# Primary Weight Estimation for eVTOL Aircraft via Neural Network Regression

*Jordan Smart*

*Stanford University, Stanford, CA, 94305, USA*

**This work demonstrates an approach for developing a neural network capable of replicating the explicit, semi-analytic methods for estimating the weight of various components of an electric vertical take-off and landing (eVTOL) aircraft which available in the SUAVE aircraft design environment. An explanation of the data-structure associated with a SUAVE vehicle is given, along with demonstration of how a suitable dataset of such vehicles has been generated, and how the data is to be made available for use in the neural network regression task.**

## I. Nomenclature

*SUAVE*: Stanford University Aerospace Vehicle Environment (deprecated)
*MDAO*: Multi-Disciplinary Analysis and Optimization
*eVTOL*: Electric Vertical Take-Off and Landing
*EVW:* Empty Vehicle Weight
*MTOW*: Maximum Take-off Weight
*FCNN:* Fully Connected Neural Network
*HRNN:* Hierarchical Recurrent Neural Network
*ReLU:* Rectified Linear Unit
$\alpha$: Learning Rate
$\beta_1$: Adam Momentum Hyperparameter
$\beta_2$: Adam Regularization Hyperparameter
$\widehat{y_i}$: Neural Network Prediction, Sample $i$
$y_i$: Actual Evaluation, Sample $i$

## II. Introduction

Current aircraft design techniques are, broadly speaking, adequate for the job. Certainly, improvements in flow simulation and structural analysis software will allow for future improvement in performance on specific tasks, but current methods and expected next-generation approaches often suffer from significant computational and labor overhead costs.

Evaluating vehicle level properties such as weight, maximum thrust, or glide ratio are performed typically one of two ways – via an integrated conceptual design software suite, or via hand calculation by an individual engineer.

The former requires access to, training, and familiarity with the specific software suite, and the creation of a fully detailed aircraft model, which can then be evaluated by complete mission simulation.

The latter is time consuming in and of itself when evaluating an individual aircraft and becomes prohibitively so when attempting to consider a range of possible vehicles.

In order to provide an evaluator capable of calculating vehicle-level properties for a range of proposals quickly and efficiently with a minimum input of time, energy, and cost, the effort detailed here seeks to create a neural network capable of replicating the performance of a full-featured aircraft design suite without the need to create a complete aircraft model.

The architecture takes in the necessary vehicle parameters and returns an estimate of the empty vehicle weight (EVW) – i.e. structural mass, mission systems and onboard energy storage, without passengers or cargo. EVW is the weight of the plane “as it sits”.

The typical datastructure of computational models for aircraft treat the aircraft itself as a tree:

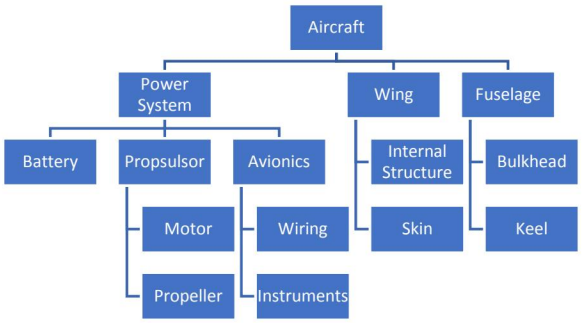


*Fig. 1:Aircraft Hierarchical Layout*

Associated with each level of this hierarchy are individual parameters, e.g. payload capacity at the aircraft level, maximum thrust at the propulsor level, etc. which together characterize the aircraft. For our purposes we need only fifteen parameters total, which form our vector as an input:

| Param. | Full Name | Min. | Max. |
|---|---|---|---|
| $MTOW$ | Max Take-Off Weight | 2E3 | 2E4 |
| $m_b$ | Battery Mass | 0.0 | 1E3 |
| $m_{PL}$ | Payload Mass | 0.0 | 5E3 |
| $L_F$ | Fuselage Length | 3.0 | 10.0 |
| $W_F$ | Fuselage Width | 2.0 | 10.0 |
| $H_F$ | Fuselage Height | 1.0 | 3.0 |
| $b$ | Wingspan | 0.0 | 20.0 |
| $c$ | Mean Aero. Chord | 0.0 | 5.0 |
| $t/c$ | Thickness-to-Chord | 0.01 | 0.30 |
| $w/f$ | Winglet Fraction | 0.0 | 0.25 |
| $N_L$ | No. Lifting Rotors | 2 | 20 |
| $N_T$ | No. Thrusting Rotors | 1 | 10 |
| $B_L$ | No. Lifting Blades | 2 | 10 |
| $B_T$ | No. Thrusting Blades | 2 | 10 |
| $R_T$ | Propeller Tip Radius | 0.0 | 5.0 |

*Table 1: Input Parameters and Ranges*

A significant obstacle to implementation of a neural network approach to this problem is the absence of a suitable dataset. As no fully developed eVTOL has yet been flown, and the total number of real-world designs is unlikely to exceed 100 at any point, it becomes necessary to generate an entirely artificial dataset with which to train the network. As our end-goal is not necessarily to exceed human-level performance on these tasks but merely to develop an appropriate regressor for the current state-of-the-art, this is wholly appropriate.

To that end, a uniform random sample of the parameters given in Table 1 were used to construct full-featured aircraft models within the SUAVE aircraft design tool, and its onboard weight estimation tools were used to generate EVW estimates which act as our labels for the purpose of comparison with our neural network's output.

## III. Related Work

As this is a novel application of neural networks, there is little in the way of directly comparable pre-existing literature. Traditional weight estimation methods and their derivation are detailed in a number of references including notably Raymer[1] and Kroo[2]. SUAVE's own implementation is a derivative of that used by A$^3$ for Project Vahana, published as part of the results of their MDAO study for that aircraft[3].

Neural networks have been applied in other areas of aircraft design, particularly in efforts to regress and replace particularly computationally intensive segments of computational fluid dynamics simulation, such as solving the discrete Boltzmann equation[4]. Unfortunately, the numerical schema, data structure, and underlying equations are so far removed from what is being attempted here that it renders any insight into architecture design irrelevant.

More relevant in topic and data structure, Kalogerakis, et al.[5] sought to generate plausible aircraft based on a low-level parameterization, but approached the problem on a visual rather

than performance basis, and implemented their generator using a conditional probability model rather than a neural network.

## IV. Dataset and Features

As mentioned above, the dataset was generated using a uniform random sampling over the design parameters between the minimum and maximum values as listed. Engineering judgment was used to determine appropriate discretization intervals for each variable, with finer discretization given to parameters against which the model is most sensitive. In total, this created a design space of 8.7E15 possible design combinations.

Limitations arise, however, with regard to the speed with which design models can be constructed and evaluated. Because each model must be fully constructed and evaluated within SUAVE, it is only possible to generate approximately 1E4 data points per day of computation. In total 1E5 data points were generated for use in training our neural network, which proved to be a significant limiting factor in its effectiveness.

## V. Methods

Although the innate structure of the problem might suggest that a convolutional neural network (CNN) or hierarchical recurrent neural network (HRNN) might be naturally suited to the task, the low dimensionality of the input and output as well as the relatively small amount of inter-dependent computation that must be done in traditional weight estimation methods suggest that those architecture's benefits would be wasted. The reduction in overall parameter size and ability to maintain "awareness" of inputs is unlikely to be necessary in this instance.

For these reasons, this effort pursues the use of a fully connected neural network (FCNN) as the architecture for the hidden layers of our network. This choice, however, leaves several other key decisions open.

Principally, though this problem seems to naturally take the form of a regression task, prior work has shown that in some cases, neural networks may still perform better when the problem is posed as a classification task rather than direct numerical regression.

Evaluating their comparative performance requires a different choice of output layer for the network, loss function, and performance metric.

To compare the network's numerical output directly with the SUAVE EVW label, we will use a rectified linear unit (ReLU) output layer with mean squared error loss:

$$J_{MSE} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2$$

As this has a directly interpretable meaning, being the square of the gross error between the network and our expectation, we can also use this as our performance metric, with human-level-performance considered to be ~1E4, corresponding to a gross error of ~100 kg. For the remainder of this paper, this architecture shall be referred to as the "Regressor" network.

To treat the problem as a categorization, we will have to first pre-process SUAVE's EVW ouput and discretize the output space into a number of possible categories. Since we anticipate human level performance to be ~100 kg, we use this as our unit of discretization, creating 100 categories from 0 kg to 10000 kg+. This has a mild normalization effect on the dataset, as it groups all outliers into a single category for the network to recognize. We then apply a softmax output layer, and use categorical cross-entropy loss:

$$J_{CCE} = \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} -\big(y_{ij}\log\hat{y}_{ij} + \big(1-y_{ij}\big)\log\big(1-\hat{y}_{ij}\big)\big)$$

Since this has no direct physical meaning, we will use categorization accuracy as our metric, expecting human level performance to be near-perfect. This shall be further referenced as the "Categorizer" network.

In both cases, an Adam optimizer was employed, with learning rate $\alpha$ varied from 1E-4 to 1E-1 experimentally, and its typical beta values of $\beta_1$ at 0.9 and $\beta_2$ at 0.999.

The hidden sections of the network were evaluated with 32, 64, and 128 hidden units, and 3, 6, and 9 layers.

## VI. Results and Discussion

No combination of hyperparameters produced acceptable performance. The Regressor network produced at best 2.7E7 MSE loss on the training set and 4.4E7 on the test set. The Categorical network produced at best 3.6% accuracy on the training set and 1.9% on the test set.

These results correspond to approximately 5000-6500 kg errors in weight estimation for the Regressor network, suggesting that the network has learned a rough order of magnitude for its output, but struggles to be any more precise than that.

Charitably the Categorizer network could be said to be performing 2-3X better than base-rate random selection, indicating again, some learning, but of limited progress.

Typical error analysis may prove to be fruitful in future work, however, the largest obstacle encountered was difficulty in training the network.

Likely owing in large part to the relatively small dataset, the both the Regressor and Categorizer networks would converge on their respective best results in fairly short time. The Categorizer network in particular would typically converge within 3 epochs and never improve from that point.

The Regressor network would typically converge with 100 epochs, but displayed very unusual convergence behavior. If, following the initialization of it's weights, it's initial loss was between the converged 2.7E7 value and ~4.4E7, it would progressively converge down to 2.7E7, and then either remain at that figure, or else jump up to 4.4E7 and instead converge at that value, as evidenced by that behavior on the test set. If the initialized loss was above the 4.4E7 value, the optimization would move to that point fairly rapidly, and not improve.

## VII. Conclusion and Future Work

Though the results of this investigation were disappointing, they did nevertheless provide several important insights and suggest several avenues for improvement.

First, it provides some validation that datasets on the order of 1E5 are inadequate for the task of training an end-to-end deep learning replacement for this particular aspect of aircraft design. Effort must be given to improving the data generation apparatus so that larger datasets can be made available.

This may go some way to mitigating the difficulty in training the network, but if the recalcitrant training behavior persists, adjustments to the optimizer must be considered. The bifurcated convergence of the Regressor network suggests that a simulated annealing optimizer would be appropriate.

More detailed evaluation metrics might provide better insight. For the Regressor network, in addition to mean squared loss, an absolute comparison in predictions, or a probabilistic breakdown of likely network predictions for a range of labels would give insight into the bias/variance tradeoff at play.

For the Categorizer, a full confusion matrix and an evaluation metric that included an element of

the magnitude of the error as well as the accuracy would be an improvement.

## VIII. References


[1] Raymer, D.P., "Aircraft Design: A Conceptual Approach, Sixth Edition", *AIAA Education Series*, 2018

[2] Kroo, I., "Aircraft Design: Synthesis and Analysis", Stanford University [online reference], URL: aerodesign.stanford.edu/aircraftdesign, 2018

[3] Bower, G. "Vahana Configuration Trade Study – Part II" Vahana [online article], URL: https://vahana.aero/vahana-configuration-trade-study-part-ii-1edcdac8ad93, 2017

[4] Hennigh, O., "Lat-Net: Compressing Lattice Boltzmann Flow Simulations using Deep Neural Networks", arXiv:1705.09036 [stat.ML]

[5] Kalogerakis, E., Chaudhuri, S., Koller, D., Koltun, V., "A Probabilistic Model of Component-Based Shape Synthesis", *ACM Transactions on Graphics, Vol. 32, No. 4*, 2012


## IX. Appendix

The necessary code has been stored on my personal fork of the SUAVE repository:

https://github.com/JTrentSmart/SUAVE

On the branch, feature-nn-regressor.

SUAVE itself has significant dependencies which may make installing the package and running the code difficult, however the relevant script is located at:

SUAVE\trunk\SUAVE\Surrogate\...
neural_network_surrogate_functions.py

for review. The relevant methods the neural network is being regressed against are those for an eVTOL stopped rotor configuration, which can be found at:

SUAVE\trunk\SUAVE\Methods\Weights\...
Buildups\Electric_Stopped_Rotor

SUAVE\trunk\SUAVE\Methods\Weights\...
Buildups\Common