40. Sripawadkul, V., Padulo, M., Guenov, M.: A comparison of airfoil shape parameterization techniques for early design optimization. In: 13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference (2010)
41. Xu, K., Kim, V.G., Huang, Q., Mitra, N., Kalogerakis, E.: Data-driven shape analysis and processing. In: SIGGRAPH ASIA 2016 Courses. SA '16, Association for Computing Machinery, New York, NY, USA (2016)
42. Yang, C., Huang, F.: An overview of simulation-based hydrodynamic design of ship hull forms. Journal of Hydrodynamics, Ser. B **28**, 947–960 (12 2016)
43. Yasong, Q., Junqiang, B., Nan, L., Chen, W.: Global aerodynamic design optimization based on data dimensionality reduction. Chinese Journal of Aeronautics **31**(4), 643–659 (2018)
44. Yonekura, K., Wada, K., Suzuki, K.: Generating various airfoil shapes with required lift coefficient using conditional variational autoencoders (2021)
45. Zhu, F., Qin, N., Burnaev, E., Bernstein, A., Chernova, S.: Comparison of three geometric parameterization methods and their effect on aerodynamic optimization. In: Eurogen. pp. 758–772 (2011)

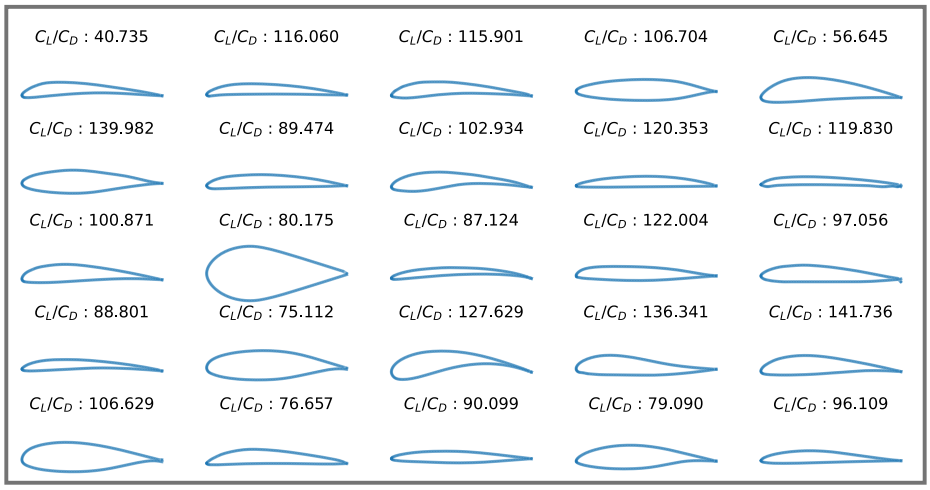

Fig. 7. Airfoils from UIUC dataset and their corresponding C_L/C_D values.

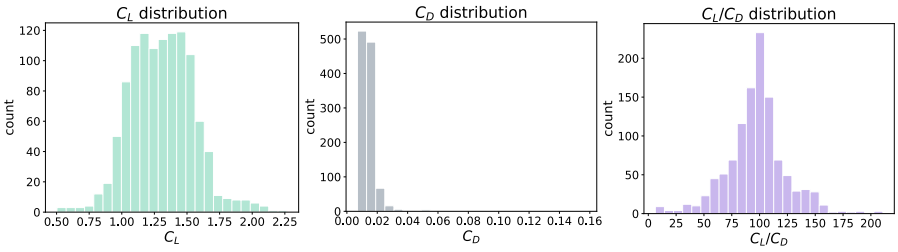

Fig. 8. Distribution of C_L, C_D and C_L/C_D ratio for all training samples.

6 Appendix

6.1 Network Architectures

DCGAN DCGAN consists of a generator network and a discriminator network. The generator takes an (n+1)-dimensional vector as input. n represents the dimension of the noise vector, which is concatenated with a C_L/C_D value and then passed as an input to the generator. As shown in Table 3, the generator network consists of upsampling and convolution layers placed alternatively. A LeakyReLU activation follows each convolution layer. The last layer produces a $500 \times 500 \times 1$ grid, the generated image. The discriminator is a CNN consisting of an input layer that takes in $500 \times 500 \times 1$ size image as input, followed by four convolutional layers and a fully connected (FC) layer of size one. The FC layer is followed by a Sigmoid activation that gives the probability of an image being a real image.

Table 3. DCGAN Architecture

Generator	**Discriminator**
Input: $\mathbb{R}^{n+1}$	Input: $500 \times 500 \times 1$
FC 128×125^2	4×4, Conv 16, LeakyReLU, stride 2
Upsampling, scale 2	4×4, Conv 32, LeakyReLU, stride 2
3×3, Conv 128, LeakyReLU, stride 1	3×3, Conv 64, LeakyReLU, stride 2
Upsampling, scale 2	3×3, Conv 128, LeakyReLU, stride 2
3×3, Conv 64, LeakyReLU, stride 1	FC 1, Sigmoid
3×3, Conv 1, Tanh	

***β*-VAE** β-VAE model consists of an encoder and a decoder, where the encoder is a CNN and the decoder is an upsampling network. The encoder takes the image of size $500 \times 500 \times 1$ as an input and maps it to a latent vector of size n through a series of convolutional layers and a fully connected layer. The latent vector is then conditioned with the corresponding C_L/C_D value of the input airfoil to form a vector of size n + 1. This resultant vector is passed as an input to the decoder. Decoder consists of a fully connected layer that maps the latent vector to a feature map of size 32 and depth of 256, followed by four transpose convolutions alternated by LeakyReLU activation functions and the Sigmoid activation layer at the end of the network. The details of the overall architecture of β-VAE model are described in Table 4. We use binary cross-entropy for reconstruction loss and weighted KLD loss i.e. β.

Table 4. β-VAE Architecture

Encoder	**Decoder**
Input: $500 \times 500 \times 1$	Input: $\mathbb{R}^{n+1}$
4×4, Conv 32, LeakyReLU, stride 2	FC $256 \times 32 \times 32$
4×4, Conv 32, LeakyReLU, stride 2	3×3, ConvTrans 256, LeakyReLU, stride 2
3×3, Conv 128, LeakyReLU, stride 2	3×3, ConvTrans 128, LeakyReLU, stride 2
3×3, Conv 256, LeakyReLU, stride 2	4×4, ConvTrans 64, LeakyReLU, stride 2
FC n	4×4, ConvTrans 32, Sigmoid, stride 2

FactorVAE FactorVAE [18], is an extension of a VAE framework that achieves better disentanglement and reconstruction quality than VAE. FactorVAE consists of three networks: an encoder, a decoder, and a discriminator. Implemented encoder and decoder architectures are similar to VAE / β-VAE architecture but

Table 5. FactorVAE Architecture

Encoder	**Decoder**
Input: $500 \times 500 \times 1$	Input: $\mathbb{R}^{n+1}$
4×4, Conv 32, LeakyReLU, stride 2	FC $256 \times 32 \times 32$
4×4, Conv 32, LeakyReLU, stride 2	2×2, ConvTrans 256, LeakyReLU, stride 2
3×3, Conv 128, LeakyReLU, stride 2	3×3, ConvTrans 128, LeakyReLU, stride 2
3×3, Conv 256, LeakyReLU, stride 2	3×3, ConvTrans 64, LeakyReLU, stride 2
2×2, Conv 512, LeakyReLU, stride 2	4×4, ConvTrans 64, LeakyReLU, stride 2
FC n $\times$ 2	4×4, ConvTrans 32, Sigmoid, stride 2

Discriminator
Input: $\mathbb{R}^n$
FC 1000, LeakyReLU
FC 1000, LeakyReLU
FC 1000, LeakyReLU
FC 1000, LeakyReLU
FC 1000, LeakyReLU
FC 2

are deeper. An additional convolutional layer and transpose convolution layers are added to the encoder and the decoder, respectively. The discriminator network is a feed-forward NN with five fully connected layers, each of size 1000 followed by another Fully connected layer of size 2. The discriminator is trained to discriminate between the latent vectors encoded by the encoder and those sampled [18]. The details of all three networks of the implemented FactorVAE are described in Table 5.

6.2 Hyperparameter Search

A hyperparameter is a parameter that is used to control the learning algorithm. Unlike the other parameters, such as weights and biases of a neural network which are learned implicitly (also known as learned parameters), the values of the hyperparameters need to be explicitly stated. The choice of values greatly impacts the quality of training and by extension, the quality of results. Hence, optimizing or tuning the hyperparameters is crucial in training deep models.

β-VAE has an important hyperparameter which is β, the weight assigned to the KLD term in the VAE's objective. The value of β controls the degree of disentanglement in VAEs, the higher the β, the better is the disentanglement result [17]. FactorVAE, in addition to β has an additional hyperparameter γ, that controls the total correlation (TC) loss. The higher the value of γ, the better the disentanglement results [18]. A common hyperparameter to all the implemented

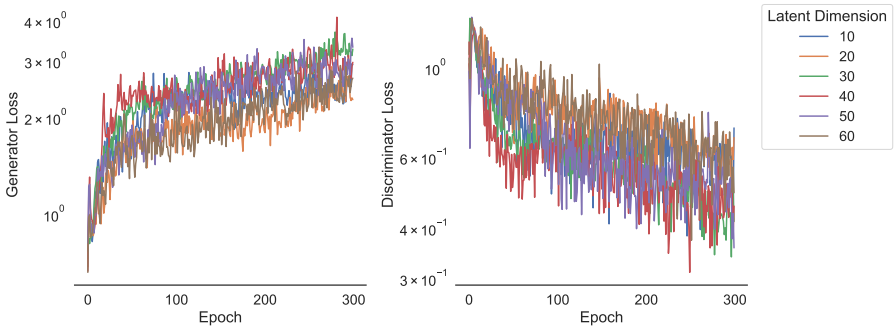

Fig. 9. Generator loss (left) and discriminator loss (right) of DCGAN for different latent dimensions.

models is the size of the latent dimension or noise vector, which also determines the quality of the results. Given the importance of these hyperparameters, there isn't a rule of thumb that can help to decide their values which can optimally solve the problem. The right set of hyperparameters minimizes the predefined objective, the loss function, thereby yielding the optimal model. Thus, we experiment extensively with a different set of hyperparameters and monitor the change in loss of our models.

In order to choose the right β, we experiment with different values ranging from 10 to 90 with a step size of 20. Figure 11 depicts the reconstruction loss and KLD loss for different values of β. We can observe that the reconstruction loss is low for lower values of β but the KLD loss remains high for lower values of β. On the other hand, reconstruction loss is slightly higher for high values of beta than the reconstruction loss for lower values of β and the KLD loss is lower for higher values of β but high for lower values of β. Hence, β value creates a trade-off between the reconstruction loss and KLD. Similarly, the latent dimension also creates a trade-off between the two losses. Figure 11 shows how both losses differ for changing latent dimensions. By observation, higher values of latent dimension result in lower reconstruction loss but higher KLD loss. Whereas lower values of latent dimension give higher reconstruction loss but lower KLD loss.

Based on the objective of FactorVAE, the TC term is weighted with a hyperparameter - γ. Higher γ is supposed to give better disentanglement without affecting the reconstruction loss and KLD. To validate that the quality of generation remains unaffected by γ, the FactorVAE model is trained with different values of γ, and the reconstruction loss, KLD, and TC is observed. For this experiment, the β value is kept constant at 50 since, the β-VAE model gives a better reconstruction vs. KLD trade-off for a β value of 50. Reconstruction loss, KLD, and TC loss are unaffected by different values of γ. This suggests that the value of γ does not affect the quality of generation in FactorVAE.

In the case of DCGAN, for optimization of latent dimension to obtain the lowest generator loss and discriminator loss, the DCGAN model is trained for

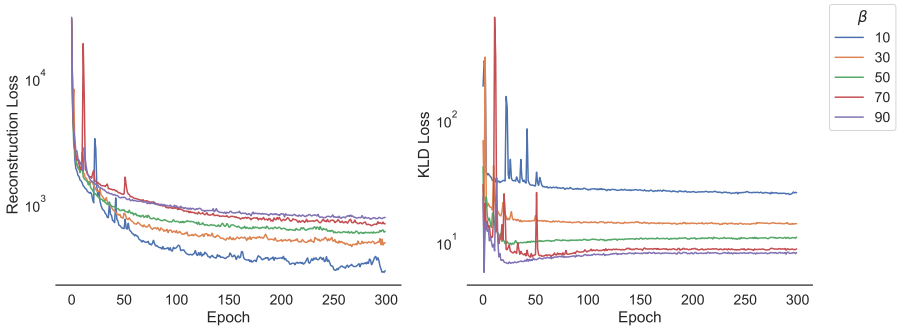

Fig. 10. Reconstruction loss (left) and KL divergence (KLD) loss (right) of β-VAE model for different β values. For lower values of β, the reconstruction loss is low but KLD loss is high and vice versa.

different sizes of latent dimension. Figure 9 shows the generator loss and discriminator loss for different latent dimensions. The latent dimension sizes vary between 10 to 70 with a step size of 10. By observation, unlike the reconstruction loss β-VAE, there is no difference in generator loss for different latent dimensions, but the discriminator loss is slightly lower for sizes 40 and 50. In the next Chapter, the outputs of the implemented models, along with the qualitative and quantitative evaluation and comparison, is discussed.

Visualizing real and fake manifolds of β-VAE To gain an intuitive understanding of the difference in the performance of β-VAE model for different values of β, we visualize the latent vectors (extracted from encoder) of real and fake samples in Figure 12. Each feature vector is the size of the latent dimension of the model. First, the feature vectors are transformed into a 2D space using PCA [1], and then the 2D vectors are further passed to t-SNE [26] to get the non-linear correlations between samples. The resultant samples from t-SNE algorithm are also placed in a 2D space. Figure 12 shows the real and fake samples in a 2D space for all the β-VAE models trained with different dimensions. Convex hulls [13] are also drawn around the real and fake samples to show their distinct manifolds.

We can observe that the real and generated samples for lower latent dimensions are closely spaced, and the manifolds are also almost similar. Whereas, for higher dimensions, the fake manifold is much bigger than the real manifold such that the fake manifold completely encompasses the real manifold. Thus, the models with higher latent dimensions have samples generated with latent vectors considerably away from the real samples. Thus, it is clear that the DnC scores in Figure 4 correlate well with the placement of samples in latent space.

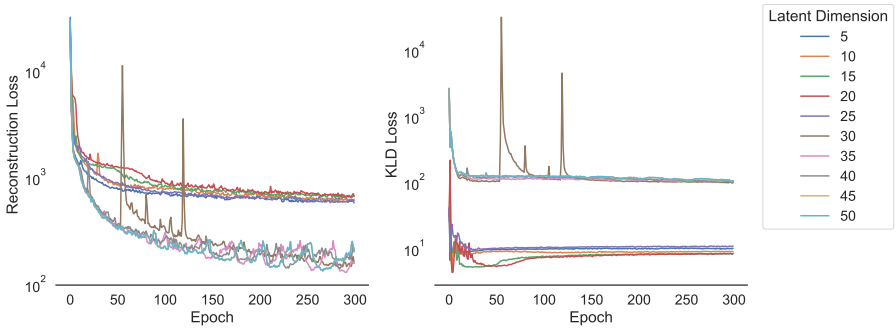

Fig. 11. Reconstruction loss (left) and KL Divergence (KLD) loss (right) of β-VAE model for different latent dimensions. Depending on the latent dimensions, there is a trade-off between reconstruction loss and KLD. For higher latent dimension, the reconstruction loss is considerably low, but the KLD is high and vice versa.

6.3 Parameterization using β-VAE

Some examples of original and approximated airfoils with their IoU are shown in Figure 13. We can see that the original and reconstructed airfoils match almost perfectly with minute differences.

6.4 Metric

Precision and recall were originally proposed by Sajjadi et al. [33] to assess the generative models for two aspects of generation quality - fidelity of generated samples and diversity. Precision measures the fraction of samples that are realistic, whereas recall measures the fraction of training manifold covered by the generator. But this approach leads to overestimating manifolds and might give misleading results; thus, [34] proposed an improvised version of Precision and Recall where the manifold is estimated using k-nearest neighbors. First, real and fake samples are taken, and their feature vectors are extracted from a generative model. Let Φ_r and Φ_g be the feature vectors for real and generated samples, respectively such that $|\Phi_r| = |\Phi_g|$. Now, manifold estimation is done by drawing k spheres, each of radius r, around k nearest neighbors, where r is the distance of the k^{th} nearest neighbor.

Following Eqn. is a binary function determining whether a given sample lies within the estimated manifold.

$$f(\phi, \Phi) = \begin{cases} 1 \ , \ if \ \ ||\phi - \phi^{'}||_2 \leq ||\phi^{'} - NN_k(\phi^{'}, \Phi)||_2, \ \ for\ at\ least\ one\ \phi^{'} \in \Phi \\ 0 \ , \ otherwise \end{cases} \tag{7}$$

where $NN_k(\phi^{'}, \Phi)$ gives k^{th} nearest feature vector of $\phi^{'}$ from Φ.

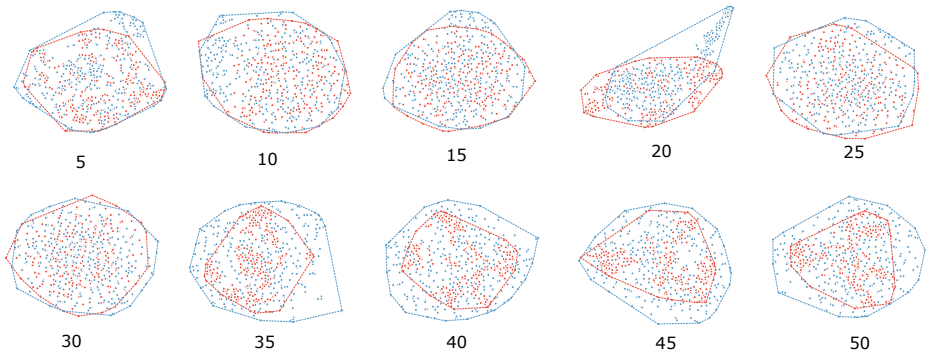

Fig. 12. Real (red) vs fake (blue) manifolds of 300 airfoils formed by extracting features from the last encoder layer (after reparameterization) of β-VAE model trained with different latent dimensions. The manifolds are dimensionally reduced to two dimensions using t-SNE. For visualization purpose, the convex hull of the two manifolds is also shown to better understand their overlap. For lower dimensions (upto 30), the real and fake manifolds are almost the same except for the latent dimension of 20. For higher latent dimensions (35 and above), the fake manifold encompasses the real manifold that shows that the generated samples are away from the real samples. Matching between real and fake manifolds highly correlate with the density and coverage scores from Figure 4.

With the help of Eqn. 7, precision and recall are calculated using following equations :

$$precision(\Phi_g, \Phi_r) = \frac{1}{|\Phi_g|} \sum_{\phi_g \in \Phi_g} f(\phi_g, \Phi_r) \tag{8}$$

$$recall(\Phi_g, \Phi_r) = \frac{1}{|\Phi_r|} \sum_{\phi_r \in \Phi_r} f(\phi_r, \Phi_g) \tag{9}$$

Using Eq. 8 and 9 we can calculate precision and recall respectively.

Density and Coverage Let x and y be real and generated samples respectively and $B(x, r)$ be the sphere in $\mathbb{R}^d$ around x with radius r, d be the dimension of x and y, $NND_k(x)$ be the k neighborhood spheres of x of k nearest neighbors and N and M be the number of real and fake samples respectively. Intuitively, density measures the quality of generated samples, and coverage measures the diversity in generated samples by comparing the real and fake manifolds. These manifolds are formed by forming spheres around their k nearest neighbors.

Mathematically, density calculates the number of real sample spheres containing generated samples and is calculated using Eq. 10:

$$D = \frac{1}{kM} \sum_{i=1}^{M} \sum_{j=1}^{N} 1_{y_i \in B(x_j, NND_k(x_j))} \tag{10}$$

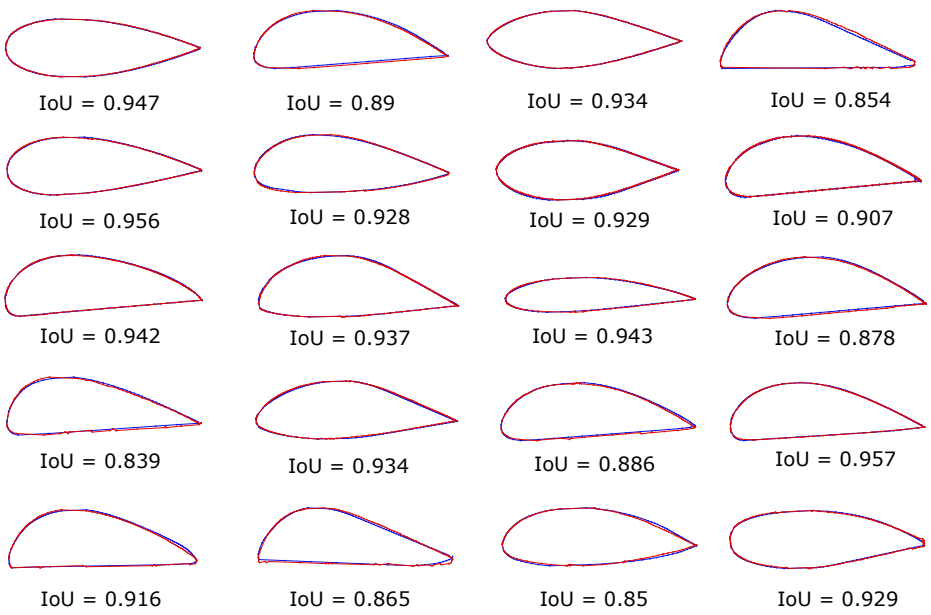

Fig. 13. Original airfoil contours (blue) vs reconstructed airfoil contours (red) and the IoU between them. The original and reconstructed airfoils are very similar and have high IoU.

Coverage measures the fraction of real samples whose neighbourhoods contain at least one fake sample and is given by Eq. 11.

$$C = \frac{1}{N} \sum_{i=1}^{N} 1_{\exists j \, s.t. \, y_j \in B(x_i, NND_k(x_i))}, \tag{11}$$

Density is lower bounded by zero but may be greater than one, but coverage is bounded between zero and one.