30 pm

# Learning Latent Mappings for Engineering Optimisation

*Using artificial neural networks to learn representations for solving constrained optimisation problems*

Alexander Atack 27745449

supervised by **Dr. David Toal**

## 1 Motivation

In computational design, it is often necessary to find a solution that maximises an objective function while simultaneously satisfying one or more constraints. Multiple optimisation algorithms exist to solve this problem, but their usefulness is limited when the constraints invalidate a large proportion of solutions, creating multiple discrete modes of potentially viable solutions. If a latent space could be defined, and a mapping from the latent to the solution space learned such that any point in the latent space satisfies the constraints when mapped into the solution space, the problem could be simplified. Standard optimisation algorithms might then practically be used in the latent space to optimise engineering problems for which they were previously too slow.

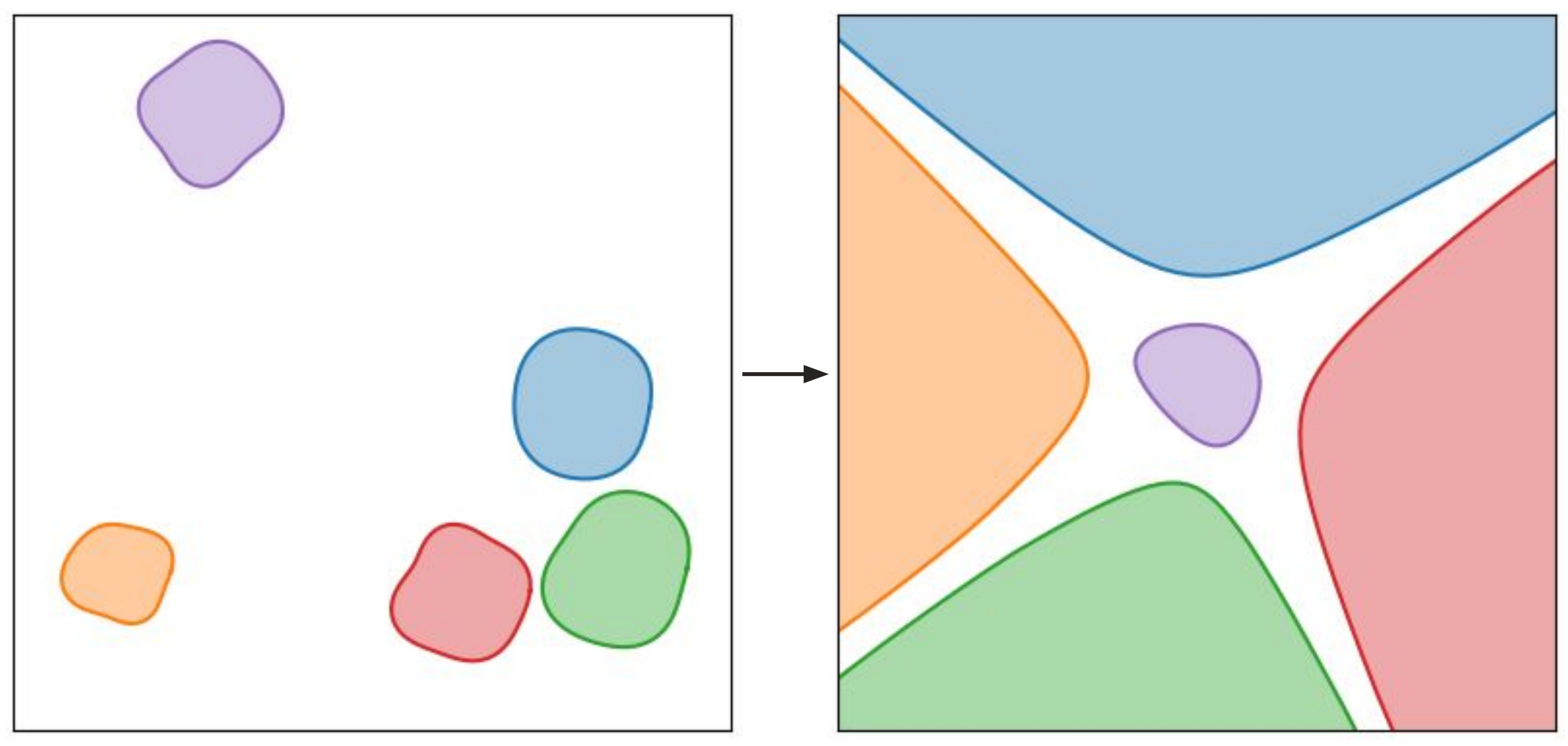

*Figure 1: Example of how discrete modes of viable solutions in the solution space (L) might be brought closer together in the latent space (R), accelerating exploration of the solution space when traditional optimisation algorithms are used in the latent space.*

It is hypothesised that artificial neural networks (ANNs) could be used to learn the mapping, referred to as the generator function, because of their uses as universal approximators [1]. The focus of this project was developing a neural network architecture and accompanying training algorithm to learn the latent mapping described above.

## 2 Architecture

Generative adversarial networks [2] are capable of learning to sample from spaces. The architecture used here was an adaptation of the GAN architecture consisting of three connected ANNs: the discriminator, which predicts whether or not a constraint will be satisfied and is assumed to be already trained from data; the generator, which maps points from the latent space to solution space; and the embedder, which allows parameterisation of the latent mapping by the relevant constraint by altering the weights and biases of the output layer of the generator.

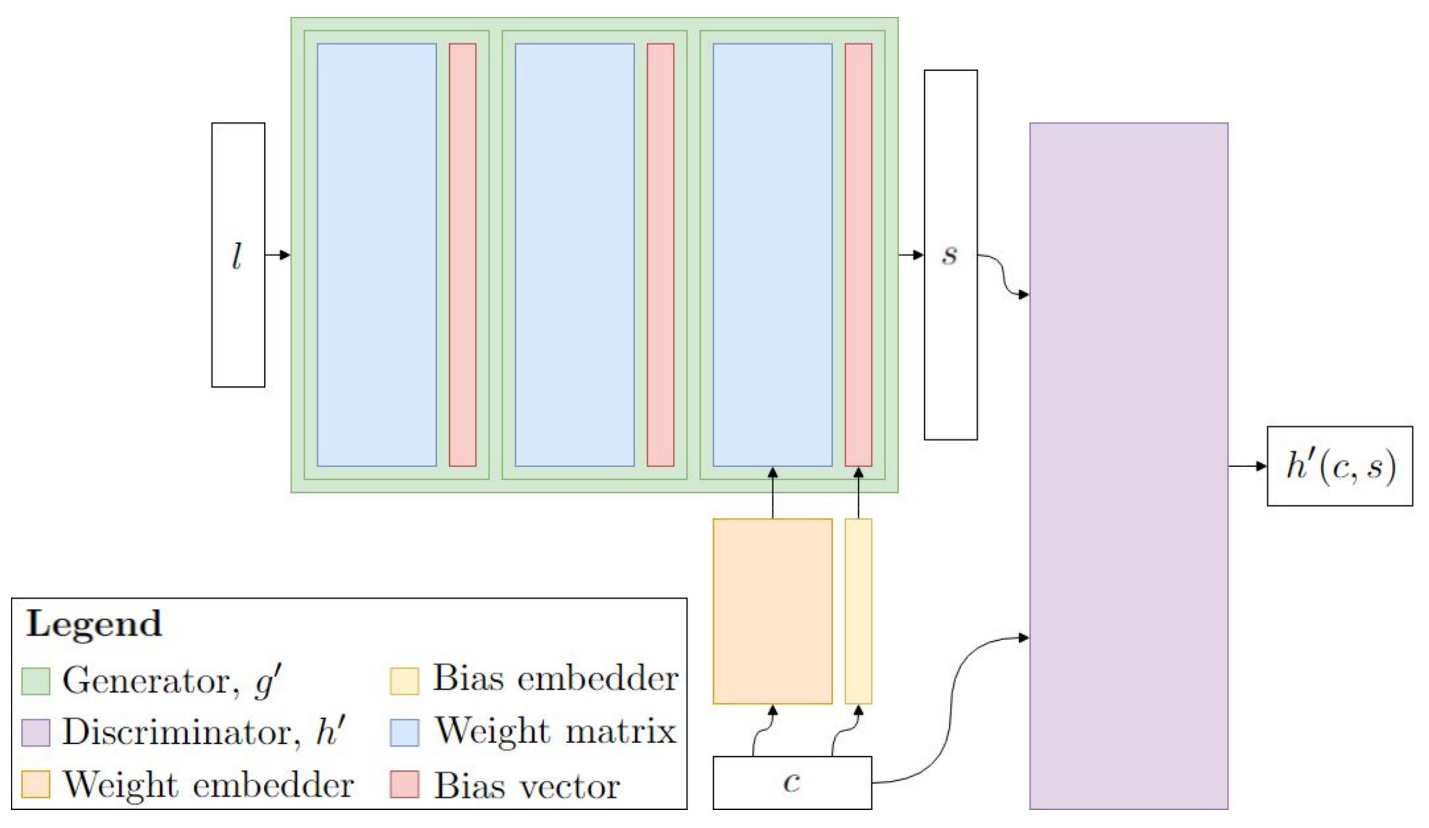

*Figure 2: Modified GAN architecture used to train the learned viable distribution to match the true viable distribution.*

During training, the weights of the discriminator remained fixed while those of the generator and embedder were updated in the normal way through backpropagation.

## 3 Training

Because ANNs are continuous by definition, the space of values obtained by feeding points from the latent space through the generator - the learned viable space - will always be unimodal. To allow the generator to approximate multimodal solution spaces such as that in Figure 1, a fuzzy approach was adopted wherein spaces were modelled as probability distributions: a higher probability density was interpreted as a higher probability of satisfying the constraints. The learned viable distribution is the distribution of solutions obtained by sampling uniformly from the latent space and passing the samples through the generator, and a weighted sum of two similarity measures - precision and recall - between the learned and true viable distributions was optimised.

$$p(g') = \int_C \int_L 1_{g'(l,c) \in V_c} \, \mathrm{d}l \, \mathrm{d}c \qquad r(g') = \int_C \int_{V_c} 1_{g'^{-1}(s,c) \in L} \, \mathrm{d}s \, \mathrm{d}c$$

Precision measures the probability that a sample drawn from the generator satisfies the constraints; recall measures the probability that a viable solution is capable of being sampled. While the precision can be estimated by taking samples from the latent space, directly estimating the recall was proven to be intractable. Instead, two proxies for recall were introduced, identity loss and separation loss, which penalised distributions not departing from a single mode in the solution space.

$$q_{\mathrm{id}} = \left| \frac{l - a_L}{b_L - a_L} - \frac{g(l,c) - a_S}{b_S - a_S} \right|^2 \qquad q_{\mathrm{sep}} = \left( q_{\mathrm{target}} - \frac{1}{b^2} \sum_{i,j=1}^{b} |B_i - B_j|^2 \right)^2$$

Identity loss encourages the generator to mimic the identity function and, as such, can only be used when the dimensionality of the latent space is equal to that of the solution space. Separation loss encourages the mean squared distance between any two sampled points to be equal to a particular target separation.

## 4 Equality constraints

To test the ability of the generator to learn to sample from the possible solutions to equality constraints, a generator was trained on an environment in which the goal was to find the inputs to a function which produce outputs within a particular range. The upper and lower bounds comprised the arguments of the constraint vector. The function used in this case was an adaptation of the Branin function [3], rescaled such that the inputs and output were all between 0 and 1.

Samples were taken from the generator for three different cases initially with lenient constraints, but the final case with a more stringent constraint, on the bounds of the function. These are displayed in Figure 4. The rightmost case represents a viable space constrained to inputs for which the Branin function evaluates to a value within 1% of 0.5.

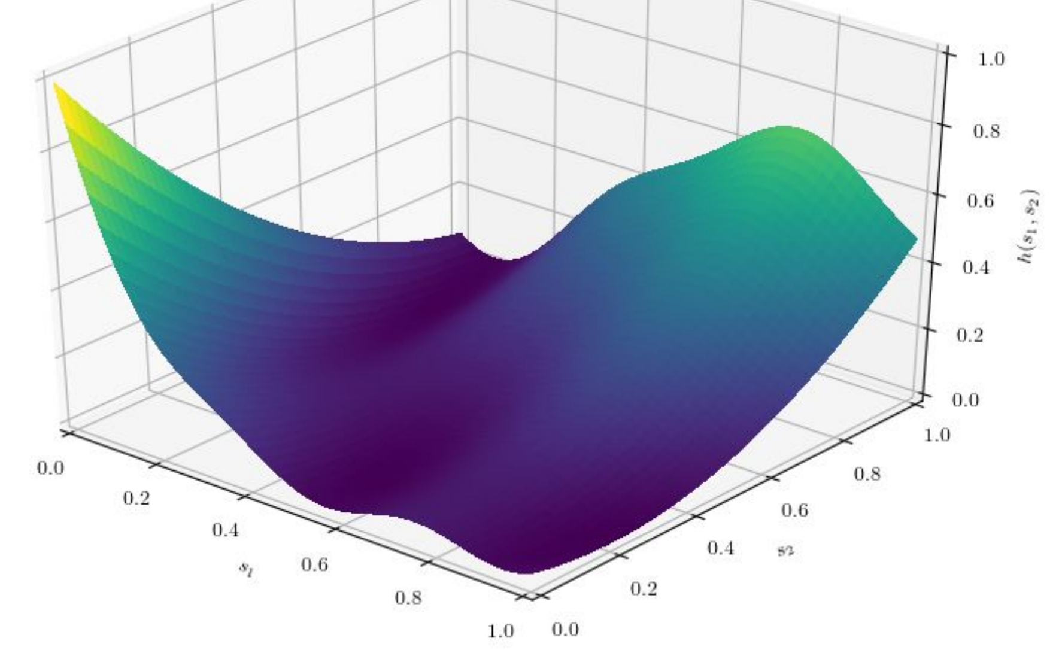

*Figure 3: Rescaled Branin function used to examine the ability of the generator to learn to sample from equality constraints.*

Generally the generator learned to sample predominantly from those regions that the discriminator classified as highly likely to satisfy the constraint. In the larger mode towards the top right, however, regions of viable solutions are frequently undersampled. In the most constrained case, most equivalent to an equality constraint, one mode is sampled almost perfectly. The other is sampled incorrectly: part of it is not sampled at all, while the other part has samples drawn from nearby but not actually within it. It should also be noted, however, that the predictions of the discriminator begin to deviate from ground truth in this region, suggesting that the model's parameters could be trained further to refine its mapping.

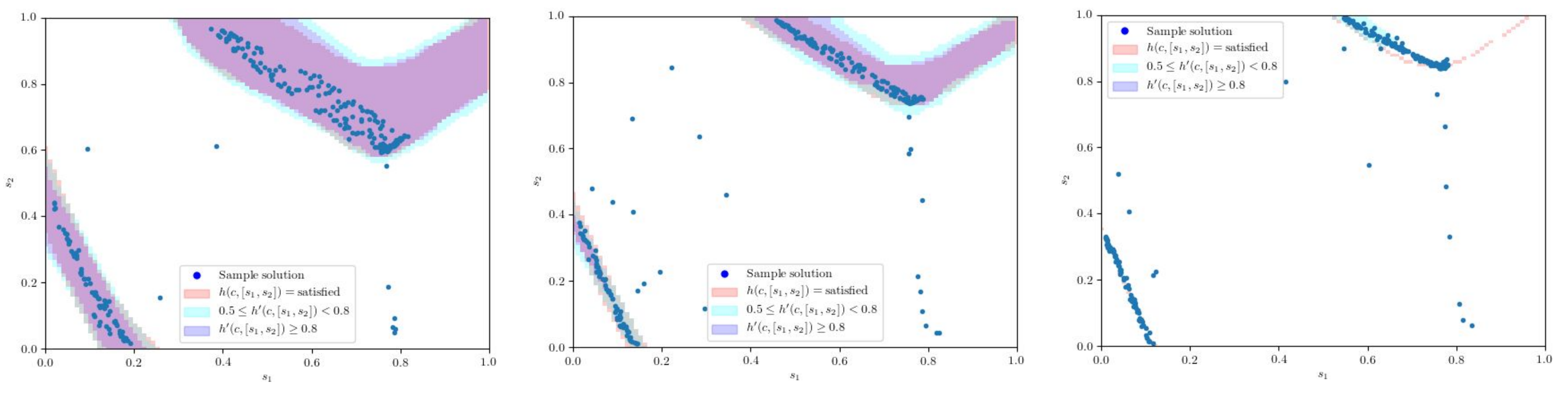

*Figure 4: Effect of increasingly stringent constraints on the samples taken from the latent space.*

## 5 Latent space visualisation

To gain an insight into how the generator constructs a mapping between the latent and solution spaces, the actual viable space was plotted for the same constraint vector in Figure 4 (C), along with constant lines for each latent variable, in the solution space (Figure 5, L). The effect of the mapping is clear to see, with the density of the constant lines increasing significantly around the solutions that satisfy the constraint. In between the two modes is an area of significantly reduced density.

The same data were also plotted in the latent space (Figure 5, R). By comparing the two, it is possible to see whether the latent mapping had the desired effect of pulling together distant modes as described in Figure 1.

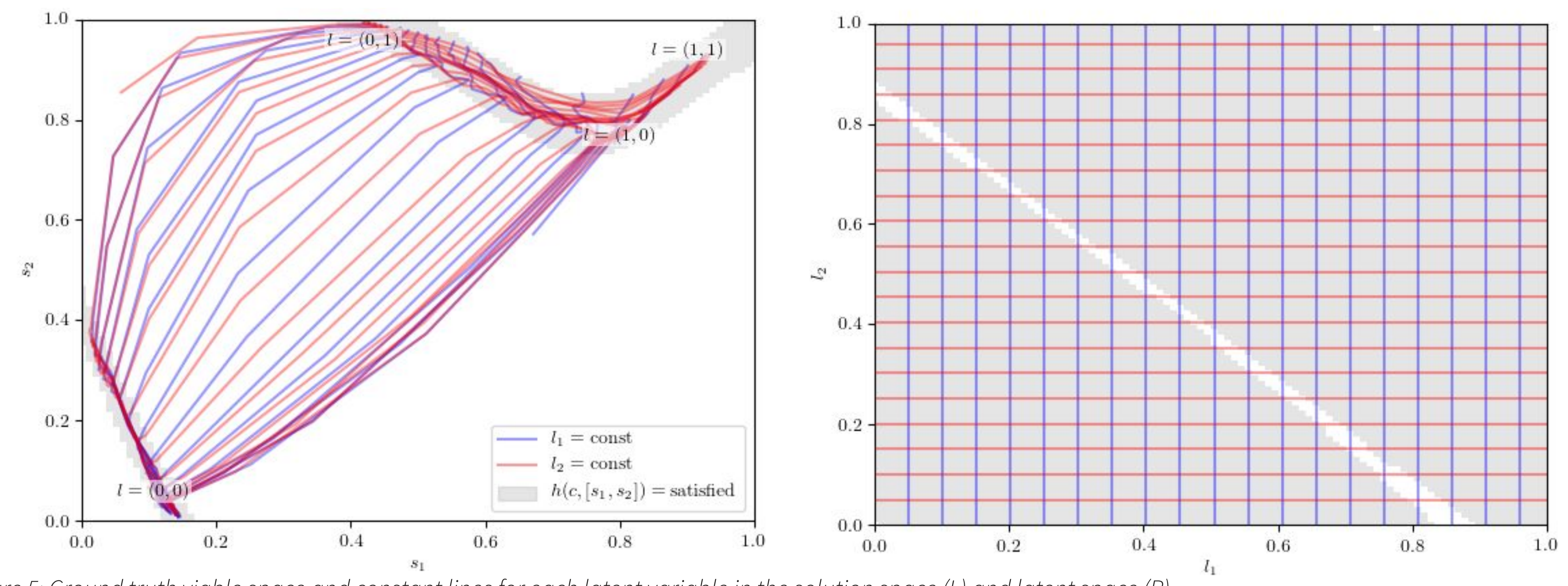

*Figure 5: Ground truth viable space and constant lines for each latent variable in the solution space (L) and latent space (R).*

With the drastic reduction in empty space between the two modes in Figure 5 (R), it can be concluded that the latent mapping achieved its goal of making optimisation over the space of viable solutions for an exploratory algorithm more tractable. Another item of interest is that the divide between the two nodes, though highly nonlinear in the solution space, is reduced to being almost perfectly linear in the latent space. While this is likely due to the use of quasi-linear ReLU activation nodes in the hidden layers of the generator, it is still an unexpected property that may be useful in exploitation of the latent space.

## 6 Conclusions and further work

An ANN architecture and training algorithm have been developed which demonstrate promising properties for learning latent mappings that simplify the task of using conventional optimisation algorithms in constrained engineering environments. The method has significant limitations, however, which may be overcome by further work. Primarily, the latent space often fails to capture all viable solutions, most likely because a proxy for recall is optimised instead of a direct estimation of the recall. More suitable proxies for recall than identity or separation loss might therefore be researched. Producing a direct and differentiable estimate of the recall of the mapping requires the generator network to be inverted, which is not currently tractable. It was shown, however, that inverting the generator would be feasible if an algorithm could be found for determining the intersection between two simplices. Nevertheless, ability of the mapping to quickly sample from a range of likely solutions to a constraint could prove useful in frequently encountered engineering environments such as topology optimisation.

## 7 References

[1] K. Hornik, *Approximation Capabilities of Multilayer Feedforward Networks*. Neural networks, 1991, Volume 4, Issue 2, Pages 251-257

[2] I. Goodfellow et al, *Generative Adversarial Nets*. arXiv:1406.2661v1 [stat.ML], 10 Jun 2014

[3] D. Bingham, *Test Functions and Datasets*. `https://www.sfu.ca/~ssurjano/branin.html` , accessed March 2019

aa7g15@soton.ac.uk

www.github.com/aatack/IP