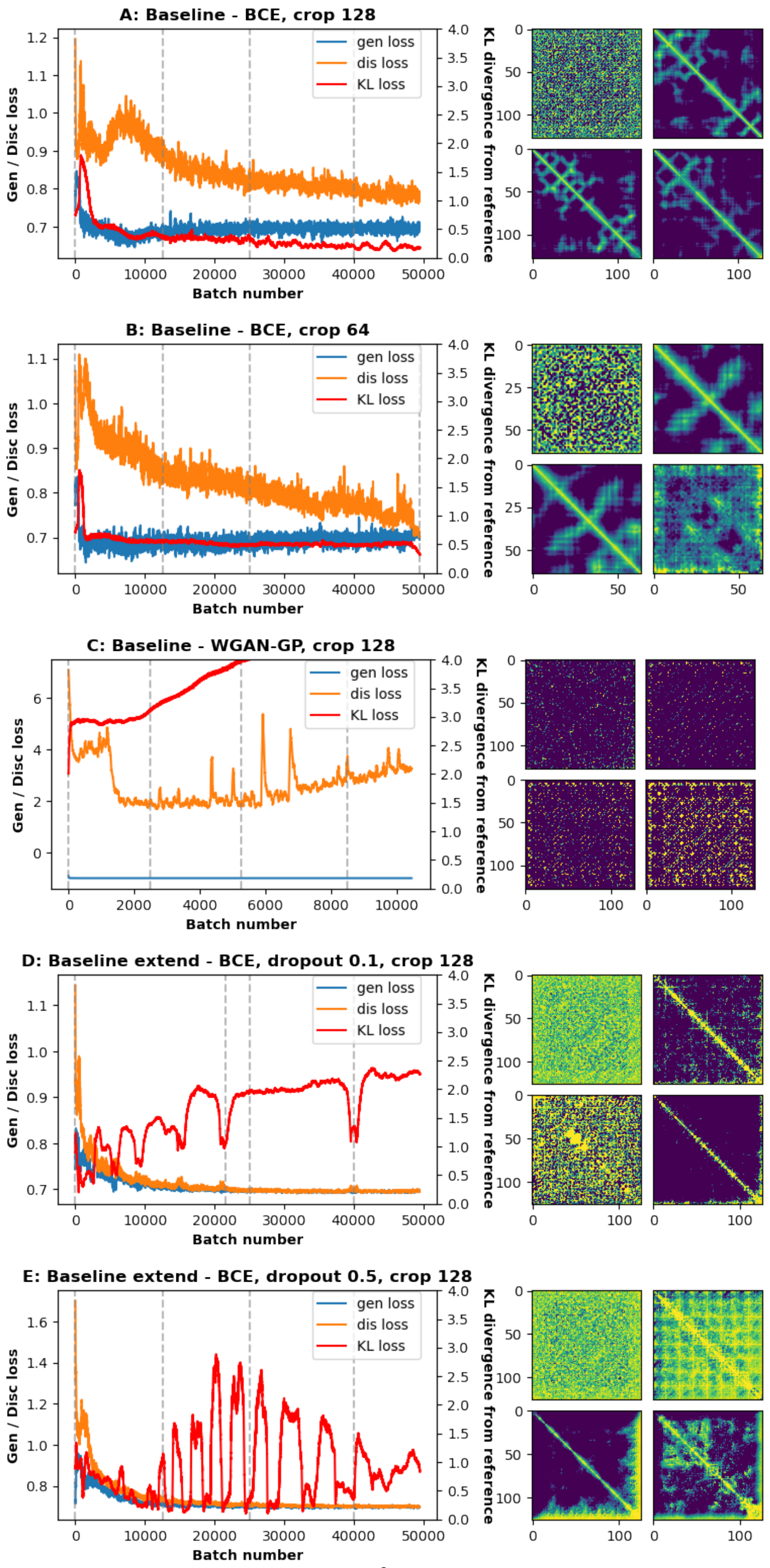


Figure 3: GAN training sessions 1: Grey lines in left plots (left to right) show where distance maps were sampled from during training (upper left, upper right, lower left, lower right, respectively).

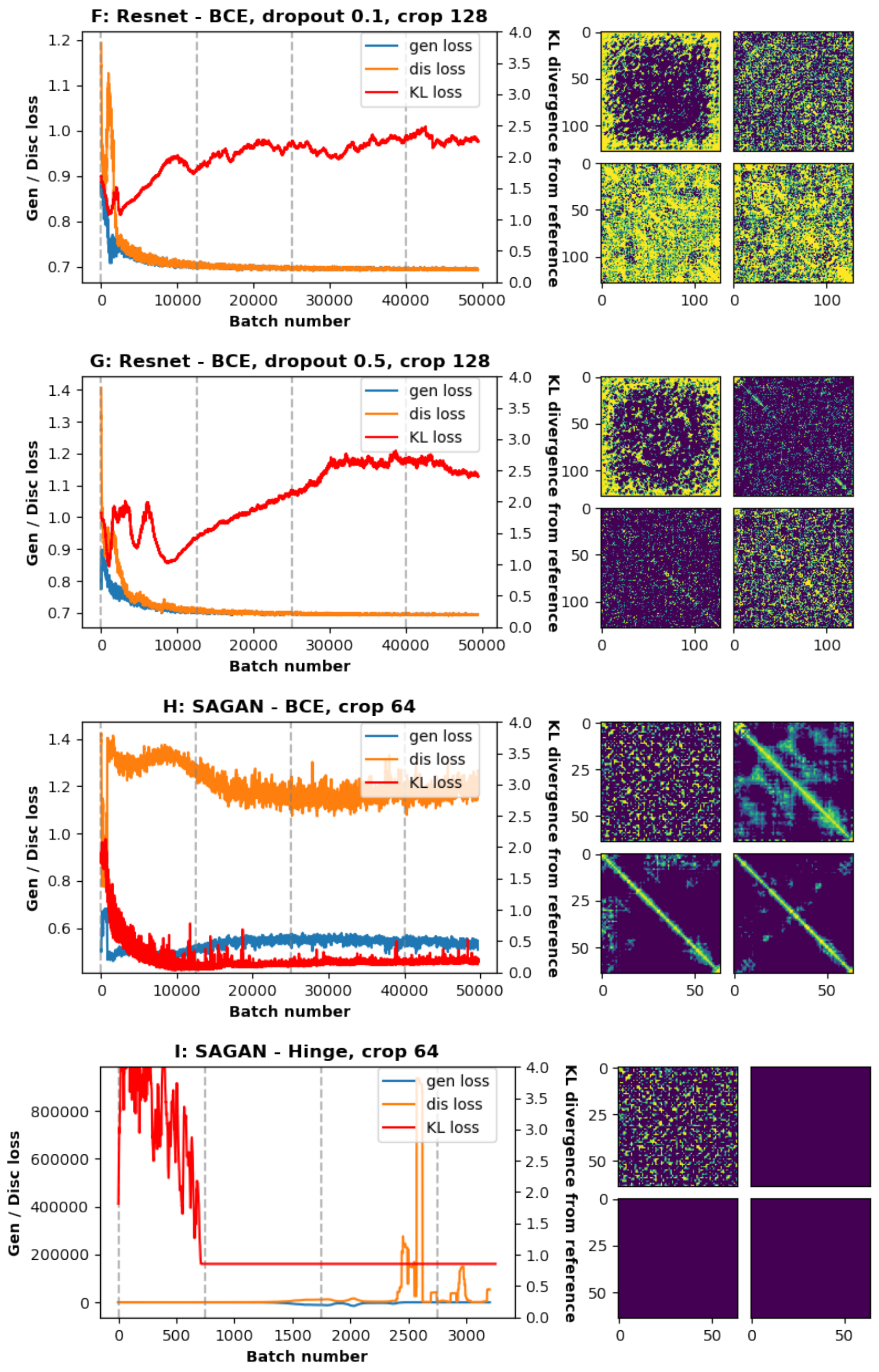
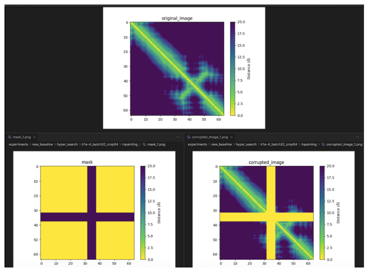
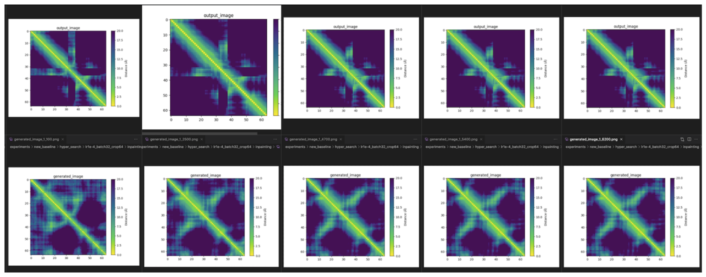
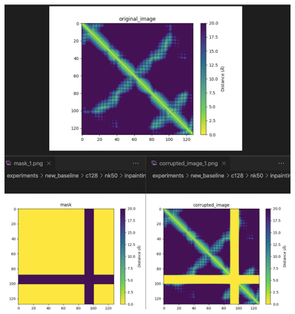


Figure 4: GAN training sessions 2: Grey lines in left plots (left to right) show where distance maps were sampled from during training (upper left, upper right, lower left, lower right, respectively).


(a)


(b)


(c)

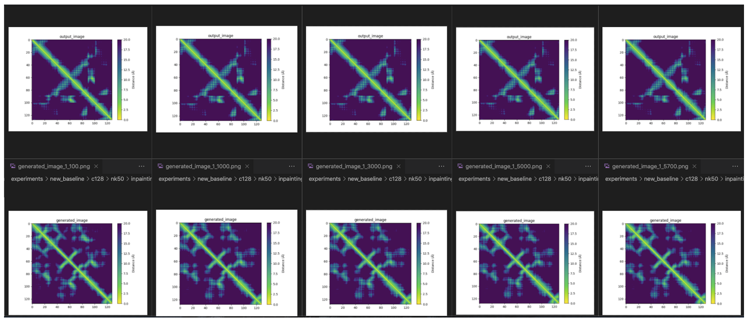

(d)

Figure 5: The first row is the results of inpainting on a 64 x 64 image, and the second row is the results of inpainting on a 128 x 128 image. Figures 5a and 5c show (1) the original image, (2) the mask and (3) the produced corrupted image by overlaying the mask on the original image. Figures 5b and 5d show output images in the first row and the generated images in the second row on the 64 x 64 and 128 x 128 images respectively.

Discussion and Conclusions

The KL divergence loss metric turned out to be a valuable asset in this study, and reveals itself as a necessary but insufficient criterion for generation of realistic protein structures. As seen in plot B, low absolute value of KL divergence does not always correspond to high quality structure generation. However, quality seems highly correlated with the metric, and the dynamics of KL divergence are invaluable for interrogating and debugging training session dynamics. It would be interesting to study the effect of *enforcing* the generator to optimize for low KL-divergence solutions during training.

The comparison of plots H and B provides some support for the working hypothesis of our project: that the Transformer self-attention mechanism could provide GANs with better ability to model and generate protein structures than convolutions alone. Though the results are far from conclusive, they do seem very promising and warrant further study of the self attention mechanism (and its derivatives, i.e., GANsformer) in GANs for protein structure generation. It must be noted, however, during the execution of experiments H and I we also attempted training with crop size 128, but the operations were too costly to fit in GPU. This is an important, tangible example of the real issues associated with the quadratic computational complexity of the self-attention mechanism. It brings into question the utility of such a model, even if it were to outperform traditional (convolution based) networks. This computational burden is only exacerbated when moving to the real problem of generating full proteins, which are commonly 300-400 amino acids long, and can reach a length of 900+.

Finally, it is worth noting here the truly exceptional difficulty of the training task attempted in this study. Of the 89 total production GPU training sessions attempted with various architectures, losses, and hyperparameters, exactly 1 produced a result that was quantitatively better than those produced by models from [1]. Experiments D and E (and most of the other 89 we tried) demonstrate the *hyper*-sensitivity and brittleness of a GAN approach to the protein structure generation task. Though promising results were eventually encountered, it earnestly calls into question the validity of this modelling paradigm entirely. More attractive, robust, and accurate solutions may lie in the direction of auto-regressive models, and diffusion-based generators.

References

[1] Namrata Anand and Possu Huang. Generative modeling for protein structures. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[2] Ivan Anishchenko, Tamuka M. Chidyausiku, Sergey Ovchinnikov, Samuel J. Pellock, and David Baker. De novo protein design by deep network hallucination. July 2020.

[3] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks, 2017.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan, 2017.

[5] Benjamin Basanta, Matthew J. Bick, Asim K. Bera, Christoffer Norn, Cameron M. Chow, Lauren P. Carter, Inna Goreshnik, Frank Dimaio, and David Baker. An enumerative algorithm for de novo design of proteins with diverse pocket structures. *Proceedings of the National Academy of Sciences*, 117(36):22135–22145, August 2020.

[6] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 01 2000.

[7] Longxing Cao, Inna Goreshnik, Brian Coventry, James Brett Case, Lauren Miller, Lisa Kozodoy, Rita E. Chen, Lauren Carter, Lexi Walls, Young-Jun Park, Lance Stewart, Michael Diamond, David Veesler, and David Baker. De novo design of picomolar sars-cov-2 miniprotein inhibitors. *bioRxiv : the preprint server for biology*, page 2020.08.03.234914, Aug 2020. 32793905[pmid].

[8] Rhiju Das and David Baker. Macromolecular modeling with rosetta. *Annual Review of Biochemistry*, 77(1):363–382, June 2008.

[9] Raphael R. Eguchi, Namrata Anand, Christian A. Choe, and Po-Ssu Huang. IG-VAE: Generative modeling of immunoglobulin proteins by direct 3d coordinate generation. August 2020.

[10] Po-Ssu Huang, Scott E. Boyken, and David Baker. The coming of age of de novo protein design. *Nature*, 537(7620):320–327, Sep 2016.

[11] Drew A. Hudson and C. Lawrence Zitnick. Generative adversarial transformers, 2021.

[12] John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[13] Lin Jiang, Eric A Althoff, Fernando R Clemente, Lindsey Doyle, Daniela Röthlisberger, Alexandre Zanghellini, Jasmine L Gallaher, Jamie L Betker, Fujie Tanaka, Carlos F Barbas, et al. De novo computational design of retro-aldol enzymes. *science*, 319(5868):1387–1391, 2008.

[14] Yifan Jiang, Shiyu Chang, and Zhangyang Wang. Transgan: Two transformers can make one strong gan, 2021.

[15] J. Kaplan and W. F. DeGrado. De novo design of catalytic proteins. *Proceedings of the National Academy of Sciences*, 101(32):11566–11570, 2004.

[16] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018.

[17] Xingjie Pan and Tanja Kortemme. Recent advances in de novo protein design: Principles, methods, and applications. *Journal of Biological Chemistry*, 296:100558, 2021.

[18] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer, 2018.

[19] Harley Pyles, Shuai Zhang, James J. De Yoreo, and David Baker. Controlling protein assembly on inorganic crystals through designed protein interfaces. *Nature*, 571(7764):251–256, July 2019.

[20] Daniela Röthlisberger, Olga Khersonsky, Andrew M Wollacott, Lin Jiang, Jason DeChancie, Jamie Betker, Jasmine L Gallaher, Eric A Althoff, Alexandre Zanghellini, Orly Dym, et al. Kemp elimination catalysts by computational enzyme design. *Nature*, 453(7192):190–195, 2008.

[21] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 2234–2242, Red Hook, NY, USA, 2016. Curran Associates Inc.

[22] Kim T Simons, Charles Kooperberg, Enoch Huang, and David Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and bayesian scoring functions. *Journal of molecular biology*, 268(1):209–225, 1997.

[23] Doug Tischer, Sidney Lisanza, Jue Wang, Runze Dong, Ivan Anishchenko, Lukas F. Milles, Sergey Ovchinnikov, and David Baker. Design of proteins presenting discontinuous functional sites using deep learning. November 2020.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[25] Raymond A. Yeh, Chen Chen, Teck-Yian Lim, Mark Hasegawa-Johnson, and Minh N. Do. Semantic image inpainting with perceptual and contextual losses. *CoRR*, abs/1607.07539, 2016.

[26] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 7354–7363. PMLR, 09–15 Jun 2019.

[27] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks, 2019.