Identifying Component Criteria for Spectroscopic Binary Stars

Samuel Lilek[1], Rami Nasser[2]
[1]lilek.sw@gmail.com,[2]raminass@gmail.com

Feb 2022

Abstract

The study of binary star systems, like the study of all other stars, is largely dependent on spectrum analysis. For multi-star systems, however, the flux emitted by each star combines with its neighbors; therefore, it is impossible to determine a star's properties without disentangling the system's combined spectrum into component spectra for each star in the system. Currently, this process is able to be achieved through a laborious process that requires multiple observations of the same star system. In this project, we use state of the art deep learning architectures to predict these properties for each star using only a single observation of their combined flux as a model input.
Code: https://github.com/raminass/binary_stars

1 Introduction

One of the most challenging aspects of studying the stars is also the most obvious: stars are very far away. The colossal distances that separate the stars of the night sky from our home on Earth undoubtedly contribute to the sense of awe that many feel when gazing at the night sky, a sense of awe that has inspired humans since the time of antiquity.

Fortunately, our modern understanding of stars has benefited largely from the ability to read and analyze the star's electromagnetic spectrum. This spectrum, which measures the flux relative to the frequency component of the energy emitted by the star, contains an enormous amount of information about the physics in the star's atmosphere: chemical composition, temperature, surface gravity, rotation rate, and many other details.

However, there is another challenge lurking, a challenge that is masked from the casual observer by the enormous distances between Earth and the stars: many of the stars we observe at night are not single stars, but multi-star systems. This fact causes even the modern tools of spectrum analysis to be a difficult task, because the component spectra of the multiple stars in the system combine with each other.

Through multiple observations and laborious mathematical processes, astronomers are currently able to "disentangle" the combined spectrum of multi-stars systems in order to get the individual spectrum for each distinct star in the system [2]. Our innovation on this project was to perform this estimation using deep learning, with a single observation of the combined spectrum. From a large synthetic dataset of binary star systems, we aimed to develop a model – or models – that could predict six critical star features: temperature, metallicity, v_sin_i, $\log g$, luminosity, and alpha.

Our method was to build a loss function and output normalization scheme that would allow for us to train a single model to predict all six of our desired parameters. Then, we began testing model performance on three basic model frameworks: multi-layer perceptron (MLP), attention-based, and 1-D convolutional. For each of these basic frameworks, we experimented with a number of different architectures. These experiments included, but were not limited to: varying the number of network layers, varying the size and dimension of the layers, adjusting the relevant kernel sizes, adjusting the learning rate and learning rate schedule, varying our normalization method, and a number of other hyperparameter tuning tasks. One of our main tasks was to experiment with using a single network to produce all 12 desired outputs (six outputs for each star in the binary system) or to create multiple networks that predicted for less parameters. As we built our single network output, we also developed our second major innovation: normalizing the target parameters. This method allowed our loss function to train towards all target parameters despite the varying magnitudes of these parameters.

2 Related Work

Deep-learning based methods are being used for various astronomical purposes, among them periodic transits detection [7], exoplanet discovery in low signal to noise ratio (S/N) direct imaging [5], and object of interest detection in spectroscopic surveys [4]. In the context of SB2 analysis, a Convolutional Neural Network (CNN) has been used to detect SB2s based on a single exposure medium-resolution spectra [6].

This paper presents a feasibility study of deep learning to extract parameters for both stars in binary systems, using the combined spectra as features. Using simulated synthetic data sec. 3, we tested, trained and validated multiple network architectures capable of predicting the stars parameters from a single observation.

3 Data

We received our data from Professor Shay Zucker from the Tel Aviv University Department of Geophysics. All of our data is synthetic; it was generated by Professor Zucker and his team, including Avraham Binnenfeld, by using the PHOENIX spectral library [3].

The data consists of approximately 80,000 observations. Each observation consists of a combined spectrum and 12 target values: temperature, metallicity, log_g, luminosity, alpha, and vsini for both stars in the system. A sample combined spectrum can be observed in Figure 1. After processing this dataset for model development, we were left with a resulting dataset consisting of an input flux matrix, $X_{n\times f}$, and a target variables matrix $Y_{n\times v}$, where n is the number of samples, f is the length of the flux profile, and v is the number of target values.

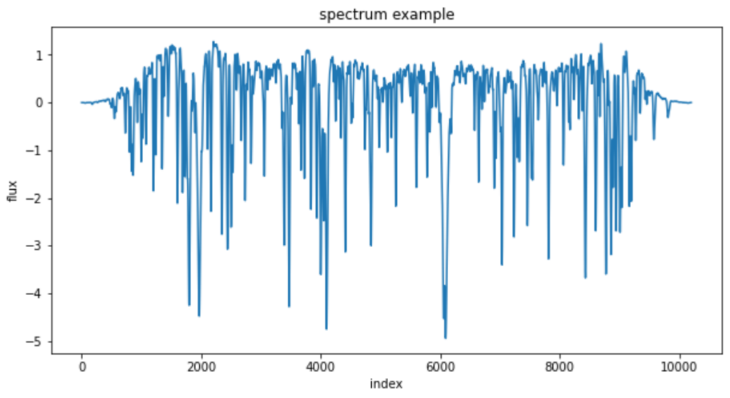

Figure 1: Sample Combined Spectrum

4 Methods

4.1 Problem Formulation and Current Approaches

Ultimately, our intention was to build a model to predict a set of target values, Y_i, for a given combined spectrum, X_i. Our method was to first develop a loss function that would satisfy this task; given that we wanted to try to predict for a variety of different outputs with a single network, the development of this loss function was a non-trivial process. After developing our loss function and training architecture, we checked a variety of modeling options, with a goal of finding a single model that was able to effectively predict for all 12 outputs. We uses the Pytorch framework for implementing our models and processing the data.

4.2 Loss Function

The goal of the loss function is to direct the network to optimize towards the right outputs. As this is a regression modeling task, we elected to use a mean squared error (MSE) loss function. Given the data and desired goal, however, our model needed to be agnostic to the order of the stars. This presented a complicating factor for our loss calculation. For example, the data may include an observed combined spectrum X_i consisting of $star_1$ and $star_2$. In this case, it is acceptable for the model to output the target values as ($star_1$, $star_2$) or ($star_2$, $star_1$) - the order does not matter, as both predictions correctly predict components for two stars in the system. Given this constraint, we added an extra step to the loss function that evaluates the prediction against the target outputs in either order, and then used the minimum loss from these two values:

$$\underset{\hat{Y} \in \{\hat{Y}_1 || \hat{Y}_2, \hat{Y}_2 || \hat{Y}_1\}}{min} \quad MSE(Y, \hat{Y}) \tag{1}$$

where $\hat{Y}_i$ is the predicted parameter vector for star i and $||$ is the concatenation of vectors.

4.3 Output Normalization

An additional innovation we used on this project was output normalization. While normalization of model inputs is a fairly standard process in deep learning (batch normalization, for example), normalization of target outputs for training appears to be relatively unused. In our case, normalization seemed like the best approach to solve the problem of multiple target outputs at varying scales. The most obvious example of this is luminosity: the target luminosity values are on a scale of 1×10^9 to 1×10^{11}. None of the other target values are anywhere near the same magnitude: temperature is on a scale of 1×10^4, but all other values are less than 10. Without some sort of normalization or weighting, the MSE loss for out combined target function would have optimized only to predict for luminosity. We decided to attempt to normalize all values as a way to ensure that none of the targets were focused on by the model to the exclusion of other desired parameters.

To validate this approach, we first attempted to use a simple MLP network to compare results for models trained on normalized and unnormalized data. For all six parameters (temperature, luminosity, v_sin_i, $\log g$, alpha, and metallicity), two identical two-output ($star_1 output$, $star_2 output$) models were created. One model was trained on normalized output data, the other trained on unnormalized output data. By breaking this task up into six different models (for each of the six targets) we were able to train on unnormalized data without preferentially optimizing towards one parameter or another. Once the normalized model was trained, we then re-tested it on unnormalized data by using the mean and standard deviation that were initially calculated during normalization. This allowed for a direct comparison of performance between normalized and unnormalized models.

All models were trained for 35 epochs, with results shown in Table 1. As can be observed, the normalized-output model outperformed the regular model (achieved a lower loss on the test set) for all six parameters. To be clear: we are not suggesting that output normalization will always have such strong results. The goal of this experiment was simply to validate our output normalization approach before using during development of our single model.

	Type of Output Used During Training	
Data Evaluated	Unnormalized	Normalized
Temperature	448244	381996
Metallicity	0.082	0.079
log_g	0.349	0.322
Luminosity	3.21E21	3.14E21
Alpha	0.065	0.056
v_sin_i	4.723	4.562

Table 1: Evaluation of Output Normalization: Total Loss

4.4 Tuning

To further improve our MLP mode, we used the Optuna package for hyperparameter tuning [1]. For the MLP model, we tuned the following parameters: learning rate and size of hidden layers. We also used Optuna during experimentation with the the attention-based model; we tuned the following parameters: learning rate, dropout rate, number of heads, and number of attention layers. In each case, the Optuna package evaluated 10 different trials over the possible values of the tunable parameters. The results were then validated on a held-out validation set consisting of 20% of the data.

4.5 Model Experimentation

Once we had confirmed the feasibility of training a functional network using output normalization, we proceeded to assess a several different neural network architectures on the problem. Primarily, our efforts centered on three architectures: MLP, attention-based, and 1-D convolutional. We saw some success with all three models, as further discussed in the paragraphs below. However, our highest performing model was our convolutional model. We attribute this to the fact that it was able to take spatial information into account.

MLP Model. This was the baseline model, with three fully connected layers. It was also the same architecture that was used to assess our loss function and output normalization, as discussed above. Our architecture consisted of a layer of 400 fully connected units, followed by 100 fully connected units, followed by a 12-unit output layer. All hidden layers used ReLU nonlinearities. This architecture is displayed in Table 2.

Layer Name	Type	Input Dim	Output Dim	Activation Function
Linear1	FC	10197	400	ReLU
Linear2	FC	400	100	ReLU
LinearOut	FC	100	12	none

Table 2: Fully-Connected Model Architecture

Attention Model. The attention model was a much larger network than the simple MLP network discussed above. The architecture consisted of two fully connected layers of 500 and 128 hidden units, followed by seven multi-headed attention (MHA) blocks, followed by a 50-unit hidden layer and a final 12-unit output later. This architecture is displayed in Table 3.

Each MHA block started with a multi-headed self attention unit of four heads. Each self-attention head was of size 128: the input to the attention block, the size of the key, query, and value matrices, and the size of the output were all 128 (or 128 by 128). The four-headed attention block was followed by a fully connected layer to reduce the dimensions back to their original size (128), followed by a ReLU nonlinearity. Additionally, we added a residual connection to the end of each block - initial testing showed that this was essential to for model training.

Layer Name	Type	Input Dim	Output Dim	Activation Function
Linear1	FC	10197	500	ReLU
Linear2	FC	500	128	ReLU
MHA1	Attention	128	128	ReLU
MHA2	Attention	128	128	ReLU
MHA3	Attention	128	128	ReLU
MHA4	Attention	128	128	ReLU
MHA5	Attention	128	128	ReLU
MHA6	Attention	128	128	ReLU
MHA7	Attention	128	128	ReLU
Linear3	FC	128	50	ReLU
LinearOut	FC	50	12	none

Table 3: Attention-Based Model Architecture

Convolutional Model. The convolutional model consisted of five 1-dimensional convolutional layers, followed by two fully connected layers. The first four convolutions used a 5×5 kernel size, with each layer doubling the number of channels (except convolutional layer 4, which maintained 64 channels for both input and output). The fifth convolutional layer used 1×1 kernel and reduced the number of channels from 64 down to 5, to reduce the number of parameters before the flattening layer. After flattening, the model went through a 100-unit hidden layer before a final, 12-unit output layer. ReLU linearities were used throughout this architecture. This architecture is displayed in Table 4.

Layer Name	Channels In	Channels Out	Kernel Size	Stride	Activation Function
Layer1: Conv1	1	16	5	1	ReLU
Layer2: Conv2	16	32	5	5	ReLU
Layer3: Conv3	32	64	5	5	ReLU
Layer4: Conv4	64	64	5	5	ReLU
Layer5: Conv5	64	5	1	1	ReLU
Flatten All Five Channels For Input to FC Layers					
Layer Name	Type	Input Dim	Output Dim	Activation Function	
Layer6: Flatten	Flatten	5	1	none	
Layer7: Linear1	FC	410	100	ReLU	
Layer8: LinearOut	FC	100	12	none	

Table 4: Convolutional Model Architecture

5 Results

We compare the different architectures based on our loss function eq. 1, the comparison is by variable.

5.1 Model Comparison

We concluded that the 1-D convolutional model was the most effective approach for dealing with this problem.

The first architecture tested, an MLP model of exclusively fully-connected layers, was surprisingly strong for full, 12-factor model development. This is especially true given that we did not expand the size of the model at all: it was exactly the same as what was used in our 2-factor models. That being said, we were able to get stronger performance with the more robust attention and convolutional based models.

While the performance of our self-attention based architecture exceeded that of the MLP model, it simply did not provide the performance boost that we expected, given the computational cost. Given the large number of parameters that were added to the model with the seven MHA blocks and extra fully-connected layers, we elected to continue testing and evaluate with a convolutional model.

Our convolutional model performed significantly better than all of our previous architectures. As can be seen in Figure 3, it converged faster and achieved better performance. This result gave us a few important takeaways. First, it established the continued relevancy of convolutional layers in deep learning. Much is made of transformers and attention layers as being the solutions to any and all deep learning problems. While this may be true in many cases, there is no doubt that convolutional layers are still able to provide excellent performance with just a fraction of the learned parameters.

Additionally, this showed the importance of taking positional information into account. The MLP and attention-based models that we used did not have any architectural method for taking in positional information. By contrast, by the nature of convolutions and their associated receptive fields, the convolutional network was able to train on positional information. We believe this was a key factor in its improved performance. One future test to assess this would be to re-train our attention-based model with some form of positional encoding.

The convergence graphs showing training performance from all three models can be seen in Figure 2. A simplified graph showing the test set performance of all three models can be seen in Figure 3.

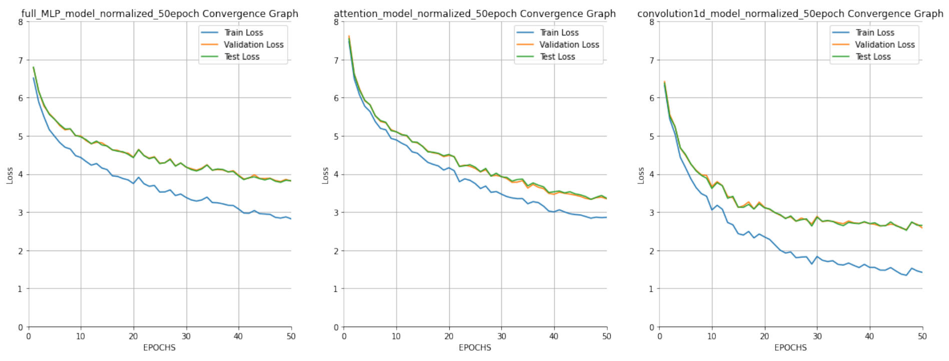

Figure 2: MLP, Attention-Based, and 1-D Convolutional Performance

5.2 Final Training and Results

As a final step, we trained the convolutional model on a much larger portion of the dataset. The results of this model training can be observed in Figure 4.

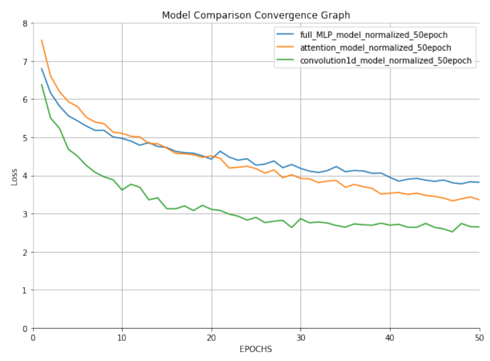

Figure 3: Test Set Performance Model Comparison

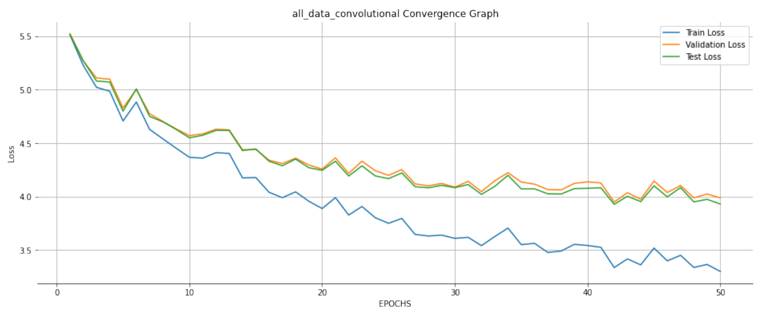

Figure 4: 1-D Convolutional Model Trained on Larger Dataset

Unfortunately, we did not find that increasing the size of the dataset led to improved loss scores. We believe this is due to the fact that the section of the 80,000 sample dataset that we used for our initial experiments - consisting of about 10,000 samples - was likely not fully representative of all of the data. As a result, we most likely overfit to this section of the data (and its associated test set) with our experimental models. Additionally, given our the output normalization that our model used, using data that was not representative of the full dataset was certain to have ripple effects. It would have the common effect of overfitting: we would be too closely trained to the sample that we used. But it would also have the secondary effect of changing the scale parameters calculated during normalization of our dataset: with a new set of data, we also get a new mean and standard deviation that we need to use for our output normalization.

Ultimately, this does not change the results of our previous experiments - it simply validates the need to train on as wide a dataset as possible, in order to deliver the most generalizable and effective model to the end user.

6 Conclusion

Our experiments determined the feasibility of disentangling and predicting star parameters from a single observation of a binary star system's combined spectrum. In the field of astronomy, this ability has the potential to save astronomers time and effort as they predict and study multi-star systems. Future research should be done to expand this work into further areas of this research. For example, our work would seem to suggest that it should be possible for a neural network to estimate the number of stars that are in a given spectrum, or to provide individual star parameters for multi-star systems with more than two stars.

Additionally, our project was able to experiment with output normalization. With output normalization, we were able to train a single network to assess multiple criteria of vastly different magnitudes. Our initial experimentation suggested that this could be done at no cost to model performance; however, additional experimentation should undoubtedly be done to confirm this. Furthermore, we were able to assess the effectiveness of 1-D convolutions; the ability of convolutional layers to evaluate data based on positional information was essential to the superior performance of this architecture compared with our other models.

References

[1] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[2] W. G. Bagnuolo Jr and D. R. Gies. Tomographic separation of composite spectra-the components of the o-star spectroscopic binary ao cassiopeiae. *The Astrophysical Journal*, 376:266–271, 1991.

[3] T.-O. Husser, S. Wende-von Berg, S. Dreizler, D. Homeier, A. Reiners, T. Barman, and P. H. Hauschildt. A new extensive library of phoenix stellar atmospheres and synthetic spectra. *Astronomy & Astrophysics*, 553:A6, 2013.

[4] P. Škoda, O. Podsztavek, and P. Tvrdík. Active deep learning method for the discovery of objects of interest in large spectroscopic surveys. *Astronomy & Astrophysics*, 643:A122, 2020.

[5] K. H. Yip, N. Nikolaou, P. Coronica, A. Tsiaras, B. Edwards, Q. Changeat, M. Morvan, B. Biller, S. Hinkley, J. Salmond, et al. Pushing the limits of exoplanet discovery via direct imaging with deep learn-

ing. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 322–338. Springer, 2019.

[6] B. Zhang, Y.-J. Jing, F. Yang, J.-C. Wan, X. Ji, J.-N. Fu, C. Liu, X.-B. Zhang, F. Luo, H. Tian, et al. The spectroscopic binaries from lamost medium-resolution survey (mrs). i. searching for double-lined spectroscopic binaries (sb2s) with convolutional neural network. *arXiv preprint arXiv:2112.03818*, 2021.

[7] S. Zucker and R. Giryes. Shallow transits—deep learning. i. feasibility study of deep learning to detect periodic transits of exoplanets. *The Astronomical Journal*, 155(4):147, 2018.