FUSION: ADVANCED STELLAR MODELING

Ansh Menghani | North Carolina School of Science and Mathematics

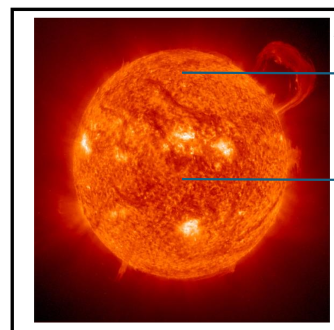
PROJECT GOAL

Develop a PINN (Physics Informed Neural Network) to accurately model stellar parameters from minimal input parameters.

SIGNIFICANCE OF STUDY

- Physics is all about approximation. No one formula or statistical model can account for every factor and predict any event with full accuracy. Many models make assumptions about the systems being modeled that aren't realistic.
- Accurate modeling has been particularly difficult in fields such as astrophysics, where the subjects being studied are such mind-blowing distances away from Earth.
- Neural Networks are trainable algorithms inspired by the human brain that can learn complex relationships easily—making simulations and modeling of complex systems much easier than ever before.
- A neural network model built for stellar modeling will transform the way scientists look at physical systems by allowing them to run unprecedentedly accurate and speedy simulations of many such systems.

BACKGROUND



A stellar parameter is an attribute of a star such as:
- Surface temperature
- Luminosity
- Radius

Nuclear fusion supports stars under their own gravity

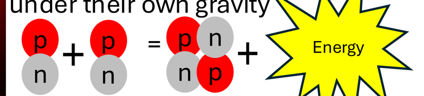
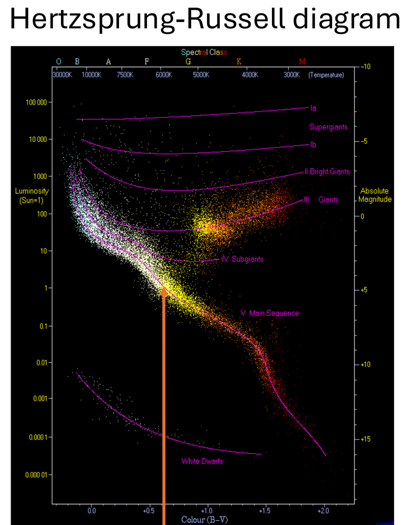


Image from NASA (public domain): science.nasa.gov/sun/
Nuclear fusion figure made by Ansh Menghani (author) in PowerPoint.

Hertzsprung-Russell diagram



The Sun:
Surface Temperature: 5778° Kelvin
Luminosity:
1 solar luminosity ($L/L_\odot$)

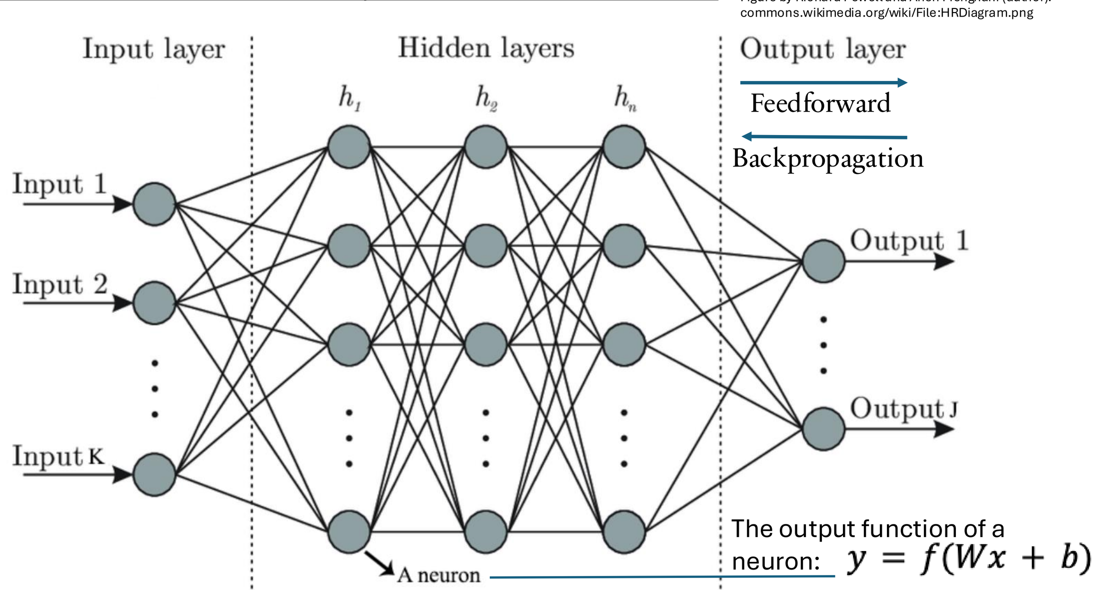
Figure by Richard Powell and Ansh Menghani (author): commons.wikimedia.org/wiki/File:HRDiagram.png

Stellar parameters are currently modeled using formulae and statistical models. However, these models make assumptions about stars that aren't fully true.

Maximum Effective Temperature (°K)	Spectral Class
4000	M
5200	K
7000	G
20000	F
34000	A
42000	B
greater than 42000	O

Maximum Luminosity ($\frac{L}{L_\odot}$)	Luminosity Class
0.1	D
25	V
100	IV
1300	III
8000	II
125000	Ib
greater than 125000	Ib

Figures made by Ansh Menghani (author) using LaTeX.

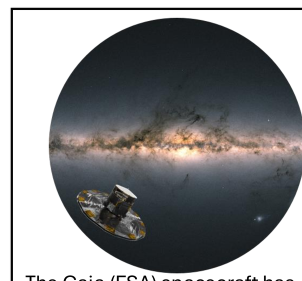


The output function of a neuron: $y = f(Wx + b)$

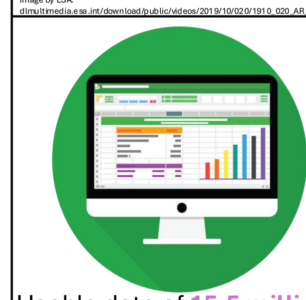
Figure by Nooraiepour, A., Duman, T., and Ansh Menghani (author) doi: 10.1109/TCOMM.2017.2704586

- A neural network can be thought of as an incredibly complicated mathematical function.
- Each "neuron" represents a variable in the function, and the lines represent the magnitude of impact (or the "weight") that variable has on the ones around it.
- Once the network's architecture is defined, these weights can be trained (optimized) and updated based on some data such that the neural network models that data.

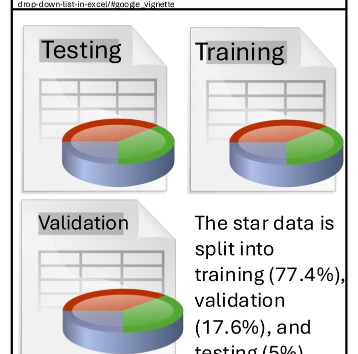
DATA COLLECTION, MODEL STRUCTURE, AND TRAINING



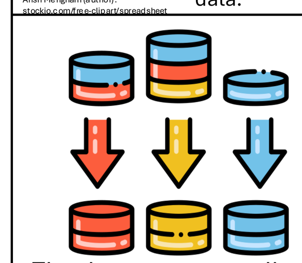
The Gaia (ESA) spacecraft has gathered data on two billion night-sky objects. This publicly-available data was used to train the model.



Usable data of 15.5 million stars for model training were extracted from the dataset.

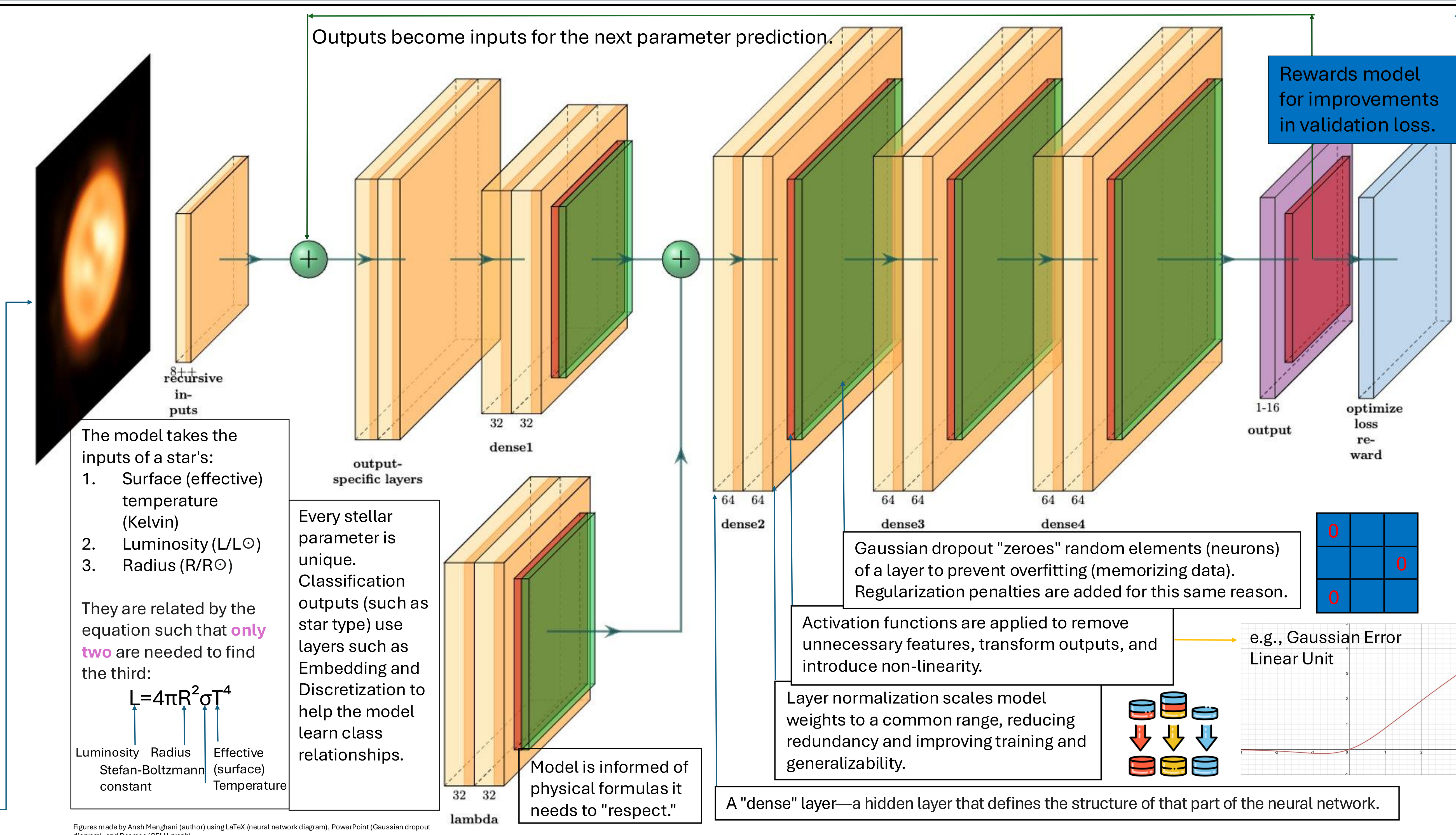


The star data is split into training (77.4%), validation (17.6%), and testing (5%) data.



The data was normalized.

Image from Flaticon (public domain): flaticon.com/free-icon/normalization_3295467



Model uses current weights to predict stellar parameters on a set of test inputs.

The model's inaccuracy (loss) is calculated by comparing its predictions with the corresponding truth values using loss functions.

The Adaptive Moment Estimation (ADAM) algorithm updates the model's weights (through backpropagation) to minimize the loss value.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{y}_i|$$

$$MSLE = \frac{1}{n}\sum_{i=1}^{n}(log(1+\hat{y}_i) - log(1+y_i))^2$$

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(1+\hat{y}_i) - log(1+y_i))^2}$$

Examples of loss functions (y_i is the "truth value" of the i^{th} datapoint in a test set, and $\hat{y}_i$ is the model's prediction of that datapoint).

After training, the neural network is able to model a star's:

1. Dimensions
2. Absolute Bolometric Magnitude
3. Absolute Magnitude
4. Absolute bolometric luminosity
5. Average Density
6. Central Pressure
7. Central Temperature
8. Lifespan
9. Surface Gravity
10. Gravitational Binding Energy
11. Bolometric Flux
12. Metallicity
13. Spectral Class
14. Luminosity Class
15. Star Peak Wavelength
16. Star Type
17. Spectral Classification
18. Bolometric Correction

Based on the star's effective temperature, luminosity, and radius!

Figures made by Ansh Menghani (author) using LaTeX (neural network diagram), PowerPoint (Gaussian dropout diagram), and Desmos (GELU graph).

RESULTS

- After finding the optimal model structure, implementing custom methods, and minimizing the model's loss (model inaccuracy) by iteratively training and optimizing the model's weights (i.e., the coefficients of the neural network's formula/algorithm), tests were run to determine how well it performed.
- These tests were run on the 5% of data set aside at the beginning of training (775,000 lines of data). The model has not seen this data before during training, testing its capabilities of making predictions on new data.

Figure 1: This represents the loss, or the model's error when fed training data each time it tested itself during training. This graph is smoothed to show the loss trend.

Figure 2: This represents the validation loss, or how well the model performed when tested with data that it hadn't seen during training. This graph has been smoothed to show the validation loss trend.

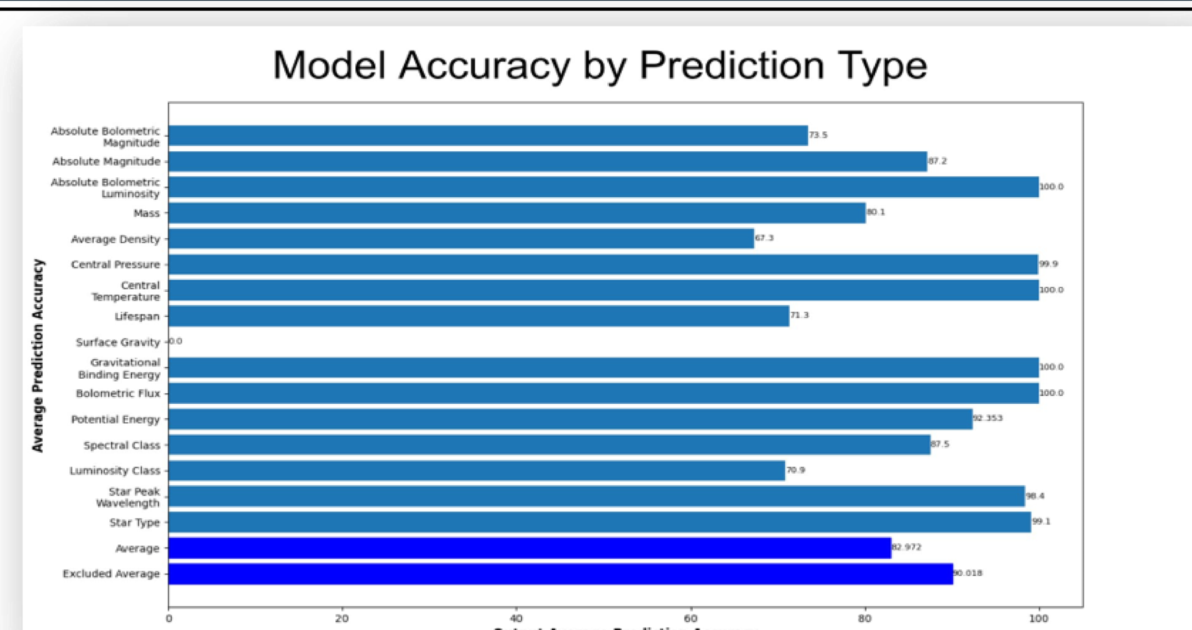
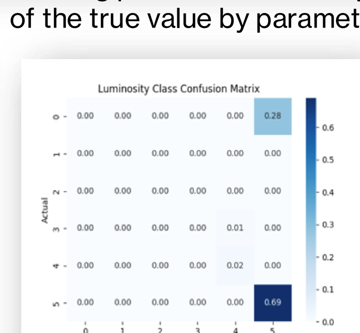
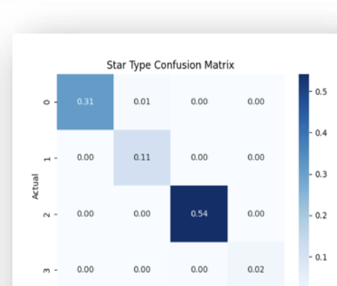


Figure 3: Percentage of 775,000 testing predictions made by the model that falls between ±5% of the true value by parameter.

Accuracy Score	0.9860
Recall Score	0.9860
Precision Score	0.9726
F1 Score	0.9792



Accuracy Score	0.7112
Recall Score	0.7112
Precision Score	0.5058
F1 Score	0.5912



Accuracy Score	0.9870
Recall Score	0.9870
Precision Score	0.9873
F1 Score	0.9871

Figures 4, 5, and 6: Classification evaluation metrics. The confusion matrix shows ratios of how often the model classified true positive, true negatives, false positives, and false negatives. The scores represent various evaluations of the model, all of which have a best possible score of 1.

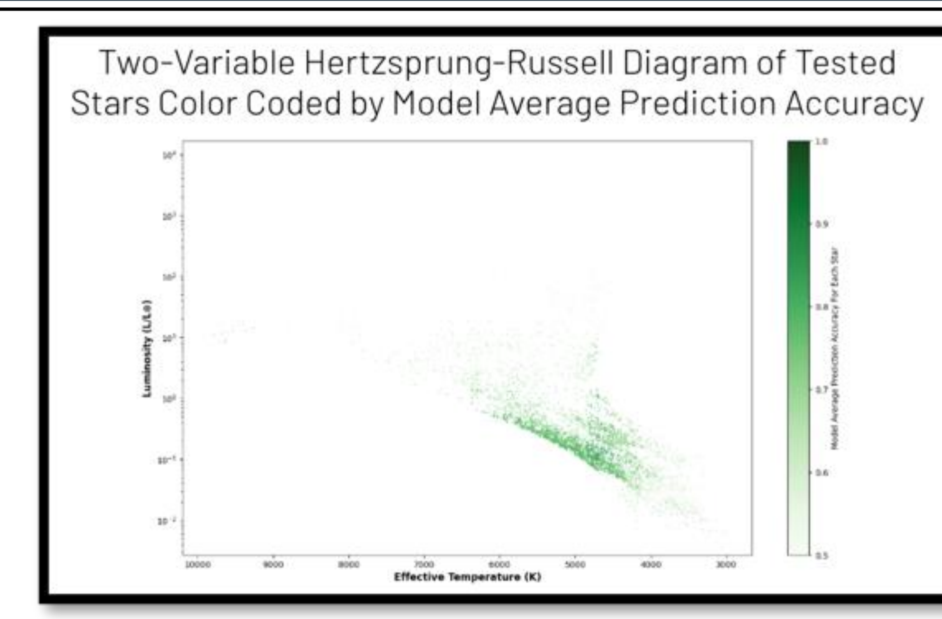


Figure 7: This represents the accuracy of each prediction the model makes and compares them to the star type of the star being analyzed (1.0 = 100%).

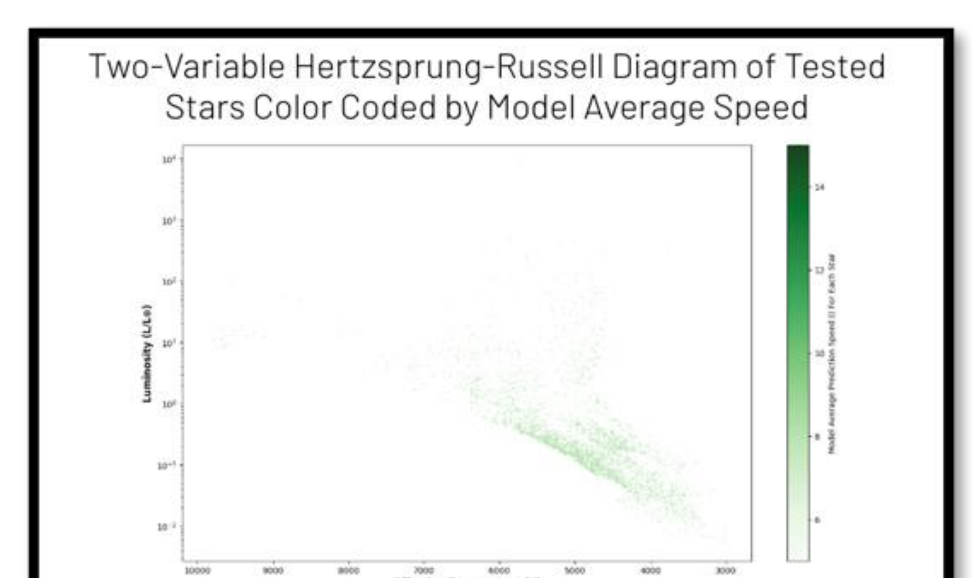


Figure 8: This represents the speed of each prediction the model makes and compares them to the star type of the star being analyzed (1.0 = 1ms).

Figures in this section made by Ansh Menghani (author) using TensorBoard (figures 1 and 2), matplotlib (figures 3, 7, and 8), and scikit-learn (figures 4, 5, and 6).

DISCUSSION AND CONCLUSIONS

Discussion

- The model behaves optimally during training, minimizing loss and validation loss.
- The model makes *good* predictions on stars with effective (surface) temperatures within 4200 and 7000 degrees Kelvin and with 0.1 to 1 solar luminosities. It performs *best* on stars with effective temperatures within 4500 and 4800 degrees Kelvin and a luminosity between 0.95 and 1.4 solar luminosities. This indicates that the effective temperature of the star is the most important input parameter.
- Some types of stars are predicted with a low accuracy due to lack of training examples of those star types.
- The input parameters have little to no effect on prediction speed.
- Four of the first nine parameters have a prediction accuracy below 85%, while only one of the next seven are under 85%. This indicates the custom method that recurses outputs to inputs works well!
- The classification evaluation metrics indicate a near-perfect model for spectral class and star type, and an "okay" one for luminosity class.

Conclusions

The goals described earlier have been met:

Final Model Accuracy: 90.018%

Average Prediction Speed: 8ms

This project can be used to conduct further research. Possible limitations of this project include running simulations on certain types of rare stars (e.g., neutron stars) due to lack of input data. In the future:

1. Parameter accuracy will be worked on and increased.
2. A more diverse dataset will be used to optimize the model for special types of celestial bodies such as neutron stars.

APPLICATIONS

This model can find previously unknown relations that cannot be easily mathematically/statistically modeled.

The model performs the job of sixteen individual models with similar speed and accuracy as one.

This model can be used to simulate many stars at once, allowing scientists to aggregate big data sets and determine what they may want to use resources on to study.

This project adds to the migration of machine learning into physics. The custom procedures used in it can be replicated to better other models.

The model can simulate objects that have very minimal information gathered due how hard it is to gather data on something so far.

In contrast to other work in this space, this project builds a neural network that models more parameters and takes a more accurate approach than mathematical and statistical models.

KEY REFERENCES

- This work has made use of data from the European Space Agency (ESA) mission Gaia (https://www.cosmos.esa.int/gaia), processed by the Gaia Data Process- ing and Analysis Consortium (DPAC, https://www.cosmos.esa.int/web/gaia/ dpac/consortium). Funding for the DPAC has been provided by national institutions, in particular the institutions participating in the Gaia Multilateral Agreement.
- Astronomical terms and constants. (n.d.). https://www.astro.princeton.edu/~gk/A403/constants.pdf
- Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras
- Types. (n.d.). Science.nasa.gov. https://science.nasa.gov/universe/stars/types
- Mart ́ın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, An- drew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefow- icz, Lukasz Kaiser, Manjunath Kud- lur, Josh Levenberg, Dandelion Man ́e, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vi ́egas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learn- ing on heterogeneous systems, 2015. Software available from tensorflow.org.