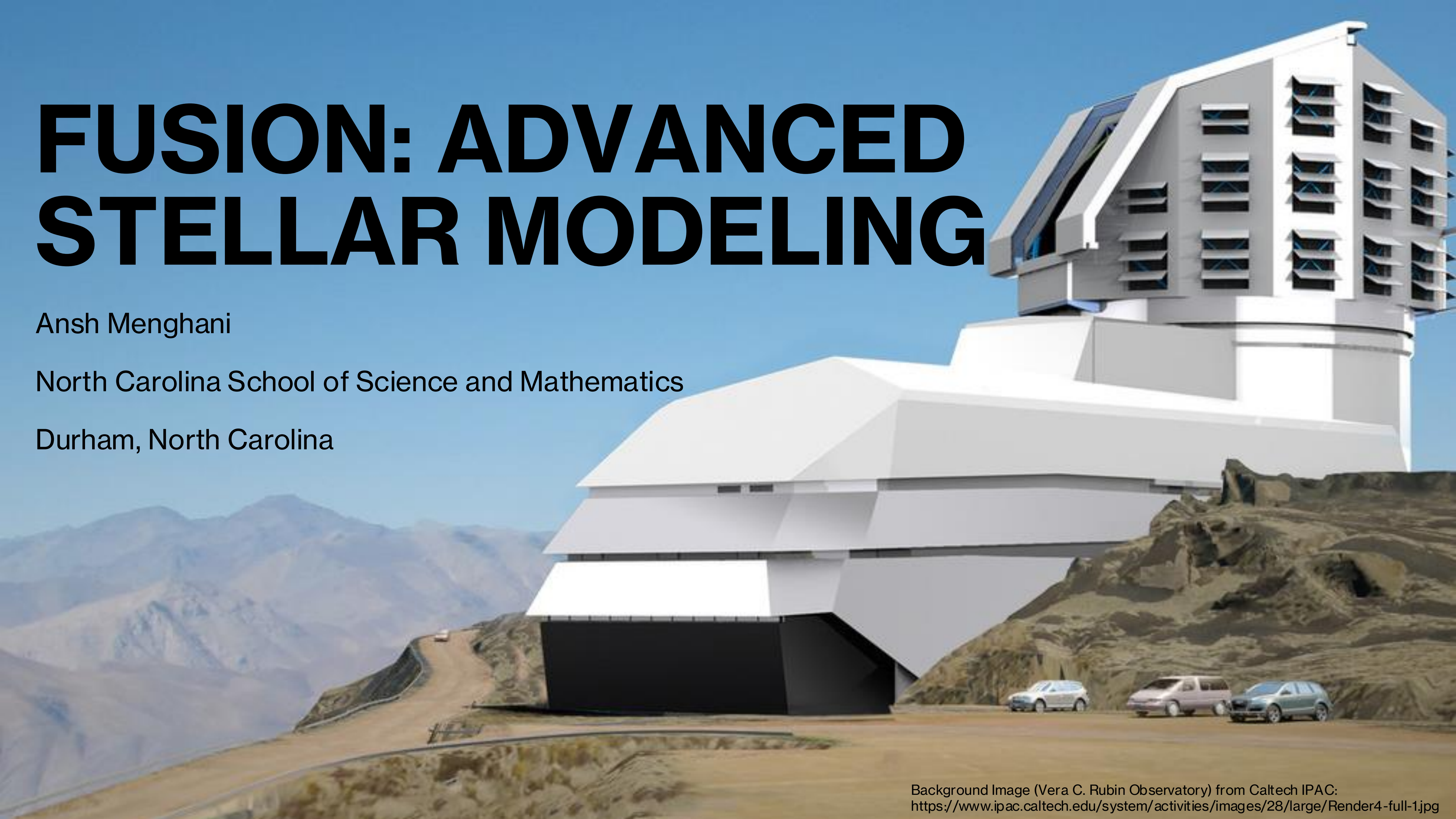


FUSION: ADVANCED STELLAR MODELING

Ansh Menghani

North Carolina School of Science and Mathematics

Durham, North Carolina

Background Image (Vera C. Rubin Observatory) from Caltech IPAC:
https://www.ipac.caltech.edu/system/activities/images/28/large/Render4-full-1.jpg

Introduction

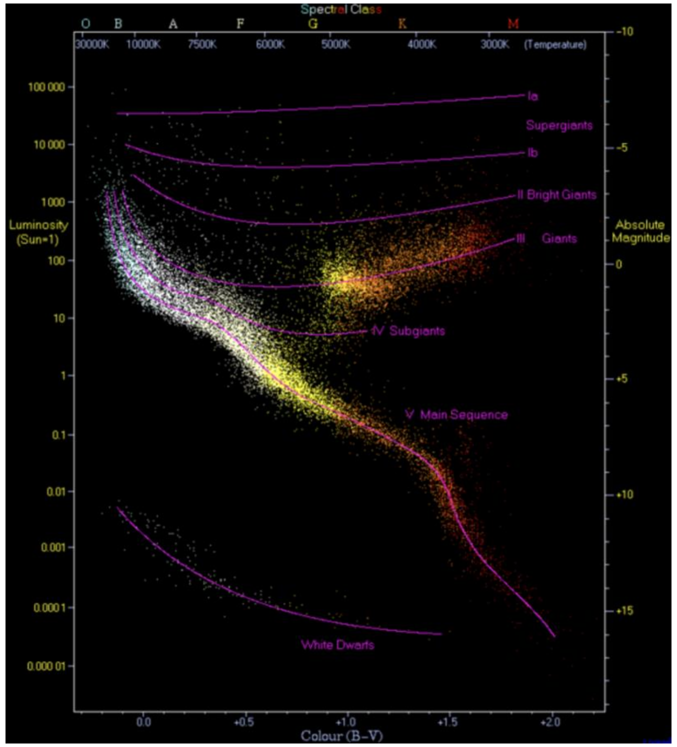
Physics is all about **approximation**. No one formula can account for every factor and predict any event with full accuracy.

Accurate predictions are especially hard in **astrophysics** where the subjects being studied are **mind-blowing** distances away from Earth and **hard** to gather data on.

Mathematical and statistical models of stars make **inaccurate** assumptions (such assuming the star is an ideal gas) and don't capture how **one part of a star affects the others**.

Neural Networks can model the **complexity** of stars and model them **better** than traditional methods.

If a neural network that models **many** stellar attributes at **once** can be created, it would provide tremendous benefit by **significantly reducing** modeling time while allowing scientists to **confirm** their results and get an **accurate and wholistic** view of they want to use their resources to study in the future.



A Hertzsprung-Russell diagram is used to plot stars based on their effective (surface) temperature and luminosity

Diagram by Richard Powell: https://commons.wikimedia.org/wiki/File:HRDiagram.png

Tables made by Ansh Menghani (author) using LaTeX

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

The Yerkes Classification system allows us to classify stars in different spectral and luminosity classes

Neural Networks

A neural network is a type of regression **algorithm** that takes certain inputs and can make **predictions** based off those. It is **trained** on data.

It adjust its algorithm based on **how close** the prediction the model makes is to being **correct** (evaluated using a **loss**, or **error** function).

This is similar to a human brain **learning from mistakes.**

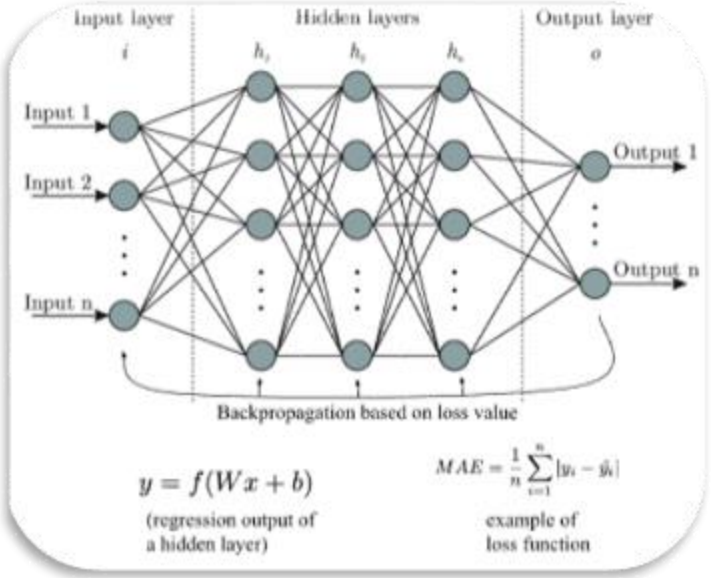
A Physics-Informed Neural Network **(PINN)** is a neural network designed to **respect** physics laws and formulas.

$$w_1 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}, \; w_2 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}, \; w_3 = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$

$$\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \\ w_1 & w_2 & w_3 & w_4 \end{bmatrix} \text{ becomes } \begin{bmatrix} \leftarrow w_1^{\mathrm{T}} \rightarrow \\ \leftarrow w_2^{\mathrm{T}} \rightarrow \\ \leftarrow w_3^{\mathrm{T}} \rightarrow \end{bmatrix}$$

Regression in neural networks is typically performed using linear algebra to allow higher-dimensional analysis of data

Jordan, J. (2017, June 29). *Neural networks: representation. Jeremy Jordan*, https://www.jeremyjordan.me/intro-to-neural-networks



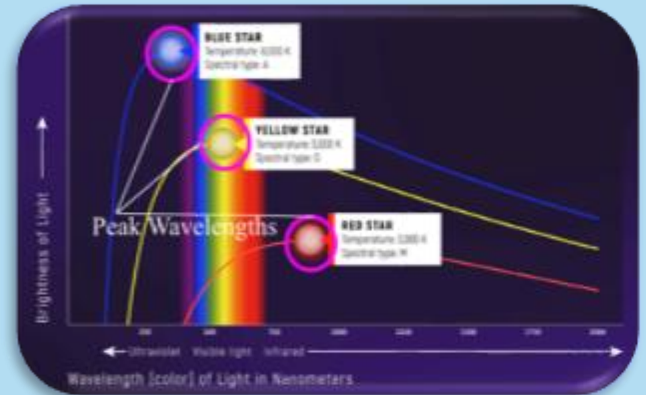
General structure of a Neural Network (inputs $\longrightarrow$ regression $\longrightarrow$ outputs $\longrightarrow$ algorithm update)

Figure by Bre, F., Gimenez, J. M., & Fachinotti, V. D. (2018): https://doi.org/10.1016/j.enbuild.2017.11.045

Stellar Parameters

Stellar parameters are the attributes of a star (i.e., its luminosity). The "big three" stellar parameters are Effective Temperature, Luminosity, and Radius because these values are easiest to gain data on and are related by one formula: $L = 4\pi R^2 \sigma T^4$ where σ is the Stefan-Boltzmann constant.

Finding the Effective Temperature:



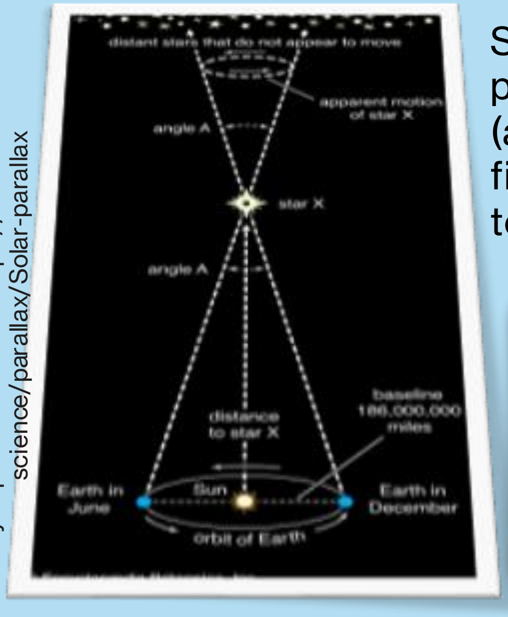
Source: JWST, Public domain – https://stsci-opo.org/STScI-01G7RQ0CRCSNKZNX18HNNAQRV0.jpg

Analyze target star's **spectra** to find the peak wavelength. Use Wien's law below to find effective temperature of the star:

$$\lambda_{max} = \frac{0.0029mK}{T}$$

Finding the Luminosity and Radius:

Encyclopedia Britannica: https://www.britannica.com/science/parallax/Solar-parallax


distant stars that do not appear to move
apparent motion of star X
angle A
star X
angle A
distance to star X
baseline 186,000,000 miles
Earth in June
Sun
Earth in December
orbit of Earth

How parallax measurements work

Step 1: Use parallax method (among others) to find the **distance** to the target

Once **luminosity** and **effective temperature** are calculated, **radius** can be calculated using the **formula above.**

Source: NASA, Public domain – https://www.nasa.gov/image-article/measuring-white-dwarf-star/

Step 2: Use apparent luminosity measurements and the distance to the star to find the **luminosity**

The Model

Goal

Develop a PINN that can **accurately model** many stellar parameters of any given star with **input values** of the star's effective temperature, luminosity, and radius.

Data Collection

- 15.5 million samples of data were selectively obtained from the Gaia Data Release 3, collected by the Gaia spacecraft.
- The data was normalized and split into training, validation, and testing data.

Gaia spacecraft

Image by ESA: https://dlmultimedia.esa.int/download/public/videos/2019/10/020/1910_020_AR_EN.mp4

Model Architecture

Outputs

- 5 parameters describing star's dimensions
- 13 parameters describing the star
- 3 classifications of the star (spectral and luminosity classes) and star type

- **Custom** input **recursion** allows for the model to increase the input size each consecutive output by **adding** the most recent output to the list of inputs.
- **Custom** validation loss **reward** encourages model to adjust its algorithm in a way such that it performs better on **new** data.
- Model is designed to **respect** physical formulas and laws governing stellar systems.
- Other methods are implemented to **prevent** overfitting, **improve** classification performance, and **transform** each layers' outputs to the desired format.

Image by Julien Despois from https://hackernoon. com/memorizing-is-not-learning-6-tricks-to-prevent-overfitting-in-machine-learning-820b091dc42

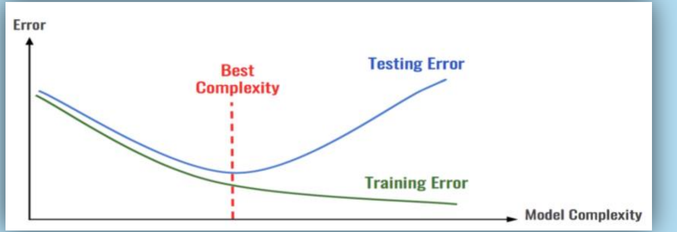
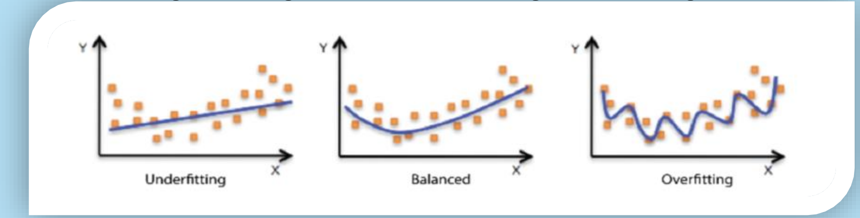


Image by AWS from https://docs.aws.amazon.com/machine-learning/latest/dg/model-fit-underfitting-vs-overfitting.html

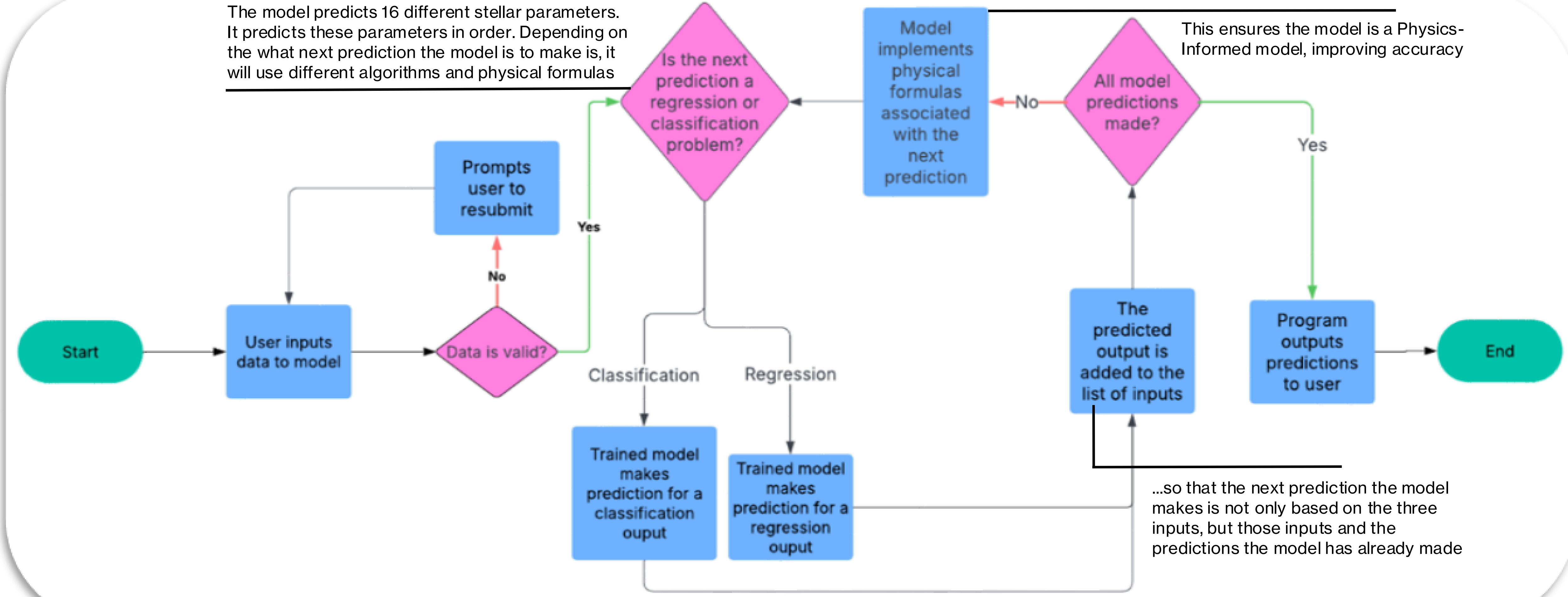


Overfitting/ Underfitting occurs when a model is too complex or too simple. A model will "memorize" training data when overfitted

Flowchart

The model predicts 16 different stellar parameters. It predicts these parameters in order. Depending on the what next prediction the model is to make is, it will use different algorithms and physical formulas

This ensures the model is a Physics-Informed model, improving accuracy



Start
User inputs data to model
Data is valid?
No
Prompts user to resubmit
Yes
Is the next prediction a regression or classification problem?
Classification
Regression
Trained model makes prediction for a classification ouput
Trained model makes prediction for a regression ouput
The predicted output is added to the list of inputs
All model predictions made?
No
Model implements physical formulas associated with the next prediction
Yes
Program outputs predictions to user
End

...so that the next prediction the model makes is not only based on the three inputs, but those inputs and the predictions the model has already made

Flowchart made by Ansh Menghani (author) using Lucid

Accuracy and Speed on HR Diagram

The model was **iteratively** trained such that it trained and tested itself by looking at data many times. The loss of a model at any point in represents how far away it is from a 100% true prediction during training.

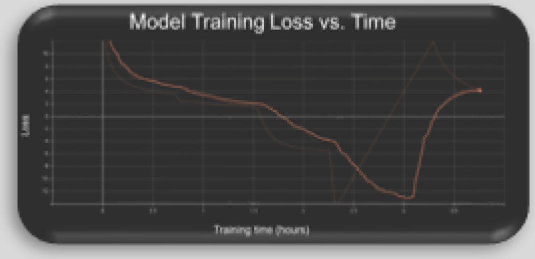


Figure 1: This represents the loss, or the model's error when fed training data each time it tested itself during training. This graph is smoothed to show the loss trend

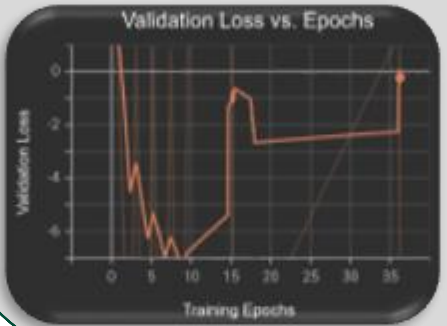


Figure 2: This represents the validation loss, or how well the model performed when tested with data that it hadn't seen during training. This graph has been smoothed to show the validation loss trend

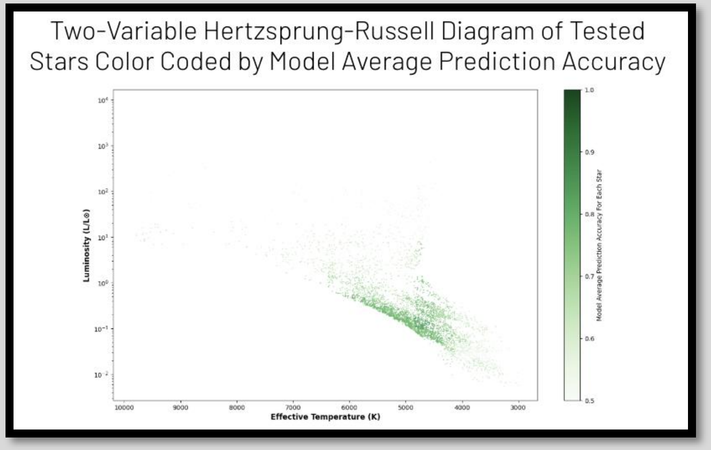


Figure 3: This represents the accuracy of each prediction the model makes and compares them to the star type of the star being analyzed (1.0 = 100%)

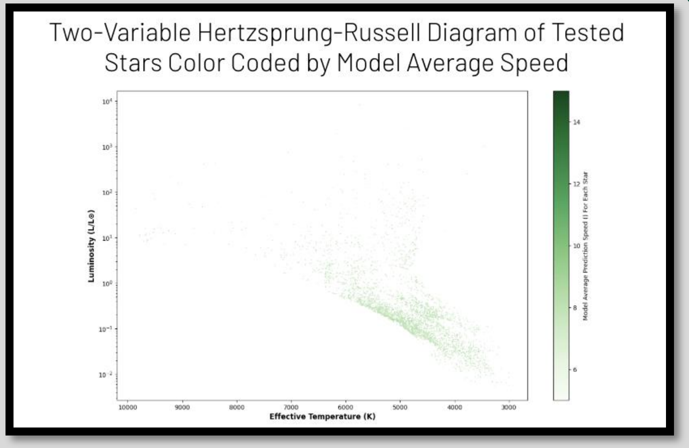


Figure 4: This represents the speed of each prediction the model makes and compares them to the star type of the star being analyzed (1.0 = 1ms)

Each datapoint is a **star** on the Hertzsprung-Russell diagram. The **darker** the dot, the **more** accuracy/speed the model exhibited when running predictions on this star.

Figures 1 and 2 made by Ansh Menghani (author) using TensorBoard. Figures 3 and 4 made by Ansh Menghani (author) using matplotlib in Python.

Parameter-Based Model Evaluation

The model's accuracy by parameter predicted is shown on the left. Analysis of classification outputs is to the right.

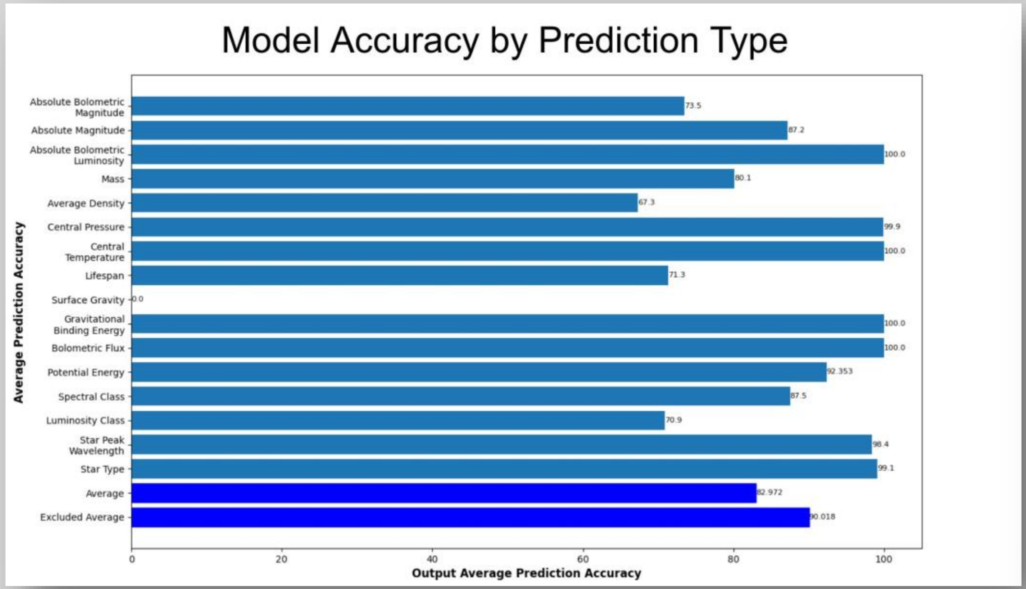
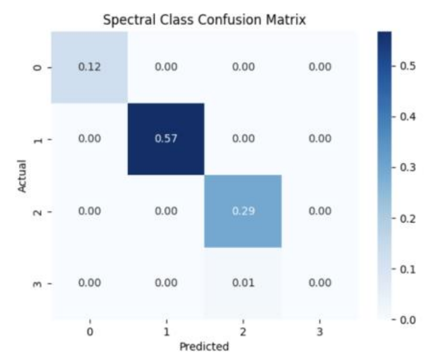
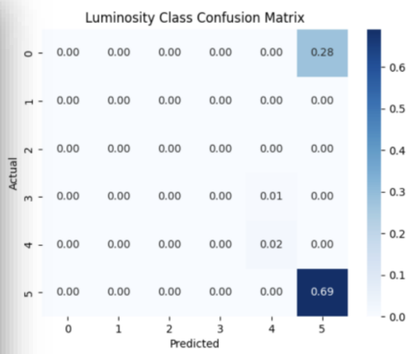


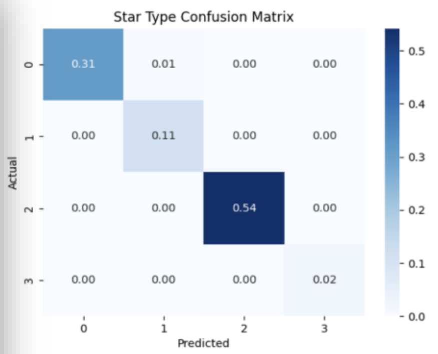
Figure 5: Percentage of 775,000 testing predictions made by the model that falls between ±5% of the true value by parameter



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 6, 7, and 8: 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 5 made by Ansh Menghani (author) using matplotlib in Python. Confusion matrices of Figures 6, 7, and 8 made by Ansh Menghani (author) using scikit-learn and seaborn in Python. Tables in Figures 6, 7, and 8 made by Ansh Menghani (author) using LaTeX.

Discussion

Refer to Figures 1 and 2

The model is attempting to obtain a loss and validation loss close to zero (figures 1 and 2), which indicates positive training behavior.

Loss and validation loss dip below zero due to the validation loss reward mentioned earlier. It is rewarding the model for improvements in validation loss by subtracting from the loss.

The model's performance is affected greatly by the star's effective temperature.

Refer to Figures 3 and 4

Main-sequence stars are modeled better than other types of stars such as:

- Stars that are not represented as much in the training dataset.
- Data-points/stars may be considered as outliers.

There is little to no variation of prediction speed.

Discussion (continued)

The low accuracy that is associated with very few parameters is likely due to how the model scales data.

Tiny errors can cause large accuracy gaps when values are small (e.g., 0.1 is close to 0.2 but a large percentage off).

Refer to Figure 5

The first nine outputs had four parameters with >85% accuracy, while the next seven only had one.

This means that the custom method used to recurse outputs into inputs (as described earlier) works!

Refer to Figure 5

Overall classification scores were very good.

The low luminosity class scores could be because it was the only parameter with six possible outputs, while the other classification parameters had four each. This may have interfered with the model's algorithm.

Refer to Figures 6, 7, and 8

Final Model Accuracy:

90.018%

Average Prediction Speed:

8ms

Conclusions

Real-World Applications

- This project further **expands** machine learning and deep learning into the field of physics as a whole
- This tool can be used to **simulate** star systems unfathomably far from Earth that are very **hard to gather data on**
  - It's the **first** stellar simulation model that does the job of **over sixteen models** (i.e., if each stellar parameter were to be modeled by a separate model)
  - The model has **better accuracy** than current models for many of the modeled parameters and is **16 times faster** than if each parameter was modeled separately. This will save tremendous amounts of time when modeling large datasets
- This model can be used to **find previously unknown relations** between different stellar parameters that may not currently share a simple mathematical relation
- The machine learning techniques used in this model (specifically the custom ones) can be **replicated in other projects** across the field of machine learning and neural networks to **greatly improve** them
- Simulations will help scientists **confirm** their observational results and help them **decide** whether a patch of the sky has something interesting to study with **minimal input data**

Goals Met?

The goals described earlier have been met by the model! ✓

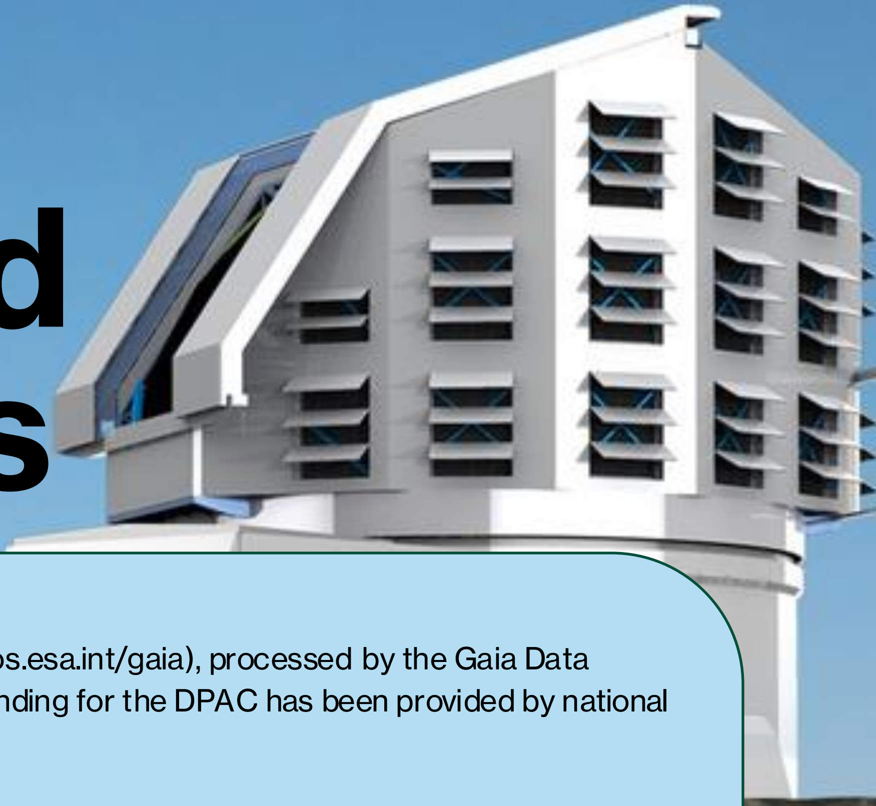
Possible Limitations

- Some parameters have a lower modeling accuracy than others
- The training dataset has less examples of rare stars (such as white dwarfs), reducing the accuracy of predictions on those types of stars

Future Goals

- Improve accuracy by further experimenting with model structure
- Gather more examples of rare stars in the dataset to improve modeling of those stars
- Develop a better user interface

Key References and Acknowledgements



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 Processing 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/const ants.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.

Background Image (Vera C. Rubin Observatory) from Caltech IPAC:
https://www.ipac.caltech.edu/system/activities/images/28/large/Render4-full-1.jpg