# Using machine learning to augment artistic creativity

There has been a long running debate about whether machines can be truly creative.  While this debate probably doesn’t have an end, there is no doubt that the artistic processes can make use of technology, whether through better brushes for painting, or 3D animation software. Recently there has been growing interest in how machine learning in general, and neural networks in particular could be used to generate art.

This article will explore a “Generative Relational Context “(GRC) neural network to augment artistic expression and show how this approach is different from most current approaches to creating art using machine learning. Instead of relying on a black box model and vast quantities of training data, the GNC start with a set of artist created or identified samples. These samples provide the starting point for training a generative model to create a space of artistic possibilities that can be explored under artistic control.

This approach can be thought of as a kind of gamification of art through the combination of machine learning and artistic sampling and exploration, that redefines the role of the artist in machine learning approaches to art. A simple “low poly” 3D model of an opening flower petal will be used as an example.

## How machine learning has been used to generate art

Initial attempts at generating art with machine learning have taken several different forms.  One obvious approach is to automate the generation of art by training a network using existing works of art as input. Then the network can be used to generate outputs different that the training data.  By far the most popular techniques are variations on the generative adversarial networks like [GAN](https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29), [CAN](https://arxiv.org/pdf/1706.07068.pdf), [DCGAN](https://towardsdatascience.com/deeper-into-dcgans-2556dbd0baac), [etc](https://towardsdatascience.com/must-read-papers-on-gans-b665bbae3317). This can be done for literary texts, musical sounds, visual representations of images etc. Inputs for culinary recipes, perfumes or dance can also be imagined.  The immediate challenges then become, how to represent the inputs as numbers, how to structure the algorithm and how to define an error signal and loss function the steers the training towards what would be recognized legitimate works of art. As you can imagine, there have been many awkward as well as “interesting” initial attempts.

Some of the most publicized attempts have included:

* generating text from existing works of prose
* generating images of faces for people that do not exist
* generating music that mimics a composer’s style
* taking an existing painting or photograph and transforming it into a different visual style.

Instead of starting with the technical issues of these approaches, I would first like to address the role of the artist and where creativity is recognized and assigned.

## The role of the artist in machine learning

For an individual painter, who has mastered the use of paint and paint brushes, the source of creativity in artistic expression is unmistakable. However, with many artists increasingly using a team of assistants or fabricators, the source of creativity can become more ambiguous. For 3D animation a large team of many different specialists may be employed in a production. Usually the directing artist is given the creative credit, even if he was only tangentially involved in the production of the work. In the extreme, with “found object” art, the creativity is simply in the recognition of an object as art, or the choosing of what is art without any actual production of the work by the artist.

Looking at machine learning through the lens of automating the role of the art assistant or fabricator, creativity would seem to stay with the artist. However, many current approaches to art in machine learning tend to diminish the role of the artist. The allure of creating a machine that has true creativity is often considered the holy grail of AI art. In a similar way to what Jaron Lanier described for machine learning approaches to language translation, machine learning in art usually uses “big data” to process thousands, millions or billions of input samples into an algorithm that uses some emergent process of interpolation and extrapolation to generate a result. This creates a kind of Star Trek Borg Cube that assimilates vast quantities of material and erases their source identity, transferring the assignment of creativity to an emergent black box. This can be seen as promoting a mechanical assistant until the “student becomes the master” and the artist is deprecated and replaced with an algorithm.

## Supporting creativity

The problem with this vision, is that it still depends on real artists for its input, even if their identity and importance is obscured. The second problem is that there is no one to drive the process of creating value, which is always holistic and subjective in nature. To quote from Jaron Lanier, “The distinction between first-order expression and derivative expression is lost on true believers in the hive. First-order expression is when someone presents a whole, a work that integrates its own worldview and aesthetic. It is something genuinely new in the world. Second-order expression is made of fragmentary reactions to first-order expression.” [<https://medium.com/@ramurrio/the-future-of-content-eecc45d46c2a>

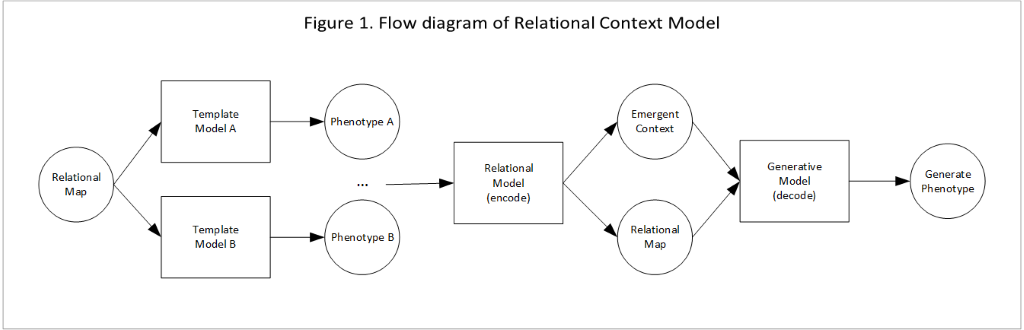
There is also another more technical problem with these kinds of algorithms, usually identified as the “The curse of dimensionality”.  Organic form and much of artistic expression is characterized by a large number of features that are coordinated as a whole, and that all depend on each other in a complex web of many to many relationships. Changing one point requires the adjustment of a large part of the rest of the form in order to maintain a functional or artistic coherence. As a flower petal opens and expands, every point on its surface must track with every other point in its shape, to maintain a holistic pattern. To use the example of 3D CGI modeling, as the complexity of an object’s shape increases (requires more coordinates), the complexity of possible 3D coordinate combinations increases exponentially, and the percentage of those combinations that represent coherent shapes diminishes exponentially.

Even more importantly, trying to extrapolate or interpolate between these values will become exponentially useless.  “In high dimensions, almost all of the data is in the outer shell. This means, among other things, that ‘neighborhoods’ typically stretch to the outer edges of multiple dimensions, that machine learning models almost always need to extrapolate, and that few other data points will be “similar” to any given data point.” ([Aaron Lipeles](https://towardsdatascience.com/@aaron.lipeles?source=post_page-----f07c66128fe1----------------------)) <https://towardsdatascience.com/the-curse-of-dimensionality-f07c66128fe1> The mathematical details of this are outside the scope of this article, but the ability of machine learning to scale to increasing number of features is often overlooked by starting with problems where an unlimited amount of data can be simulated, for example in games, or by using relatively simple examples, with ambiguous “artistic” results.

One way to overcome these problems is to reinstate an actual artist as the coherent source of input, and then have an algorithm support the development of a space of new creative possibilities that is dense enough to be navigated using an artist’s subjective judgments, without getting lost in the vast empty space of the “curse of dimensionality”.

## Machine learning as artistic augmentation

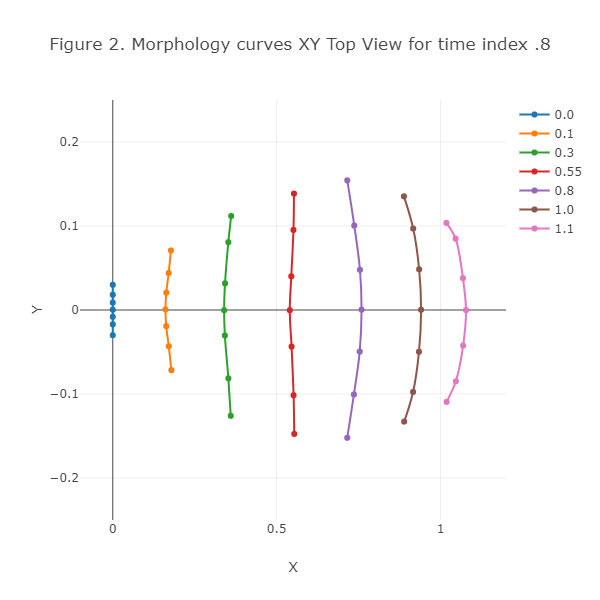
To address these problems a Relational Context Model shown in Figure 1. will be presented. **While most current generative approaches use neural networks as a black box to distill vast quantities of sample data without the guidance of an artist, GRC will instead try and more directly support and augment the creative expression of an active artist.** There will be three steps to this process. First reference content is created or identified by an artist, and captured in neural networks that will be call Template Models. Then two or more Template Models are used to train a Relational/Generative Model. This generative workflow has two parts, an encoding “Relational Model” and a decoding “Generative Model”.



## Artist generated templates

To illustrate the process, 3D coordinates for a computer graphics imagery (CGI) **mesh** will be used, similar in shape to the opening of a flower petal. Each petal represents a point in time and consist of a 7 by 7 mesh of 3D (x,y,z) coordinates with a separate mesh/petal for each of the 7 indexed points of time. These time indexes are like movie frames, that progress from a common closed bloom to two different fully opened shapes. Shape coordinates are formatted for the neural network as a rank 3 tensor represented as a 3-dimensional array with a shape of (7,7,21). The first dimension represents 7 time indexes (each a petal at a moment in time), made up of 7 morphological steps, where each step represents one row of coordinates in the grid of a mesh. The 21 coordinates of that morphological step are arranged as 7 X-coordinates followed by 7 Y-coordinates followed by 7 Z-coordinates.

Figure 2 shows a 2D top view of the 3D coordinates for one of the petal objects (at one time index), with the actual data listed after the figure. Figure 3 shows the rendered shapes for 5 timesteps of each template, omitting the first and last offset timesteps. The offsets are used to model the curves beyond the modeled range in order to remove edge effects in learning.



tensor( [[-0. , 0. , -0. , 0. , -0. , -0. , -0. , -0.03 ,

-0.017, -0.008, 0. , 0.009, 0.018, 0.03 , -0. , 0. ,

-0. , 0. , 0. , 0. , 0. ],

[ 0.179, 0.171, 0.163, 0.161, 0.163, 0.171, 0.178, -0.072,

-0.043, -0.019, 0.001, 0.021, 0.044, 0.071, 0.011, -0. ,

-0.012, -0.018, -0.012, -0.002, 0.012],

[ 0.36 , 0.353, 0.342, 0.34 , 0.343, 0.353, 0.361, -0.126,

-0.081, -0.03 , -0. , 0.032, 0.081, 0.112, 0.029, 0.008,

-0.019, -0.028, -0.02 , 0.009, 0.03 ],

[ 0.554, 0.552, 0.546, 0.54 , 0.545, 0.551, 0.553, -0.148,

-0.101, -0.044, -0. , 0.04 , 0.095, 0.139, 0.04 , -0.002,

-0.03 , -0.042, -0.028, 0.003, 0.034],

[ 0.716, 0.736, 0.754, 0.759, 0.754, 0.737, 0.716, -0.152,

-0.101, -0.049, 0. , 0.048, 0.101, 0.154, 0.028, -0.008,

-0.038, -0.052, -0.038, -0.007, 0.028],

[ 0.889, 0.916, 0.934, 0.941, 0.935, 0.917, 0.888, -0.133,

-0.098, -0.05 , 0. , 0.048, 0.097, 0.135, 0.009, -0.022,

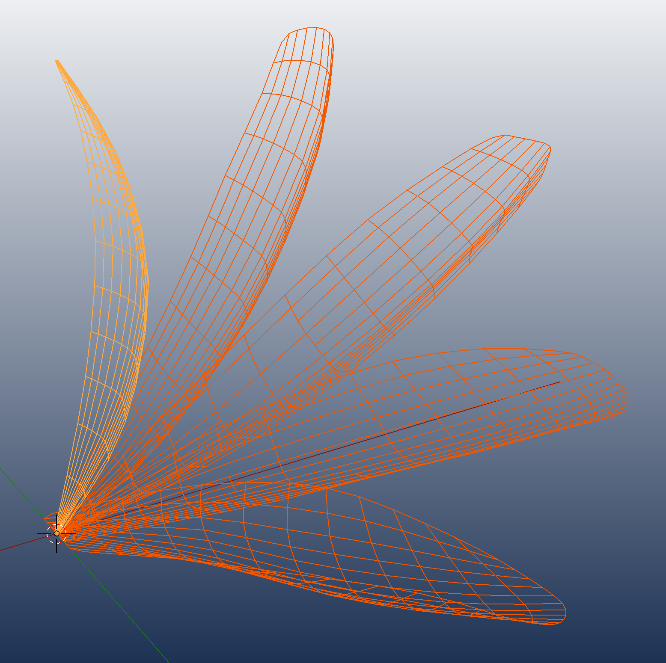
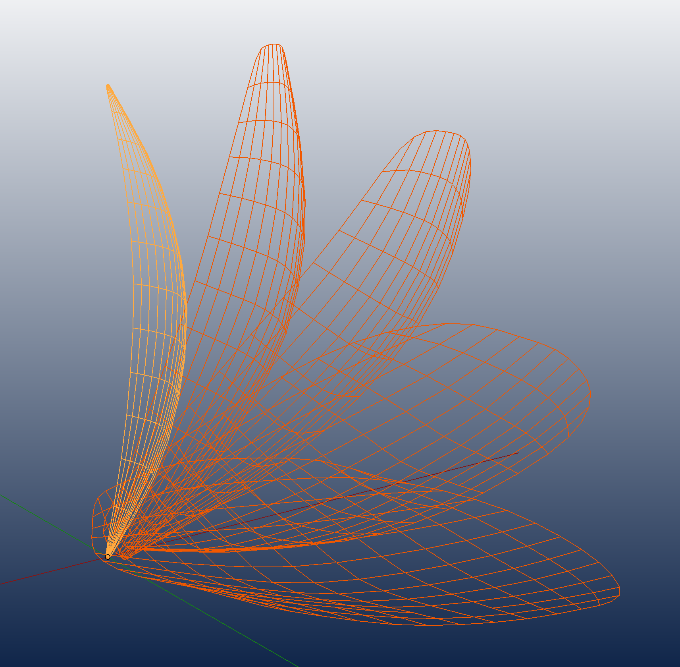
-0.041, -0.056, -0.042, -0.022, 0.01 ],

[ 1.019, 1.047, 1.069, 1.078, 1.069, 1.046, 1.018, -0.109,

-0.085, -0.042, -0. , 0.038, 0.085, 0.104, -0.018, -0.028,

-0.036, -0.043, -0.036, -0.028, -0.022]

])Figure 3. Shows the rendered 3D mesh for two template shapes with SubSurf smoothing.



## Building the Template Models

Once the output shapes have been identified for the Template Models, the shapes are paired with elements of an input Relational Map. This Relational Map is designed to index a set of properties that unify the changes across the objects in some ordered way. In the current example, the input to the Template Model is a 2D relational map consisting of two index values ranging from 0 to 1 (with an offset to 1.1). These values create a rank 2 Tensor of shape (7,2) represented as a matrix (2 dimensional array), with the first index representing a time step and the second index representing a morphology step. For the shape in Figure 2 the relational map has a time index=.8 and a range of morphological steps as shown below:

tensor([[0.8000, 0.0000],

[0.8000, 0.1000],

[0.8000, 0.3000],

[0.8000, 0.5500],

[0.8000, 0.8000],

[0.8000, 1.0000],

[0.8000, 1.1000]

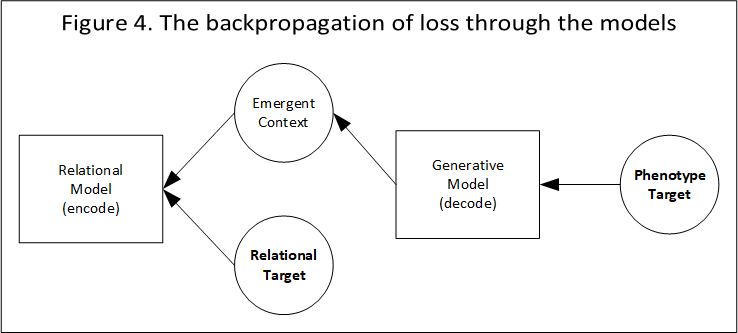
])

Each Template Model has 4 fully connected layers with shape [2, 10, 15, 21] that connect 2 values of a row in the input Relational Map to 21 values of a row of the output shape. This provides a mapping between the temporal/relational structures of an object that is independent of it shape, and the actual coordinates of the objects in physical space. This separation between the relational constraints that give organic form its holistic character and the rendering those relationships into a specific form, will be critical to the generative process. Once these template models have been trained using standard back-propagation techniques, they can be used to generate an unlimited amount of continuous valued training data for our Relational and Generative models.

## Building the Relational and Generative Models

The second step of the process, trains the Relational Model using training data from multiple Template Models to create an inverse transform of that data. This inverse transform maps 3D coordinate data from the different template models to their shared Relational Map. These relational map coordinates are then transformed back into 3D phenomenal space coordinates by a Generative model. T **This allows us to take a shape in 3D coordinate space and translate it into locations on a “relational map” in time/morphology space.** One problem that must still be addressed, is that all the templates use the same Relational Map, making its inverse into a single model ambiguous. This problem is usually solved by “tagging” or one-hot-coding the data with an index to disambiguate the data. Otherwise the Generative Model will average together the different output values for the same relational indexes.

However, instead of explicitly setting a template index as a target for the Relational Model output, the Relational Model has an additional context output, whose value will emerge from the training process. The Relational Map and Emergent Context value output by the Relational model, can then feed forward the information the Generative model needs to recover the 3D shapes of the template models. During training, the backpropagation of error moves back through the weights of the Generative model and then back through the weights in the Relational model, as shown in Figure 4. There is also a second flow of error from the Relational Map targets back through the Relational Model. This will allow the Relational model to differentiate the Emergent Context values necessary to make the relational to phenomenal (information preserving) transform reversible back to the original input shape.



If templates share the same 3D shape, for time and morphological indexes, then there is nothing to distinguish, and the context values will be the same, producing the same shape, from the same relational constraints.  As the 3D shapes diverge, the templates “trace” in context space will also diverge along different manifolds. Context points between the manifolds will interpolate between the different template shapes for a given relational map. If you project from the context average outward beyond the template context trace boundaries, you would expect to see that emergence of new shapes, diverging from the relational constraints of the templates. **If these emergent shapes are deemed to have artistic value, they could be used to identify new template models in an evolutionary bootstrap ecology of increased artistic value, diversity and complexity**.

## Using context values to explore the generation of new shapes

To see how the context values change across the relational map, we can input the reference 3D shape values used to train the templates, back into the Relational Model and observe the resulting context outputs.

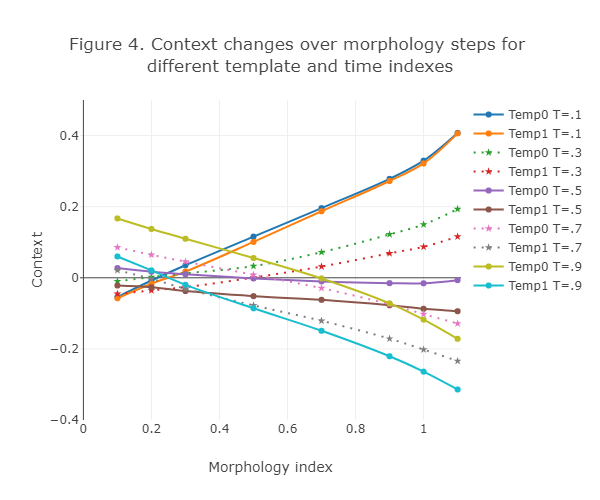


Figure 4. shows how the context values change across morphological steps for lines that represent different time indexes for the two Template Models. At time index .1 both templates model a closed petal with the same mesh. Since the mesh curves are the same, the context values are the same. As the petals expand over time, the meshes diverge and the context values separate.

This provide a model for shape generation that can be used to interpolate/extrapolate between the two templates. **However, in order to have any hope of finding artistically valuable new shapes, you need to project phenomenal (3D) values to a much smaller and compact “latent” space of coherent configurations.** There are two levels to this problem. First you need to be able to interpolate between coherent shapes, and second you want those shapes to be of as high artistic value as possible. This crosses over into a discussion of reinforcement learning and genetic algorithms, that will have to be deferred to another article. But it emphasizes the important of mapping phenomenal (3D) shapes to a compact representation that respects the implicit relational constraints of the forms. From this compact space it then becomes possible to identify context values that support high value exploration across that space. Instead of exploring in the space of phenomenal shapes, which quickly becomes computationally intractable, the relational morphological/time map that was created with the templates, provides a completely compact set of indices. Every point in the space represents a valid morphological stage at some point in time. The context values that emerges during the learning process can now be used as a kind of “game controller” like parameter to interpolate and extrapolate from training points in context space and projected out into 3D object space, in search of new shapes of artistic value.

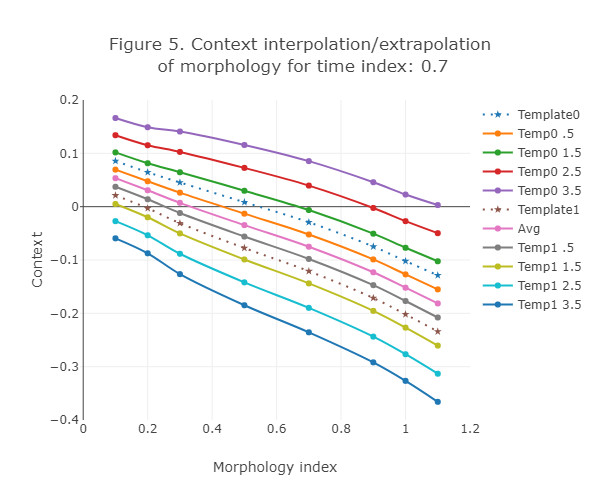


Figure 5. shows the context curves for the reference template shapes at time index 0.7 as dotted lines. Context values between the two reference curves (dotted lines) will interpolate between those two reference shapes when projected into a 3D space.

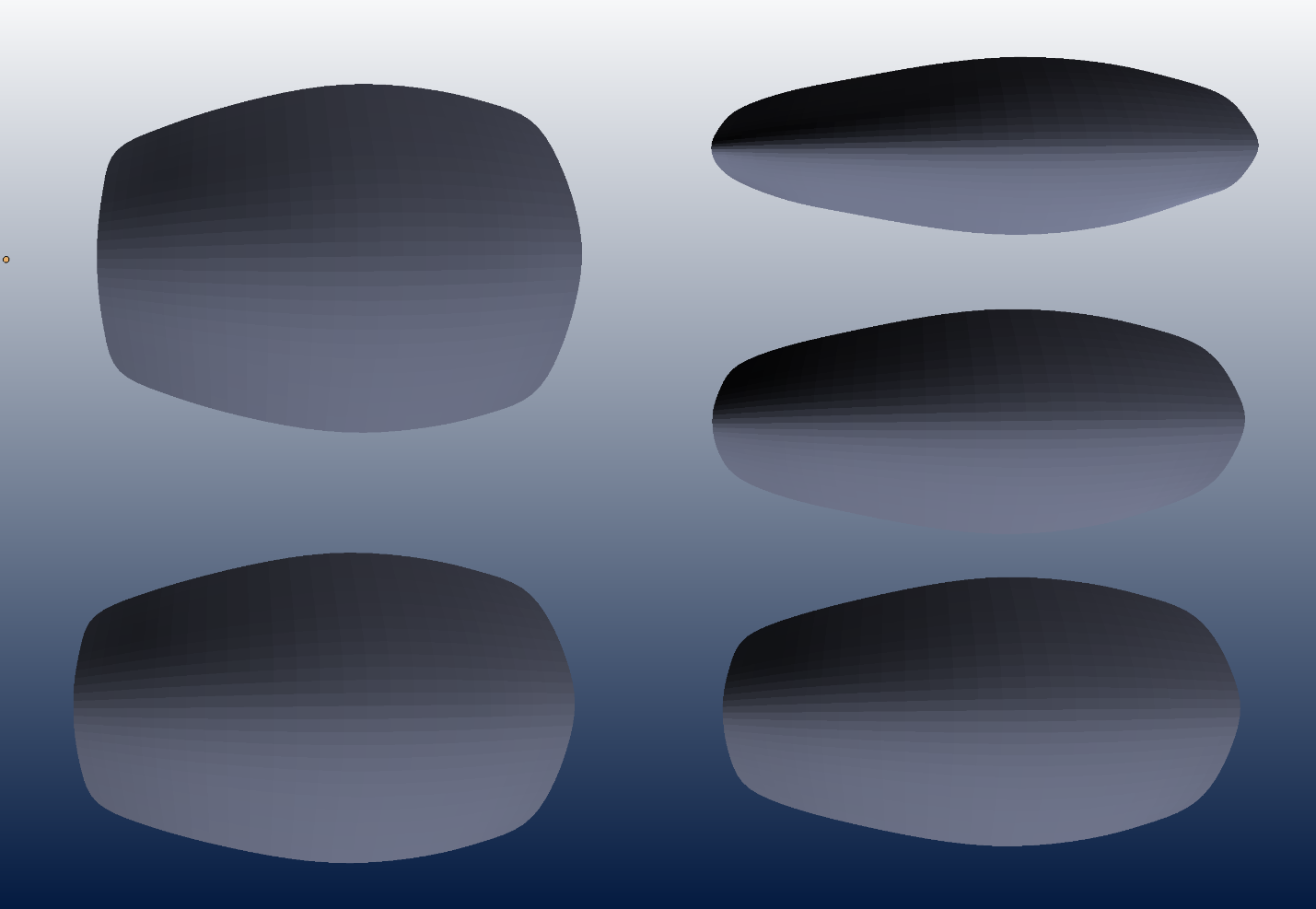


Figure 6. Top view 3D renderings of Generative model’s interpolation between reference context values

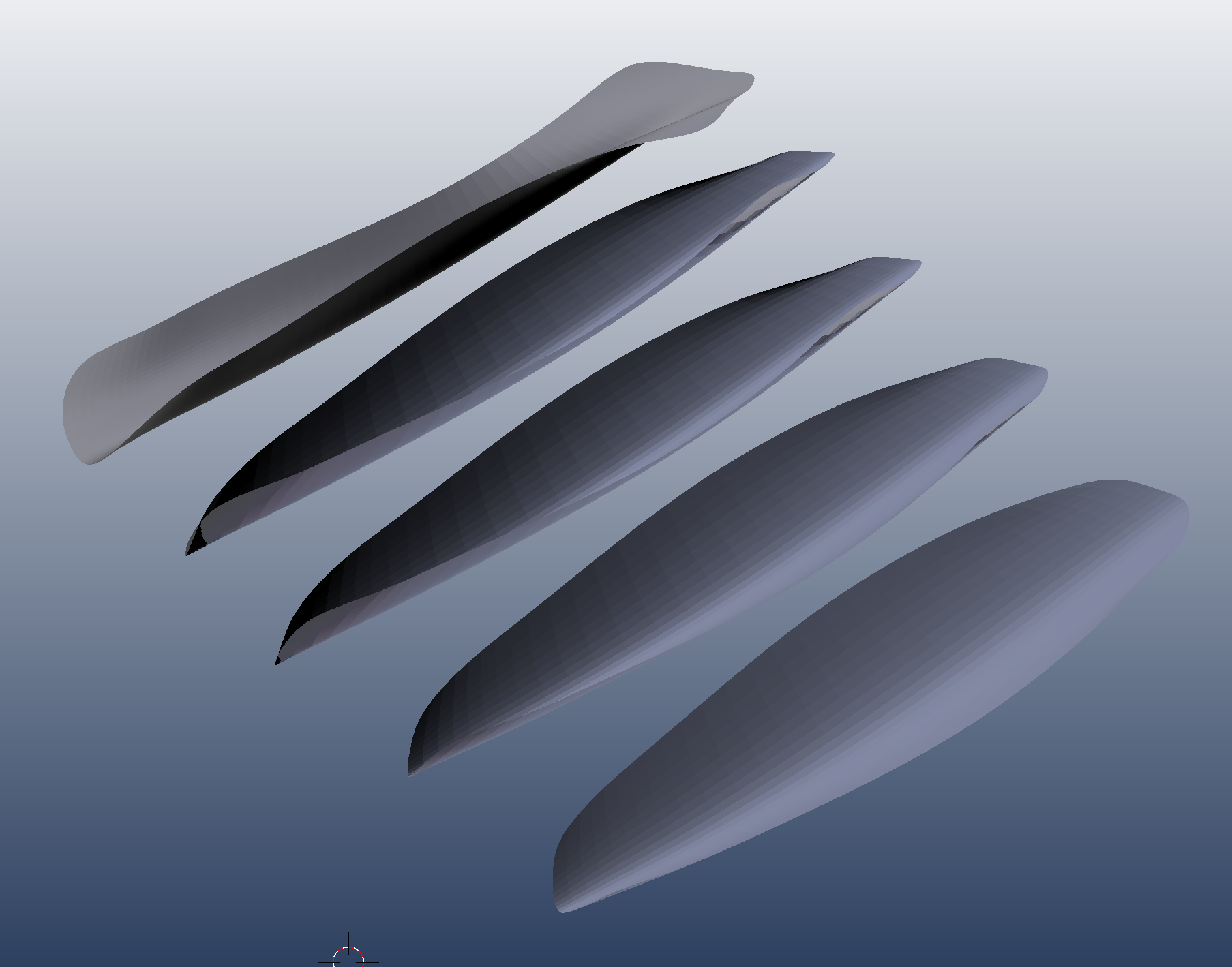


Figure 7. 3D renderings of Generative model outputs using an exploratory context path

The context lines outside the reference curves extrapolate beyond the constraints in the Relational Map. **This in turn allows the Generative Model to produce shapes beyond what it was trained to produce, while enforcing the relational constraints, to a diminishing degree**. Figure 7 shows one path of context values along the normal vector outside the second Template Model’s context values.

## Example code

In conclusion I would like to highlights some aspects of the code used in this article. While the code presents a fairly trivial example of generating a “low poly“ mesh to support a 3D model of an expanding flower petal. Hopefully it will allow for the clear presentation of the issues required to scale up to more complex models.  While a complete review of the code is outside the scope of this article, I want to highlight some more unique features of the model. The full code will be available on a GitHub repository to support a future article.

First, I want to show the network model that was used. One notable feature is the use of the ELU activation function that seems to work better for small problems like this, where large scale GRU performance isn’t an issue. For this problem the CPU actually ran faster than using the GPU. There are also a couple of extra parameters in the constructor to add size and descriptive information.

## A simple PyTorch Model structure:

class Net(nn.Module):

def \_\_init\_\_(self, layerSizes, name):

super(Net, self).\_\_init\_\_()

self.shape = layerSizes

self.name = name

self.seq = nn.Sequential(

nn.Linear(layerSizes[0],layerSizes[1]),

nn.ELU(),

nn.Linear(layerSizes[1],layerSizes[2]),

nn.ELU(),

nn.Linear(layerSizes[2],layerSizes[3]),

nn.ELU()

)

def forward(self, x):

return self.seq(x)

## Model initialization and parameters:

tmModel0 = Net([2,10,15,21],'First template model')

tmModel1 = Net([2,10,15,21],'Second template model')

path=CURRENT\_MODEL\_SAVE\_DIR+'templateModel'

tmModel0.load\_state\_dict(torch.load(path+'0.pth'))

tmModel1.load\_state\_dict(torch.load(path+'1.pth'))

tmModel0.eval()

tmModel1.eval()

relModel = Net([21,15,10,3],'Relational Model with 1 Context output')

genModel = Net([3,10,15,21],'Generative Model with 1 Context input')

relModel.train()

genModel.train()

parms={'epochs':100000,'lrR':.0001,'lrG':.0002,'numSamples':1,

'rFilter':torch.Tensor([1,1,.05]), 'tModels':[tmModel0,tmModel1],'device':device}

## Training loop

For the training loop there is a lot more going on. The function gy.createRcGenTargetsFromModelList creates targets for the Relational Map and 3D coordinate target values for the Generative Model that are sampled from randomly weighted values of the two Template Models. A more notable issue that was not discussed in the article, is how gradient information is passed back through the Generative and Relational Models. In PyTorch this requires that you use “retain\_graph=True” when moving back through the Generative model. You also want the Generative model gradient to influence the context value without disrupting the relational map of the Relational model targets. Trying to do this in a high-level framework like PyTorch turned out to be challenging. After many hours trying to develop a custom autograd function, I hit on a much simpler solution. I simply mask the context value passed into the loss criterion of the Relational model, so that its value is determined by the error gradient from the Generative model. A small .05 amount of the gradient was allowed to pass through the mask in order to help scale the context value.

criterion = torch.nn.MSELoss(reduction='sum')

relOptimizer = torch.optim.Adam(relModel.parameters(), lr=lrR)

genOptimizer = torch.optim.Adam(genModel.parameters(), lr=lrG)

for epoch in range(epochs):

#Each epoch gets a new training set based on the template models

relTarget, genTarget, \_ = gy.createRcGenTargetsFromModelList(tModels,

numSamples, numContextDims)

genOptimizer.zero\_grad()

relOptimizer.zero\_grad()

relOutput = relModel(genTarget)

genOutput = genModel(relOutput)

lossG = criterion(genOutput, genTarget)

lossR = criterion(relOutput\*rFilter, relTarget\*rFilter)

(lossG).backward(retain\_graph=True)

(lossR).backward()

genOptimizer.step()

relOptimizer.step()

## Future Work

The buried lede of the article is in figure 5, where context values are shown to define the boundaries of a latent volume in context space of coherent phenomenal forms. One possible avenue of future research is to explore how these context values could be used to develop a coherence gradient across context space. This gradient would provide a kind of compass signal for exploratory actions outside that coherent volume, back to the border of a known region of coherence. Another avenue of future research might to use the stability properties of this more compact emergent representation of state as a basis for the accumulation of value estimates and reinforcement learning. The suggested metaphor, is of an artist disrupting his work in impulsive ways and then bringing the work back into focus at greater levels of artistic complexity. Another metaphor might be an ecosystem disrupted by a series of disasters that continually reform at greater levels of complexity, seemingly against the forces of entropy. For fans of the singularity, the second Terminator Movie or the second of the 12 labors of Heracles, you will have to wait for additional experimentation before I can comment.