AUTOMATED DESIGN USING NEURAL NETWORKS AND GRADIENT DECENT

Oliver Hennigh
Mexico
Guanajuato, Gto, Mexico
loliverhennigh101@gmail.com

ABSTRACT

In this paper, we propose a novel method that makes use of deep neural networks and gradient decent to perform automated design on complex real world problems. Our approach works by training a neural network to mimic the fitness function of the optimization task and then, using the differential nature of the neural network, we perform gradient decent to maximize the fitness. We demonstrate this methods effectiveness by designing an optimized heat sink and both 2D and 3D wing foils that maximize the lift drag ratio under steady state flow conditions. We highlight that our method has two distinct benefits. First, evaluating the neural networks prediction of fitness can be orders of magnitude faster then simulating the system of interest. Second, using gradient decent allows the design space to be searched much more efficiently.

1 INTRODUCTION

Automated Design is the process by which a object is designed by a computer to meet or maximize some measurable objective. This is typicaly performed by modeling the system and then exploring the space of designs to maximize some desired fittness function whether that be automotive car styling with low drag or an integrated circuit with small profile. The most notable historic example of this is the 2006 NASA ST5 spacecraft antenna designed by a evolutionary algorithm to create the best radiation pattern. More recently there has been (Flow Sculpter) and (optical computer). While there have been some significant sucesses in this feild the dream of true automated is still far from realized. The main challanges faced are heavy computational requirements for accuratly modeling the physical system under investigation and often exponentialy large search space. These two problems negatively complement eachother making the computation requirements intractable for even simple problems. For example, in the realively simple flow problems explored in this work, a heavily optimized flow solver running on modern gpus requires around 5 minutes to perform each simulation. Given that as many as 3,000 designs need to be tested to acheive reasonable performance, this results in a total computation time of 9 days on a single GPU. Increassing the resolution of the simulation or expanding the parameter space quickly make this an unrealizable problem without considerable resources.

Our approach works to solve the current problems of automated design in two ways. First, we learn a computationaly effeicent representation of the physical system on a neural network. This trained network can be used evaluate the quality or fittness of the design several orders of magnatude faster. Second, we use the differentiable nature of the trained network to get a gradient on the parameter space when performing optimization. This allows significanly more efficient optimization requiring far fewer iterations then other methods. These two abilitys of our method overcome the present difficulties with automated design and allow the previous mention 9 day optimization to be run in only 10 mins. While we only look at two relavively simple problems in this work, we enphasize that the ideas behind our method are applicable to a wide variety of automated design problems.

The first problem tackled in this work is designing a simple heat sink to maximize the cooling of a heat source. The setup of our simulation is ment to somewhat minimc the conditions seen in a heat sink on a computer processor. We keep this optimization problem realatively simple though and use this only as a first test and introduction to the method.

We also test our method on the significantly more difficult task of designing both 2D and 3D wing-foil with high lift drag ratios under steady state flow conditions. This problem is of tremendous inportance in many engineering areas such as aeornotical, aerospace and automotive engineering. Because this is a particularly challanging problem and often times unintuative for designers, there has been considerable work using automated design to produce optimized designs. We center much of the discussion in this paper around this problem because of its difficulty and view this as a true test our methods advantages and disadvantages.

As we will go into more detail in later sections, in order to perform our flow optimization tests we need a network that predicts the steady state flow from an objects geometry. This problem has previously been tackled here where they use a relatively simple network architeture. We found that better perform could be obtained using some of the modern network architecture developments. For this reason, in addition to presenting our novel method of design optimization, we also present this superiour network for predicting steady state fluid flow with nerual networks.

This work has the following contributions.

- We demonstrate a novel way to use neural networks to greatly accelerate automated design problems.
- We present this method in such a way that it can ready applied to many other automated design problems.
- We provide a new network architeture for predicting steady state fluid flow that vastly out performs previous models.

2 Related Work

This is touches on several vastly diffrent subjects. Because of this we have provided a breif review of the related areas.

3 Speeding up Computational Physics with Neural Networks

Our method revoles around being able to emulate

In recent years there has been incredable intrest in the application of neural network to computational physics problems. One of the main applications being to emulate the desired physics for less computation then the Physics simulation. Examples applications range from simulating 3D high energy particle showers seen here to

Our method uses a network to predict steady state fluid flow from boundary conditions. This idea was first presented in (Guo et al., 2016) where they showed. Our flow prediction network has several key difference to this original work. First, we heavily improve the network architecture by keeping the network all convolutional and taking advantage of both residual connections and a U-Network architeture. This proved to drasticaly improve accuracy while maintain fast computation. Second, it takes in the binary representation of the boundary conditions instead of the Signed distance map. We found that with our improved network architecture, we over came the issure in their work using such a representation of the boundary.

3.1 Automated Design Optimization of Airfoils

To data, there has been substantial work in automated aerodynamical design for use in aeronotical and automotice applications. The standard work flow is to first parameterize the search space of designs then iteratively simulate and evaluate the. We use the A veriety of search methods have been used with some sucess in the optimization process. Simulated Annealling, Genetic Algorithms and Particle swarm have all been used with vering degrees of sucess.

Automotive Aerodynamic Design Exploration Employing New Optimization Methodology Based on CFD

Adaptive Shape Parameterization for Aerodynamic Design

Pros and Cons of Airfoil Optimization 1

Multi-level CFD-based Airfoil Shape Optimization With Automated Low-fidelity Model Selection

A Surface Parameterization Method for Airfoil Optimization and High Lift 2D Geometries Utilizing the CST Methodology

A Universal Parametric Geometry Representation Method CST

4 GRADIENT DECENT ON PARAMETER SPACE

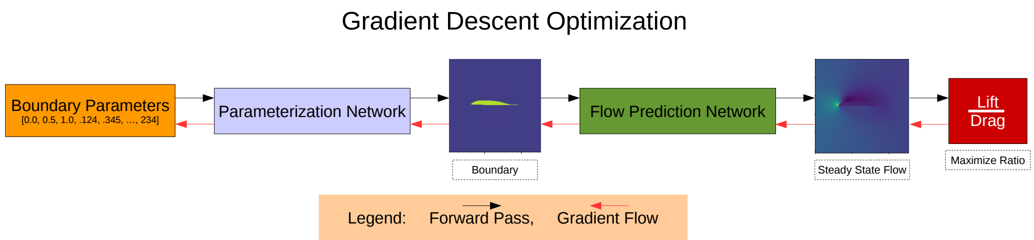

Figure 1: Illustration of Proposed Gradient Decent Method

Our automated design optimization problem can be viewed in concrete terms as maximing some desired fitness function $F(x)$ where $F : X \to \mathbb{R}$ for some space X of design parameters.

$$\max_{\forall x \in X} F(x) \tag{1}$$

In some real world setting, evaluating the fittness function F can be computationaly demanding as is the case with our fluid simulations. The first aspect of our work is to replace F with a computationaly effeicent neural network F_{net}. This can offer considerable speed improvements as we will disscuss bellow. The second aspect of this work is the observation that F_{net} is differentiable and provided that the parameter space X is real valued we can obtain a usable gradient in the direction of maximizing the fittness. This allows gradient decent to be performs where as in most setting F is not differentable and requires other search techniques such as simulated annealing or genetic algorithms. This allows faster optimization to be performed with fewer iterations as we will demonstrate bellow. The requirement of X to be real valued presents some chalanges though. To show case this and our solutions to them we go through the example problem of optimizing the fin height on a heat sink.

In our simple heat sink problem, X contains 15 real valued parameters between 0 and 1. Each of these parameters corrisponds to the height of an aluminum fin on the heat sink as seen in the figure. In our optimization problem we assume that there is a fixed amount of aluminum and scale the total lenghth of all the fins to meet this requirement. The presents an intereseting problem of deteremining the optimal length each fin should have to maximize the cooling of the heat source. The simplest application of our method would be to have a neural network to take in the 15 fin height values and output a single value corresponding to the tempuature at the heat source. This apporach has the draw back that if you want to add another aspect to the optimization like makeing sure the left side is cooler then the right side you would need to retrain the network. Another solution is to have the network again take in the fin parameters but output the full heat distrobution of heat sink. This allows diffrent quantitize to be optimized but is still limiting in that our network only runs on a single parameter set up. Our solution to this problem is to train two networks. The first network, P^{heat}_{net}, takes in the fin parameters and generates a binary image corresponding to the geometry of the heat sink. We refer to this as the parameterization network. The second network, S^{heat}_{net}, predicts the steady state heat distorbution from the geometry. Because the parameterization network is performing an extremely simple task and training data can be generating cheaply, we can quickly retrain P^{heat}_{net} if we want to change the parameter space. The same approach is used for the steaty state flow problem and a figure dipicting this can be found here. This approach allows our network to be as veratile as possible while still allowing it to used on many design optimization tasks.

Up until now we have not discussed how to generate data needed to train these neural neural networks. Generateing the data to train the parameterization network is realitively simple. If the paraeterization we are using is known we simply make a set of parameter vectors and there corissponding geometrys. In the case of the heat sink this is a set of examples composed of the 15 parameters and there corrisponding binary representation of the head sink. Putting together a dataset for S_{net}^{heat} or S_{net}^{flow} (fluid flow network) is somewhat more complicated though. The simplest solution and the apporoch used in this work is to simulate the respective physics on objects drawn from the object design space. For the heat sink problem this would entail a dataset of object geometrys and their corrisponding steady state heat distrobutions. This method has the disadvantage that the network only sees examples from the current parameter search space and if it is changed the network may not be able to accuratly predict the physics. We argue this is not a significate issue for several reasons. First, in the work seen here the network is able to generalize effectively to objects outside its train set. During the course of this work we tested our network on a veriety of datasets and found similar generaling abilities. Second, it is easy imagine a setup where a network is trained on a large set of diverse simulations and then finetuned on the current desired parameter space when desired. For these reasons we feel that this approach of generating simulation data is not significantly limiting and feel that employing some highbread approach would only distract from the underlying method presented.

4.1 Flow Prediction Network

The core componet of our method is being able to emulate the physics simulation with a neural network. For the steady state flow problem here has already been work doing just this found here. As mentioned above, we have made some improvements to this network architecture design. A figure illustrating our network can be found here. This model resembols the u-network architeture seen here with a series skip after each down sample. The advantages of this style of network are its high performance on image to image type tasks, trainablility on small datasets, and fast evaluation time in comparison to networks with large fully connected layers. The trainability on small datasets make this particulary effective on predicting steady state flow because generating simulation data to train on is time consuming. Our network is able to train on realively small datasets of only 4,000 flow simulation in comparision to the 100,000 required in previous work predicting steady state flow. Other modifications we have used are the use of gated residual blocks blocks that allow the gradient to be propogated extremely effeicently and heavely lowers training time.

5 Experiments

5.1 Datasets

To train the parameterization networks we generated a set of 10,000 examples for each system consisting of a parameter vector and their corrisponding geometry.

The heat sink simulations dataset consists of BLANK training examples simulated with implicit finite difference method.

5.2 Training

For all networks we used the adam optimizer. For S_{net}^{heat} or S_{net}^{flow} a learning rate of 0.0001 was used until the loss platued and then the learning rate was dropped to 0.00001. Mean Squared Error was used as the loss function however for the flow prediction network we scaled up the loss from the pressure field by 30 to roughly match the loss from the velocity vector field. Without this modification the network would train much slower and only learn the pressure field once the velocity vector field was firmly learned. During the course of this work other loss functions were experimented with such as adding a loss term in predicting the fluid flow forces on the object. This had benificial effects but complicated the method and is specific to fluid flow so was left out from the final network. The parameterization networks also used Mean Squared Error with a constant learning rate

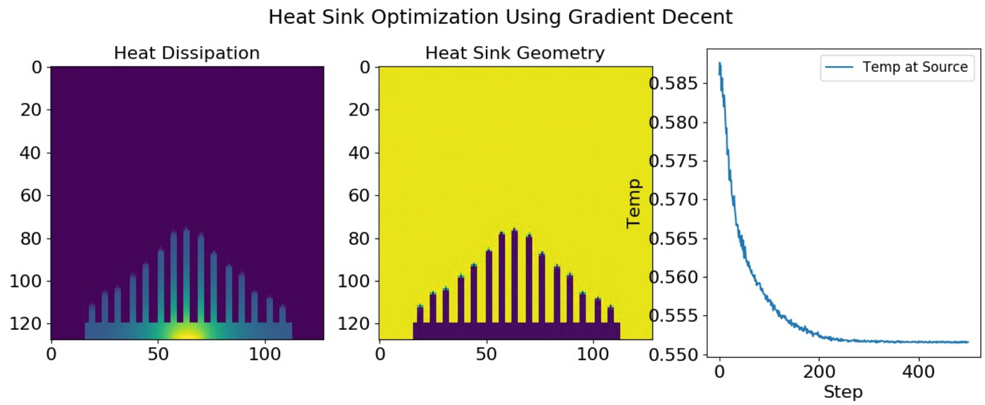

Figure 2: Optimization of heat sink using our gradient decent method.

of 0.0001. We found the parameterization networks trained extremely quickly and reqiured little parameter optimization.

5.3 Gradient Decent Optimization Details

We found a few minor tweaks to improve . We found that using gradient decent with momentum to perform very well. There

5.4 Heat Sink Optimization

As discussed above, the heat sink optimization task is to find a set of fin heights that maximaly cool a constant heat source given a fixed total lenght of the fins. The set up roughly corrisponds to an aluminum heat sink placed placed on a cpu where the heat source is treated as a . There is no heat disipation between the underside of the heat sink but all the area not on the heat sink is kept at a constant tempurature. The intuative solution to this optimization problem is to place long fins near the heat source and shorter fins farther away. Balancing this is a difficult task though because changing the length of any fin has a global effect on how much heat is disipaed by all the other fins.

After training our networks P_{net}^{heat} and S_{net}^{heat} we perform our proposed gradient optimization on the 15 fin heights to minizize the heat at the source. In figure **??** we see the optimized heat sink and see that the design resebles what our intuition tells us. We also see how the .

We found several minor imporvements

5.5 Flow Prediction Accuracy

5.6 Automated Design of 2D and 3D Airfoils

5.7 Comparison of Computation Times

6 Conclusion

In this work we hav

7 Sample Tables

Acknowledgments

This work was made possible through the `http://aigrant.org` run by Nat Friedman and Daniel Gross. This work would not have be possible without this very genourous support.

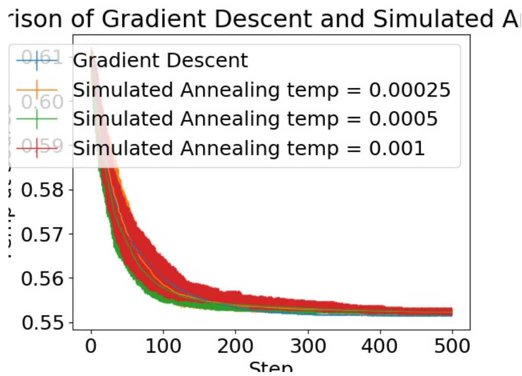

Figure 3: A comparison of our gradient decent based method and simulated annealing in optimizing the design of a heat sink. Several starting tempurature for the simulated annealing algorithm are provided. Each optimization was performed 40 times from the same starting design and the bars show the standard deviation over these runs.

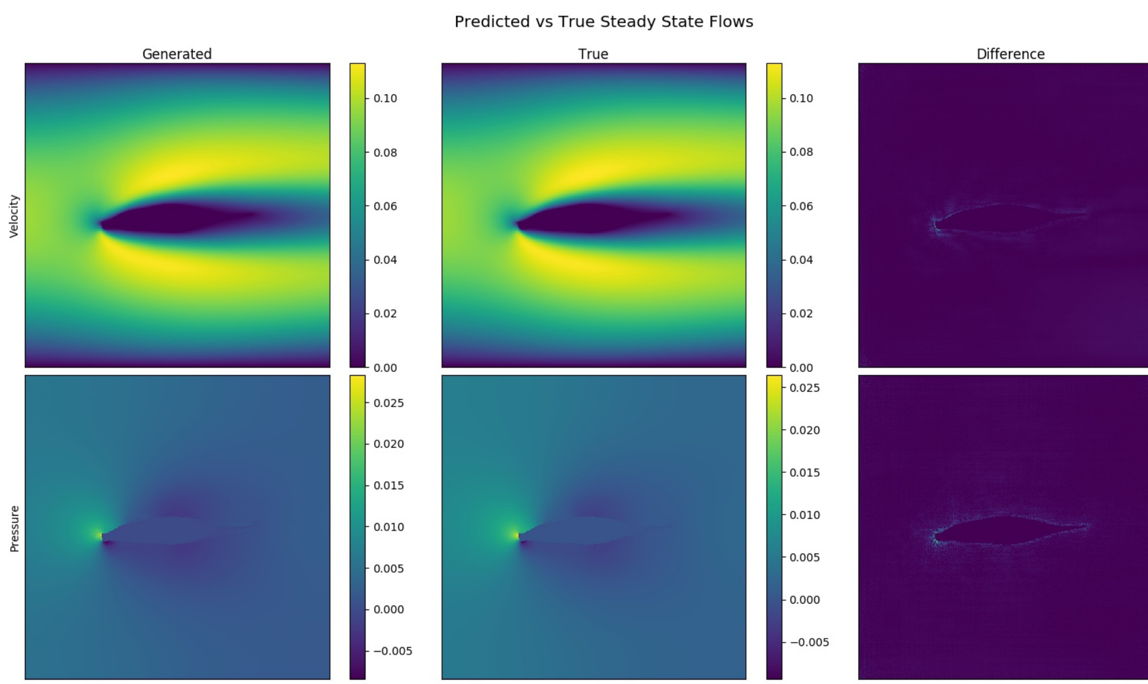

Figure 4: Flow Accuracy

REFERENCES

Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 481–490. ACM, 2016.

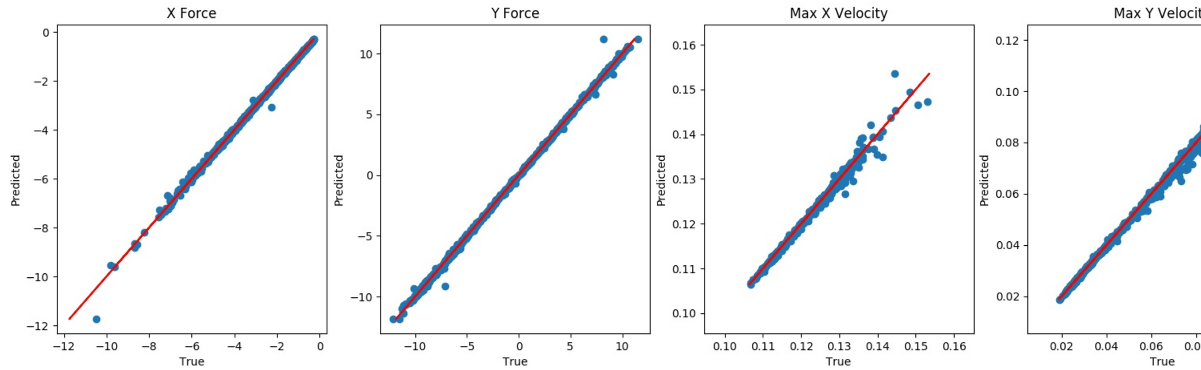

Figure 5: Flow

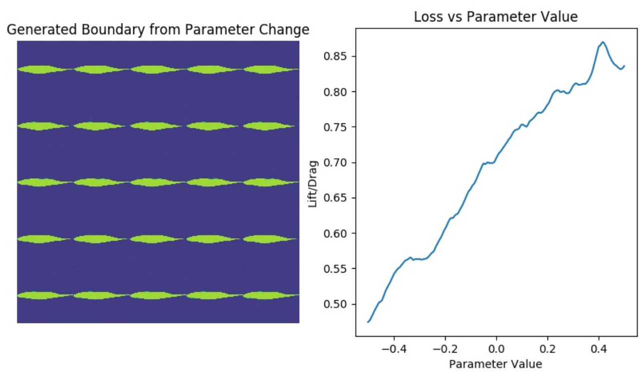

Figure 6: Flow

8 APPENDIX

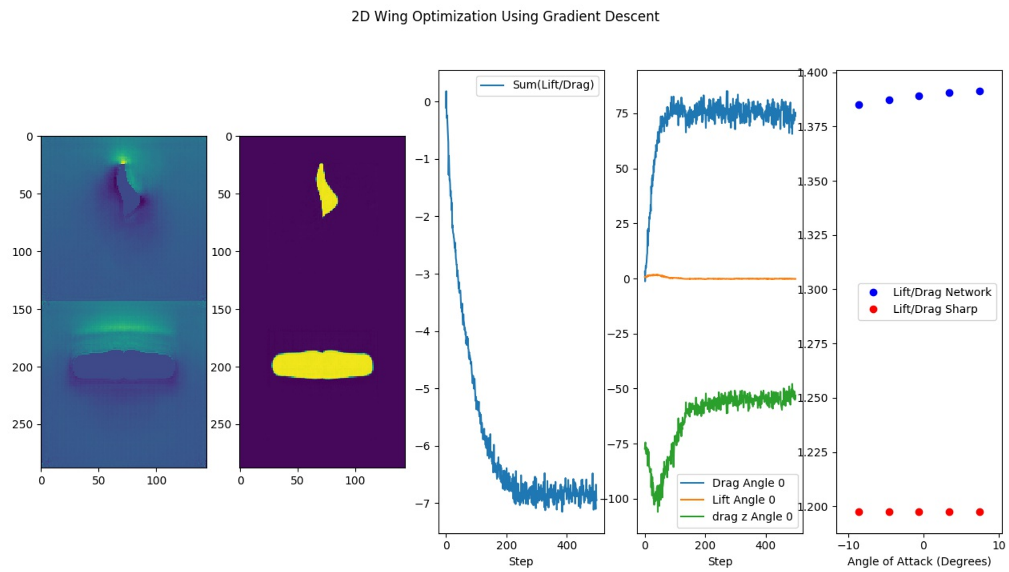

Figure 7: Flow

../test/figs/learn_comparsion.jpeg

Figure 8: Flow

Table 1: Sample table title

Batch Size	1	2	4	8	16
Flow Net 512^2	0.150 sec	0.101 sec	0.077 sec	0.065 sec	0.058 sec
Param Net 512^2	0.083 sec	0.045 sec	0.026 sec	0.015 sec	0.011 sec
Learn Step 512^2	0.494 sec	0.345 sec	0.270 sec	0.231 sec	Nan
Flow Net 144^3	0.826 sec	0.686 sec	0.627 sec	0.623 sec	Nan
Param Net 144^3	0.195 sec	0.144 sec	0.119 sec	0.106 sec	0.093 sec
Learn Step 144^3	3.781 sec	Nan	Nan	Nan	Nan

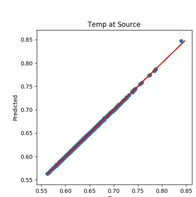

Figure 9: Flow

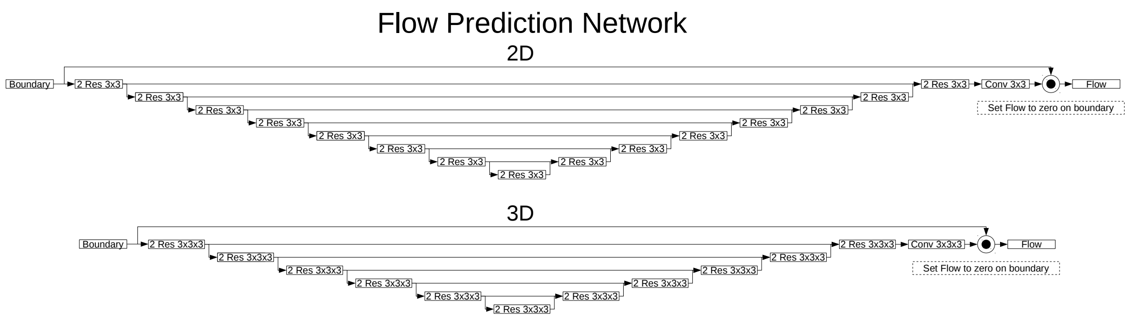

Figure 10: Flow