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 can account for every factor and predict any event with full accuracy. Over the years, scientists have used different methods to make these predictions, such as mathematical formulas and statistical models.

Modeling has been particularly difficult in fields such as astrophysics, where the subjects being studied are such mind blowing distances away from Earth.

Recent advances in machine learning techniques have greatly increased the ability of these models to understand complex relationships that are present in stars.

Specifically, neural networks have the capabilities to model multiplex relationships when there is multiple dimensions of information involved.

A neural network model that can successfully and accurately model the parameters of stars will allow researchers to have more confidence in their results, knowing that while the complex processes observed in stars may not be able to be fully modeled using mathematical and statistical models, the accuracy of their modeling will be greatly improved when using a properly trained neural network to do similar tasks.

IMPORTANT RESEARCH

- Stars are big balls of hot gas held together by their own gravity. These stars undergo nuclear fusion, the process of releasing energy on an atomic scale by combining atomic nuclei. From the beginning of a star's life to its end, it has varying values of parameters (attributes) such as temperature, luminosity, and radius. Some of these are easy to measure, while others are not.

- Many of these parameters have equations associated with them. Most of these equations assume some "ideal" conditions, such as an ideal gas. Of course, this is not the case in the physical universe.

- Stars can be plotted on a Hertzsprung-Russell digram based on their Luminosity and Effective (surface) Temperature, as shown below. Stars can also be sorted into different Spectral Classes (based on temperature) and different luminosity classes.

- There are many different types of stars. From smallest to largest: white dwarves (the leftover core of a medium-sized star), a brown dwarf (an object too small to be a star but too large for a planet, red dwarves, main-sequence stars (most stars), supergiants, and hypergiants.

Yerkes spectral and luminosity classification tables

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

Above tables made by Ansh Menghani using LaTeX.

Observational Hertzsprung-Russell Diagram

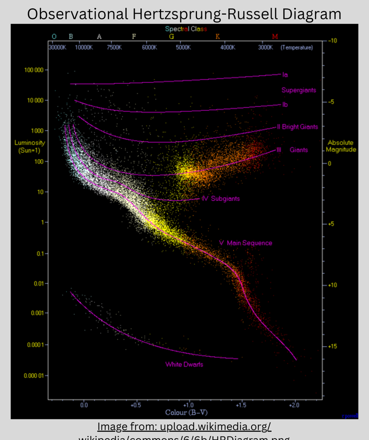


Image from: upload.wikimedia.org/wikipedia/commons/6/6b/HRDiagram.png

- A neural network will take *x* inputs, perform multiple "hidden" iterations of regression (where it uses the weights parameters in the regression) on the inputs, and use that to generate *y* outputs based on the training data. This is the basic structure of a neural network and many other techniques can be employed to improve the model. The basic output of a neural network regression layer is (where *Wx+b* is the regression output and *f* is a function that transforms the output in any way the user desires: $y = f(Wx + b)$

- Once the neural network calculates the output's of its regression, it uses a loss function to calculate how far off it was from the "true" answer. A common loss function used is called Mean Absolute Error, which calculates the difference between the predicted and actual values over the whole dataset and takes the average of those values:

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

- The model attempts to minimize this loss function during training to find the best fit model. The advantage of this method with neural networks is it allows them to model high-dimensional complex relationships.

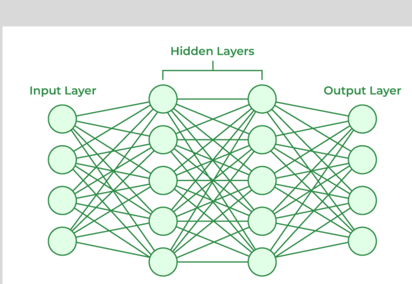


Image by GeeksForGeeks from www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

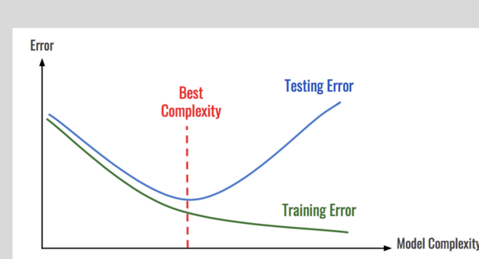


Image by AWS from https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

MODEL STRUCTURE

The model will take two of the three input features of a star someone wants to moel:

1. Effective Temperature (Kelvin)
2. Luminosity (L/L☉)
3. Radius (R/R☉)

All three of these compliment each other:

$L=4\pi R^2\sigma T^4$

Using the inputs above, the model will run it's algorithm to determine the star's:

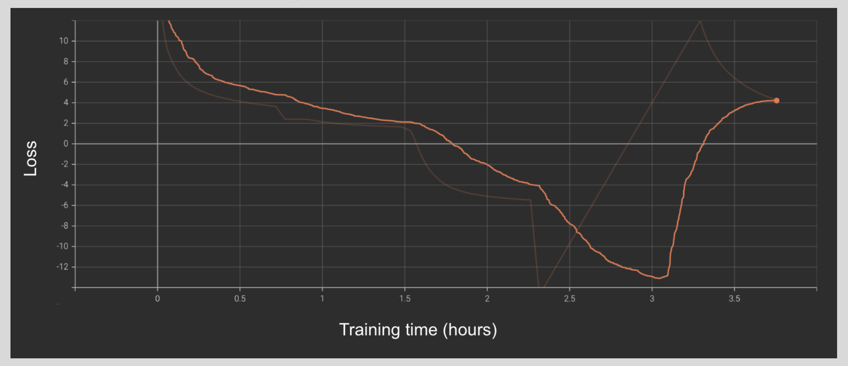
1. Dimensions
2. Absolute Bolometric Magnitude
3. Absolute Magnitude
4. Absolute bolometric luminosity
5. Average Density
6. Central Pressure
7. Central Temperature
8. Lifespan
9. Surface Gravity
10. Gravitational Binding Energy
11. Bolometric Flux
12. Metallicity
13. Spectral Class
14. Luminosity Class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

The model structure employed many techniques to output the best performance. Here are some additions to the fundamental neural network structure that are used:

1. The Gaussian Error Linear Unit, Parametric Rectified Linear Unit, and Softmax functions are used to remove unnecessary modeled data points during training, transform data, and introduce non-linearity
2. Embedding is used for classification outputs to help the model learn the relations between them
3. Data is normalized using Layer Normalization
4. Gaussian dropout increases noise in the model to prevent overfitting/underfitting
5. The losses used are Mean Squared Logarithmic Error (MSLE), Root-MSLE, Mean Absolute Error, and Categorical Crossentropy
6. L1L2 regularizers are used to add a penalty to the model's "loss" function. This discourages overfitting
7. The model has been informed of physics formulas governing its outputs for higher accuray
8. A custom procedure has been built to reward the model for decreases in validation loss
9. Each output of the model will become an input for the model's next prediction, relating every stellar parameter

RESULTS

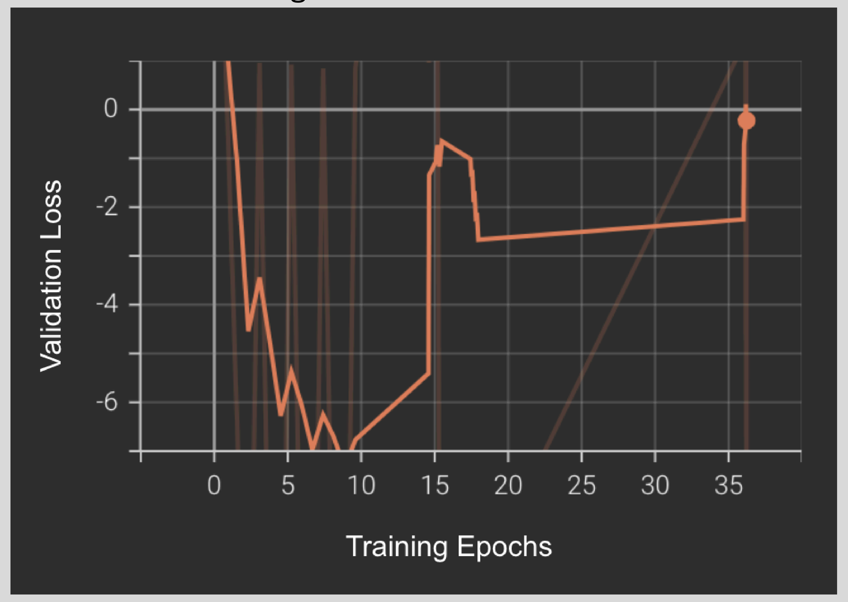
Figure 1: Loss vs. Time



Graph made by Ansh Menghani using TensorBoard.

Model loss vs. time (i.e., how far off the model was when testing itself during the training process. Ideally, this value should get as close to zero as possible while training.

Figure 2: Loss vs. Time



Graph made by Ansh Menghani using TensorBoard.

Model validation loss vs. time (i.e., how far off the model was when making predictions on new data during the training process. Ideally, this value should get as close to zero as possible while training.

Figure 3: Two-Variable Hertzsprung-Russell Diagram of Tested Stars Color Coded by Model Average Prediction Accuracy

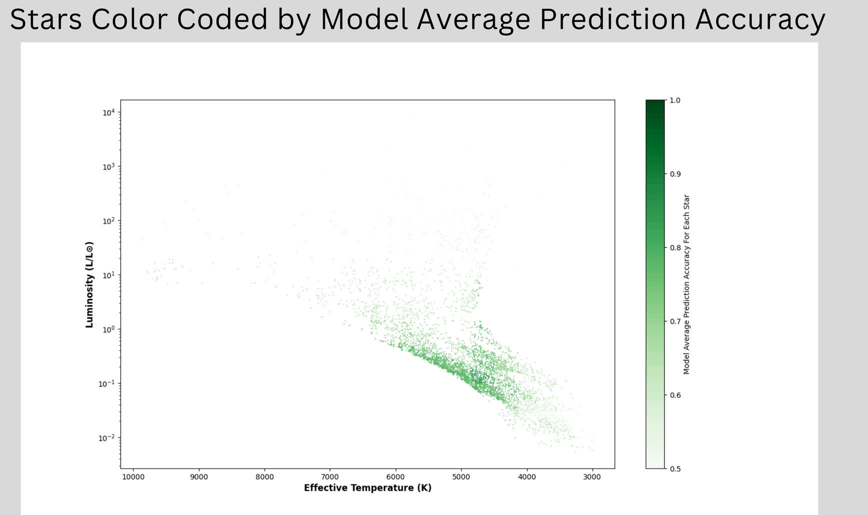


Image made by Ansh Menghani using matplotlib.

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 4: Two-Variable Hertzsprung-Russell Diagram of Tested Stars Color Coded by Model Average Prediction Speed

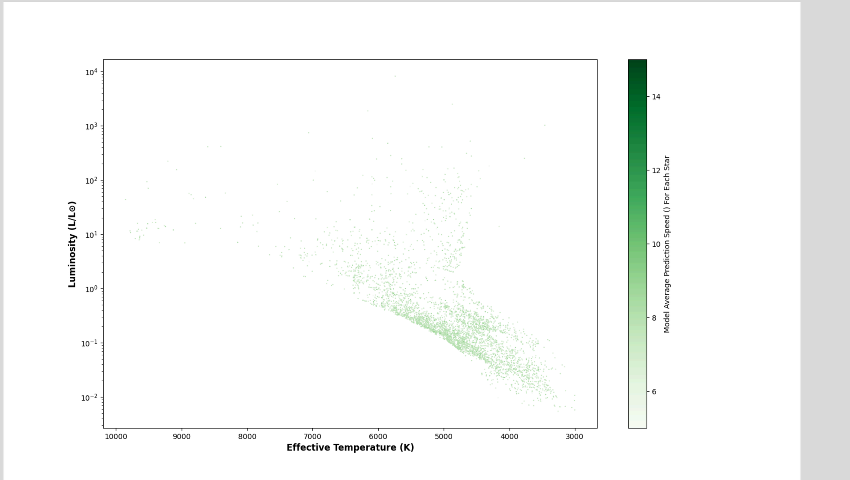
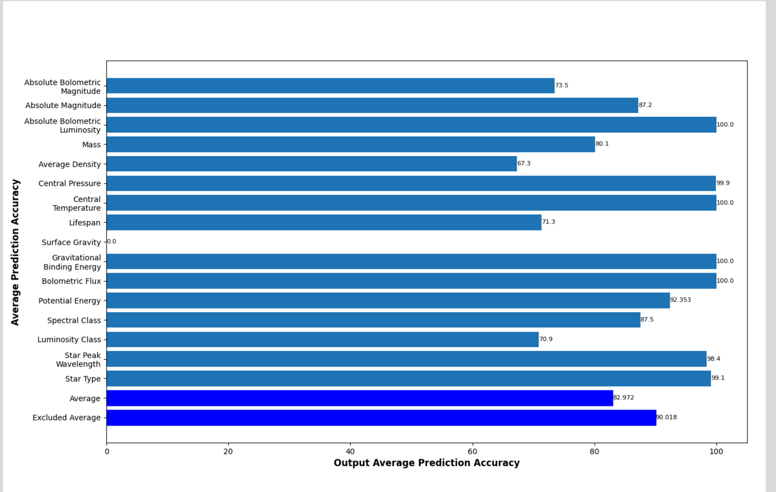


Image made by Ansh Menghani using matplotlib.

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).

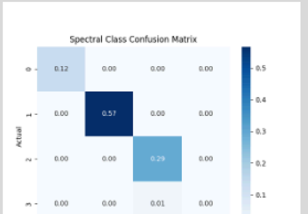
Figure 5: Model Accuracy by Prediction Type



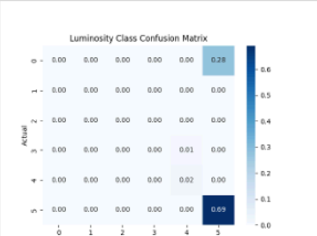
Graph made by Ansh Menghani using matplotlib.

This represents the prediction accuracy of each stellar parameter the model predicts.

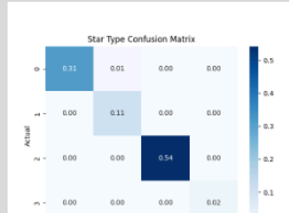
Figures 6, 7 & 8: Confusion Matrices and Classification Output Metrics



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

Images made by Ansh Menghani using sklearn. Tables made by Ansh Menghani using LaTeX.

These three figures represent the metrics that evaluate how well the model performs when classifying each star into a Spectral/Luminosity Class and its Star Type. The confusion matrices show how often the model picked the right classifications, and the scores provide metrics to evaluate this. An accuracy, recall, precision, or F1 score of 1.0 is perfect.

DATA COLLECTION AND PREPROCESSING

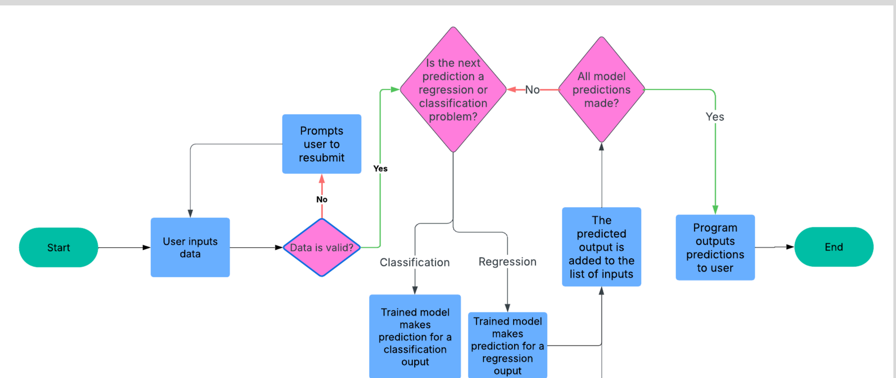
- The data used was from the Gaia DR3. Collected from the Gaia spacecraft, Gaia DR3 includes billions of stellar parameters observed from objects in the night sky.
- Out of these, ~15.5 million lines of data (403 million+ stellar parameters) were used in the project.
- 5% of the data (775,000 lines) were set aside for testing, and 18.5% of the remaining data was used in validation testing during model training.
- The data was transformed and normalized into model-readable data before training.

MODEL TRAINING

- During training, the model looks at all the data multiple times. If the model's validation loss does not improve over a certain period of time, training stops and the best "version" of the model is restored.
- The model trains in TensorFlow's "Graph Mode." This provides the best accuracy.

TESTING PROCEDURE

- Obtain the 5% of data left for testing the model (~750000 lines of data).
- For each row of data, run predictions.
- Calculate the accuracy of these predictions, and also calculates recall, precision, and F1 score for the classification outputs.
- Take average of metrics and plot them.



Flowchart made by Ansh Menghani using Lucid.

DISCUSSION

- The model behaves optimally during training, minimizing loss.
- We see that the model makes *good* predictions on stars with temperature within 4200 and 7000 degrees Kelvin and with 0.1 to 1 solar luminosities. It performs *best* on stars within 4500 and 4800 a luminosity between 0.95 and 1.4 solar luminosities. This indicates that the 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.
- The input parameters have little to no effect on prediction speed.
- The output with a 0% accuracy is likely due to its logarithmic scaling or due to overfitting within that parameter
- 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.

Overall Model Accuracy: 90.018%

Average Model Speed: 8ms

APPLICATIONS

- This project adds to the migration of machine learning into physics. The techniques 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.
- The model performs the job of sixteen individual models with similar speed and accuracy as one.
- This model can find previously unknown relations that cannot be easily mathematically/statistically modeled.

CONCLUSIONS

This project had very intriguing outcomes and the goals described earlier have been met. It has many possible practical applications as discussed. These practical applications can be used to conduct further research. Possible limitations of this project include running simulations on certain types of 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 objects such as neutron stars
3. A user interface will be built and the application will be made available for the public to use

KEY REFERENCES

1. 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.
2. Astronomical terms and constants. (n.d.). https://www.astro.princeton.edu/~gk/A403/constants.pdf
3. Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras
4. Types. (n.d.). Science.nasa.gov. https://science.nasa.gov/universe/stars/types
5. 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.