Supplementary Information

Hydrodynamic simulations

The fluid flow around an object is theoretically described by the Navier-Stokes equations,

$$\rho\frac{\partial \boldsymbol{v}}{\partial t} + \rho(\boldsymbol{v}\cdot\nabla)\boldsymbol{v} = -\nabla p + \eta\nabla^2\boldsymbol{v}, \tag{7}$$

where p denotes the pressure and η the viscosity coefficient. Equation 7 describes a second order non-linear partial differential equation and is except for a few special cases only numerically solvable. Here we consider the laminar (i.e. non-turbulent) flow of an incompressible fluid that obeys

$$\nabla\cdot\boldsymbol{v} = 0\,. \tag{8}$$

The resistance of an object in fluid dynamics is characterized by the drag coefficient c_d,

$$c_d = \frac{2F_d}{\rho v^2 A}. \tag{9}$$

In the reference frame of the object, F_d is the component of the force caused by the fluid in flow direction which represents drag. ρ is the density of the fluid, v the flow velocity and A is the size of the object, such as area or diameter in flow direction. Strictly speaking, Equation 9 does not hold true for laminar flow but c_d still provides a quantitative account of the drag in laminar flow if v is kept constant throughout all simulations.

Figure 4 shows example simulations from the dataset generation. We generate random shapes by picking radii from a uniform distribution at different polar angles. Subsequently we use a set of Fourier descriptors to obtain a smooth shape which facilitates the geometry meshing. The discretized space allows us to solve the Navier-Stokes equations (Eq.7) with a standard finite element solver (QuickerSim, MATLAB). The left wall is defined as fluid inlet with constant flow profile directed in positive x-direction. Furthermore we apply slip boundary conditions to the upper and lower wall (i.e. $v_y = 0$) and consider slip at the object boundaries. In analogy to the three dimensional case we define size A as the maximum object extension perpendicular to the flow direction. All simulations are performed with a scalar viscosity $\nu = 0.02$. Generated shapes are resized to 42×56 pixels to reduce memory requirements.

Shape dataset: Implementation details

We map the input images of size 42×56 to the latent space (25 dimensions) and back via a convolutional autoencoder. The encoder consists of two convolutional and one dense layer (200 units, ReLu) and we model mean and variance of the latent Gaussian distribution with two separate networks. The decoder consists of two dense layers (200 units each, ReLu) followed by two deconvolutional layers. The final deconvolution network uses sigmoid activation. We use a simple rounding operation to extract binary shape masks which we can then use as input for the hydrodynamic simulation. The discriminative network mapping z to a 20-dimensional feature space consists of two fully-connected layers (250 units each, ReLu). From here a linear output layer maps to the scalar y. The adversarial network consists of one hidden layer (100 units, ReLu activation). The size of the training set varies across experiments. For training in the drag optimization experiment we consider 2500 labeled and 2000 unlabeled shapes, respectively. Validation and test set consist of 300 samples each. We use early stopping with regret 2 based on the validation-set loss. The models are trained in TensorFlow with a learning rate of 0.0001 for all experiments. We use the package GPflow (Matthews et al., 2017) to fit Gaussian process models.

Protein dataset: Implementation details

Encoder and decoder are parametrized as fully-connected networks with one hidden layer, ReLu activation function and 200 hidden units. The discriminative network mapping z (60 dimensions) to a 20-dimensional feature space consists of two fully-connected layers (250 units each). From here a linear output layer maps to the scalar y. The size of the training set varies across experiments. Validation and test set consist of 2,000 labeled sequences each and we use early stopping with regret 2 based on the validation-set loss. For training in the Bayesian optimization experiment we use the entire dataset except for the validation and test set. We ensure that generated designs are valid, discrete structures by choosing the nearest neighbor. The learning rate is set to 0.0001 for all experiments. Computations are performed in TensorFlow and using the software package GPflow (Matthews et al., 2017).