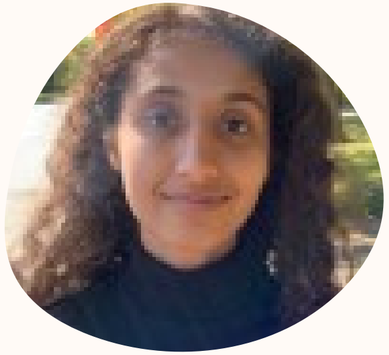
Project Walkthrough

About Us

Bahar Chidem
Software Eng Specialist
@UOFT

Vaibhav Lakshmi Santhanam
Software Eng Specialist
@UOFT



Tara Jorjani
Software Eng Specialist
@UOFT

Arina Azmi
Information Systems Specialist
@UOFT

Celestial Labelling

multi-class classification model that identifies celestial objects found in deep space

problem statement

Only 5% of the universe is known to scientific researchers, with much of the celestial bodies found in deep space unidentified.

By generating further classifications of celestial objects, astronomical research can be advanced significantly to improve our understanding of the cosmos.

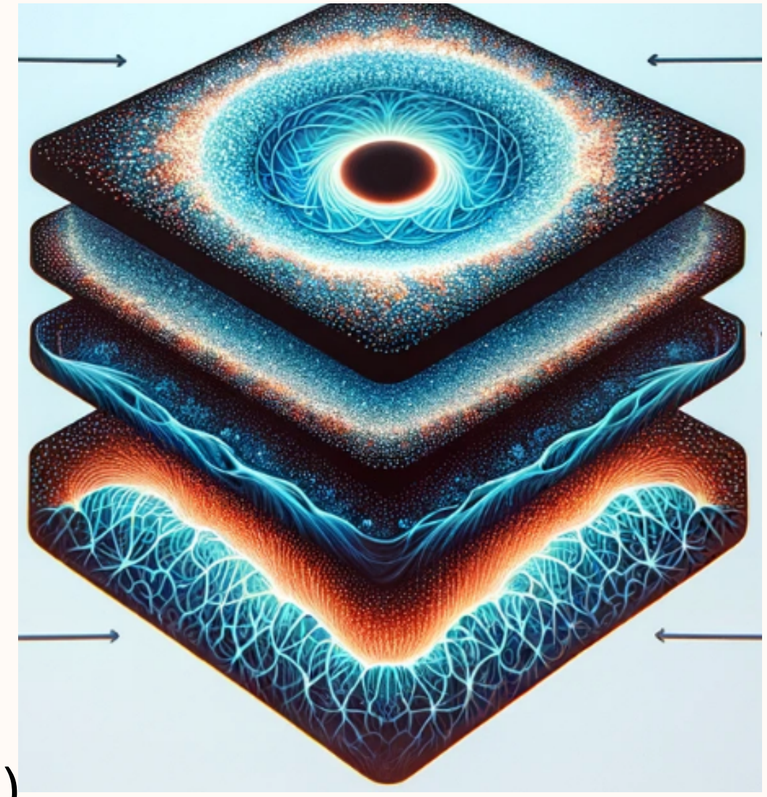
Solution

Our Celestial Labelling model addresses this problem by using multi-class classification to distinguish between different types of celestial entities. Based on a neural network model, it identifies numerous celestial objects with high accuracy based on a number of optical factors.

The Celestial Labelling model is a valuable asset in expanding scientific and academic research of the vast cosmos and the mysteries surrounding it.

Version 1: Neural Network part 1

1. Input Layer
    - Receives standardized values of features like `alpha`, `delta`, the five spectral band measurements (`u`, `g`, `r`, `i`, `z`)
2. Hidden Layers
    - neurons that process the inputs from the previous layer and pass them on
    - uses weighted sum & activation function (ReLU) to introduce non-linear processing.



Version 1: Neural Network part 2

3. Output Layer
    - uses the softmax activation function to output a probability distribution across the possible classes (Galaxy, Star, Quasar, etc.)

4. Back-propagation and Optimization
    - adjust its internal weights based on the error in its predictions
        - tensorflow/Keras
    - Adam optimizer to compute adaptive learning rates for each parameter.

Version 2: Neural Network - more features

New Additions:

Early Stopping and Model Checkpointing

```
# Early stopping and model checkpointing to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=1)
model_checkpoint = ModelCheckpoint('best_model.keras', monitor='val_loss', save_best_only=True, verbose=1)
```

This prevents overfitting through the monitoring of the validation loss.

- No improvement during 10 epochs -> training stops early and the best model is saved

Process helps to improve the accuracy of our model.

Citations:

https://machinelearningmastery.com/multi-class-classification-tutorial-keras-deep-learning-library/

https://hackernoon.com/multiclass-classification-with-keras

https://python.plainenglish.io/neural-network-multiclass-classification-model-using-tensorflow-67ec2c245d0e

https://machinelearningmastery.com/k-fold-cross-validation/

Thank you