COSC440 RESEARCH REPORT

JAKE FAULKNER

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 METHOD

In this report two models are evaluated. The first is a convolutional autoencoder (Figure 1 on page 8) and the second a recurrent autoencoder (Figure 2 on page 9). The convolutional encoder is composed of a number of 1d convolutional layers, ten 3×3 filters per layer, along with another set of single filter 3×3 layers that convert the filters into a vector in the latent space. Max pooling layers were added to add some time translational invariance to the model. The convolutional decoder is effectively the mirror image minus the max pooling layers. The part three convolutional model is the same as the part two model, but will instead be trained using a loss function that encourages fitting the transit least squares model rather than light curve reconstruction. The recurrent autoencoder is composed of GRU layers with 32, 16 and 8 activation units, and is generally a lot simpler than the convolutional layer. Both the convolutional and recurrent networks have dropout enabled to prevent overfitting.

The models were trained with detrended subsamples of stellar light curves from 72 star systems with published transiting planets with known transit periods and planetary masses around 1 M_j. The `MODELS` folder contains one file for each stellar light curve used in training. Each light curve was fitted using the `transitleastsquares` python package and (upon checking the results parameters matched with the published results) the model's curves were saved along with the light curve samples. By aligning the model with the light curve samples we produce a light curve and model pair. Both of these curves will be used to train the autoencoder. In addition to alignment all curves and models were translated down and normalised so that each was centred at zero and in the range $[-1, 1]$. The model is trained by feeding the light curve into the autoencoder and then measuring performance using the loss function in Eq 1, which is a linear combination of the lightcurve reconstruction loss and the model fit. The hyperparameter α is used to control how much the autoencoder attempts to reconstruct the lightcurve ($\mathbf{L}$) vs how much it attempts to fit the model ($\mathbf{M}$). Whilst various values of alpha in the range of $0.2 \leqslant \alpha \leqslant 0.4$ were trialed, $\alpha = 0$ was used for the final models.

$$L(\mathbf{X}, \mathbf{L}, \mathbf{M}) = \alpha(\mathbf{X} - \mathbf{L})^2 + (1 - \alpha)(\mathbf{X} - \mathbf{M})^2 \tag{1}$$

All models are trained with an 80-20 train-test split, with a further 20% of the training data being used for validation, and early stopping enabled. The Adam optimiser with the default learning rate of 10^{-3} is used for all models. Final testing was performed by manually reviewing the end to end model performance on a further 14 stars in the HAT-P family, the results of which will be discussed in the quantitative results, comparing the χ^2 and CDDP metrics, and the output are used qualitatively as well.

The intention of the changes made to the loss function is to investigate whether an autoencoder can model physical features of stellar transits, and to compare and contrast the recurrent and convolutional autoencoder to see which can learn a better representation. The hypothesis is that the network should be able to learn *something* about the physical properties of the network, though what in particular may be hard to tell.

2 RESULTS AND DISCUSSION

Both models achieved reasonable results, with the convolutional model being easier to train and test on account of the parallelisation advantages, and in general both models achieved a test loss of $\leqslant 0.0500$, with the end-to-end testing demonstrating that both the part two and part three models could improve measures of model fitness.

2.1 Convolutional Autoencoder (With Original Loss Function)

The original model was trained for 50 epochs with $\alpha = 1$ on the custom loss function, with the results measured in Table 1. By setting $\alpha = 1$ we are asking the model to do a simple reconstruction fit. In Figure 3 on page 9 we can see the output on system we will use for comparing all the models, HAT-P-14.

Table 1: Table of test systems fit using the part two autoencoder.

Published		Part Two Autoencoder		
System	Pub. Period (days)	Period (days)	χ^2	CDPP
HAT-P-2	5.633	5.635	8246.245	31.623
HAT-P-3	2.900	2.900	5437.187	48.910
HAT-P-4	3.057	3.055	7022.547	75.697
HAT-P-7	2.205	2.198	1621.681	34.336
HAT-P-9[a]	3.076	—	—	—
HAT-P-11	4.888	4.863	6851.965	19.332
HAT-P-12	3.213	3.214	12593.719	180.890
HAT-P-14	4.628	4.625	9076.318	28.171
HAT-P-16	2.776	2.778	6150.352	48.170
HAT-P-17	10.338	5.180	12522.630	63.052
HAT-P-18	5.508	5.510	16423.212	263.405
HAT-P-21	4.124	4.124	6152.725	43.448
HAT-P-22	3.212	3.211	5322.576	65.932
HAT-P-24	3.335	3.354	6066.576	55.993

a After processing the TLS method returned a `NaN` period.

Comparing this to the recurrent and convolutional autoencoder it is fair to say that this model has learned to remove noise in the light curves. The quantitative results put it somewhere around the recurrent autoencoder in terms of TLS model fitness, but qualitatively the part two model produces cleaner fits than either part three (compare Figure 4 on page 10 to Figure 8 on page 11 or Figure 6 on page 10). In particular both part three models have a habit on producing strange transit curves outside the original transit period. The simple reconstruction model trained for part two does not have this disadvantage, though the other two models were trained to specifically fit the transit model itself rather than the original curve, so it's a little hard to compare them like this.

2.2 Recurrent Autoencoder

The Recurrent Autoencoder was trained for 50 epochs and achieved a final test loss of 0.451. The network was tested on the HAT-P light curves, and we can compare the fit on the original light curve (Table 2 on the next page) and the network's light curve in Table 3 on the following page. It can be seen that despite the projection down to a latent space of eight dimensions a remarkable amount of the transit information is preserved, though the parameters of the transit on the fit for the model have changed slightly.

Table 2: Table of test systems fit transit least squares (TLS).

Published		Transit Least Squares		
System	Pub. Period (days)	Period (days)	χ^2	CDPP
HAT-P-2	5.633	5.635	13587.877	281.804
HAT-P-3	2.900	2.900	11792.605	620.021
HAT-P-4	3.057	3.055	14314.521	1030.751
HAT-P-7	2.205	2.206	600.602	1030.751
HAT-P-9	3.076	3.925	16272.129	935.067
HAT-P-11	4.888	4.863	4468.206	27.225
HAT-P-12	3.213	3.214	15681.259	1400.956
HAT-P-14	4.628	4.624	14168.422	338.397
HAT-P-16	2.776	2.775	14041.914	634.241
HAT-P-17	10.338	5.171	11990.795	317.457
HAT-P-18	5.508	5.507	12695.560	1446.413
HAT-P-21	4.124	4.126	16115.531	622.602
HAT-P-22	3.212	3.211	3353.128	238.176
HAT-P-24	3.335	3.354	12886.855	891.681

Table 3: Table of test systems fit using RNN Autoencoder and TLS.

Published		RNN		
System	Pub. Period (days)	Period (days)	χ^2	CDPP
HAT-P-2	5.633	5.626	2837.260	29.433
HAT-P-3	2.900	2.892	2596.672	64.145
HAT-P-4	3.057	3.055	4324.447	78.005
HAT-P-7	2.205	2.206	3703.990	36.485
HAT-P-9	3.076	0.723	10735.801	2896.444
HAT-P-11	4.888	4.863	2909.167	15.339
HAT-P-12	3.213	3.210	6916.197	180.230
HAT-P-14	4.628	4.625	9101.431	34.027
HAT-P-16	2.776	2.778	10117.060	75.077
HAT-P-17	10.338	5.223	11394.868	58.236
HAT-P-18	5.508	5.500	5602.0734	255.266
HAT-P-21	4.124	4.124	8887.875	52.802
HAT-P-22	3.212	3.216	7693.480	57.269
HAT-P-24	3.335	3.359	5515.483	69.190

Apart from some outliers like HAT-P-17 (where neither CNN nor RNN got it right), and HAT-P-9 (where the processing destroyed the transit) the RNN model generally speaking agrees with the published results to within about two decimal places. Drilling down into specific systems we can observe some characteristics of the model and hypothesise how exactly the model is fitting the dataset.

In Figure 5 on page 10 we can see what the autoencoder has done with HAT-P-14, in a graph comparing the original light curve with the output processed by the model. One immediate thing to notice is that the autoencoder has removed nearly all the peaks in brightness. This is to be expected as the TLS model output the autoencoder was trained on assumes that when the planet is not transiting in front of the star it has a constant flux output. The RNN appears to produce a periodic signal, which is also expected as the transit least squares model also produces a periodic signal, so we might conjecture that the model learns the period of the transit from the input curve. Outlier data points are generally not reproduced, though oddly enough a number are — the criteria describing what outliers are represented is something of a mystery.

Figure 6 on page 10 shows a side by side comparison of the light curve folded on the predicted period for the original and processed curves. The model's output has some fascinating properties. The obvious phenomena is these "phantom transits", transit like dips in relative flux that appear consistently in the model output but do not correspond to the primary transit identified by the TLS algorithm. One conjecture on why those appear could relate to the dataset the model was trained on, the model was trained with at least 50% of the windows having at least one transit in them. One might conclude therefore that the model has learned to produce a transit in every window to reflect the fact that it likely did in the training data. An interesting direction of further research could be playing with the balance of transit and non-transit windows to remove these phantom transits.

In Figure 7 on page 11 we take the most obvious failure of the model in HAT-P-9. HAT-P-9 is an outlier in that the fit was horrible to begin with in the original curve. It appears that because the dip was subtle (and because the training data perhaps didn't have enough of these close calls), the model did a poor job. Instead, the model fit a very predictable periodic function that turned out to be a poor guess of the transiting planet.

2.3 CNN Autoencoder

The convolutional autoencoder was trained for 50 epochs with a final test loss of 0.0379. Testing on the sample lightcurves produces results in Table 4. Comparing the CNN results with the RNN results in Table 3 on the previous page generally shows that the convolutional autoencoder produced better results, and this is borne out in the qualitative analysis. Why the convolutional autoencoder performed better probably comes down to the fact that the RNN may require more data to generalise properly (pretty aggressive dropout was required to prevent overfitting), and the fact that RNNs require more training time to learn the parameters of the system. The advantages of the recurrent autoencoder approach would be that provided you could get it to train, you could generalise to arbitrary window sizes, not easily done with the convolutional model.

Table 4: Table of test systems fit using CNN Autoencoder and TLS.

Published		RNN		
System	Pub. Period (days)	Period (days)	χ^2	CDPP
HAT-P-2	5.633	5.626	2939.543	27.274
HAT-P-3	2.900	2.906	6405.919	44.018
HAT-P-4	3.057	3.055	2202.421	78.356
HAT-P-7	2.205	2.206	1221.368	26.062
HAT-P-9[a]	3.076	—	—	—
HAT-P-11	4.888	4.863	1833.507	14.914
HAT-P-12	3.213	4.617	10729.521	30.368
HAT-P-14	4.628	4.625	9101.431	34.027
HAT-P-16	2.776	2.778	8568.362	83.748
HAT-P-17	10.338	5.171	10019.102	45.587
HAT-P-18	5.508	5.519	3942.967	187.247
HAT-P-21	4.124	4.124	5551.942	41.839
HAT-P-22	3.212	3.211	5299.401	48.0139
HAT-P-24	3.335	3.354	3766.298	55.894

a HAT-P-9 after processing returned a `NaN` result.

Looking at HAT-P-14 in Figure 8 on page 11 we can see that, compared to the RNN, the convolutional autoencoder appears to produce a curve that sticks closer to 1 when no transit occurs but is more noisy. The convolutional autoencoder also

still produces the phantom transits. Both models also selectively remove outliers, and the reason for this is still quite unclear.

2.4 Exploring Latent Space Physical Properties

The motivating reason behind make the latent space as small as it was (eight dimensions), was to find out if either network could learn to encode properties of the physical systems in the latent vectors. To test this, the testing data was run through the autoencoder and each window in the test set was classified according to whether it had a transit or not. Then using a t-distributed stochastic neighbor embedding (t-SNE) the latent space is projected down to two dimensions for visualisation. The results can be seen in Figure 10 on page 12 and 9 on page 11. we can analyse and visually inspect clusters. Comparing the CNN and the RNN together it does appear to be the case that the recurrent autoencoder made a better clustering of curves with transits vs curves without transits. Extremely curious are the clusters of points logically disjoint from the others in the recurrent embedding, we'll come back to those later. Another interesting idea would be to identify what about the latent space defines a transit. To do this we fit a decision tree to classify the output of the encoder into transit and non-transit events. The choice of decision tree was made because decision trees enjoy great transparency in how they classify data and are therefore a useful exploration tool. Fitting a decision tree over the latent space of the CNN autoencoder with a max depth of three using `scikit-learn` and the `DecisionTreeClassifier` gives insight in Figure 11 on page 12. The accuracy of the tree in classifying the windows is 82%, and it seems pretty clear that the fourth component of the encoded latent vector controls something about the transit. Armed with this knowledge we took the instance with the highest $\mathbf{X}_4$ value in the test dataset, whose encoding is $\mathbf{X} = (-19.164, -14.579, -20.560, 32.432, -20.709, -13.890, -18.874, -17.958)$, and plotted the change in the reconstruction of the light curve as the fourth component is decreased overtime in Figure 12 on page 13. Very clearly as the fourth component is varied the structure of the transit changes, first widening from what is seen in $\mathbf{X}_4 = 32.432$ to $\mathbf{X}_4 = 2.432$ and then eventually flattening in $\mathbf{X}_4 = -7.568$ and increasing to a peak when $\mathbf{X}_4 = -37.568$. This seems clear enough to conclude that the autoencoder has learned physical properties of the latent space.

As mentioned earlier, the recurrent network has learned some interesting clustering behaviour, using some simple k-means techniques we can identify where these clusters are and investigate what they represent. In Figure 13 on page 13 you can see the result of using the `scikit-learn` package and the `KMeans` class to cluster our points with eight distinct means. A good representation of the clusters is identified. The clusters $\mathbf{C}_2$, $\mathbf{C}_5$, and $\mathbf{C}_6$ (Fig 14 on page 14) appear to represent transit that are "bottomed out" so to speak, their distinction from the other clusters being a flat bottom as opposed to a sharp transit. The clusters $\mathbf{C}_7$ and $\mathbf{C}_8$ (Fig 15 on page 14) are what could be considered pretty typical transits. The centres $\mathbf{C}_3$ and $\mathbf{C}_4$ (Fig 16 on page 14) have a wave like property, peaking before and after the main dip. It appears the recurrent neural network has learned to discriminate between different types of transits, though this cannot be confirmed without further inquiry.

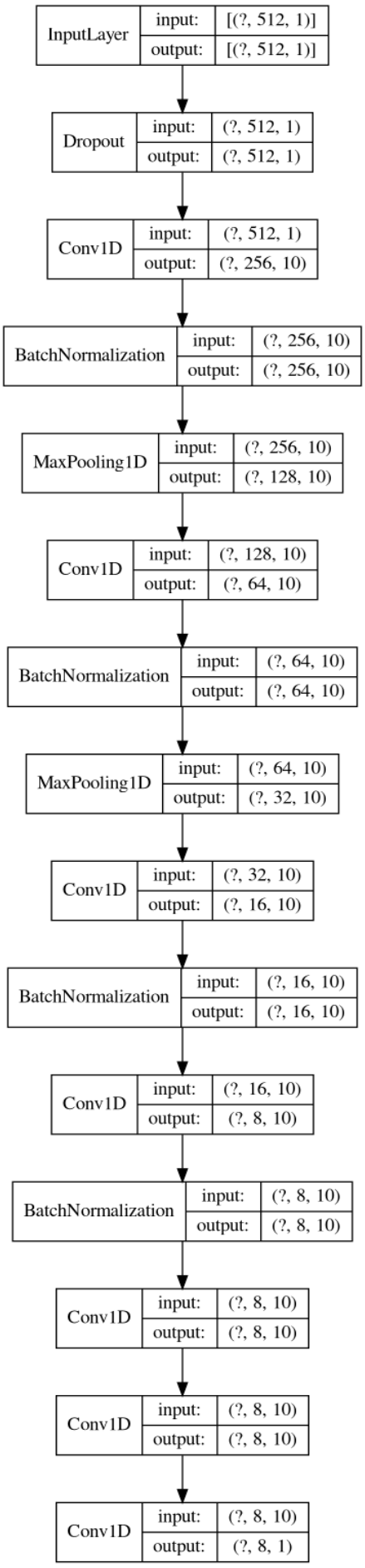
3 CONCLUSION AND FURTHER RESEARCH DIRECTIONS

The hypothesis was that by changing the loss function to ask the autoencoder to fit a fitted model output rather than simply reconstruct the curve, we might be able to find evidence of physical properties in the latent space. Through analysing the latent space of two models in trained in this fashion it has been shown that this is generally the case. The convolutional model has been shown to definitely contain

information about if a transit exists or not, and the recurrent model shows evidence of modeling different types of transit events.

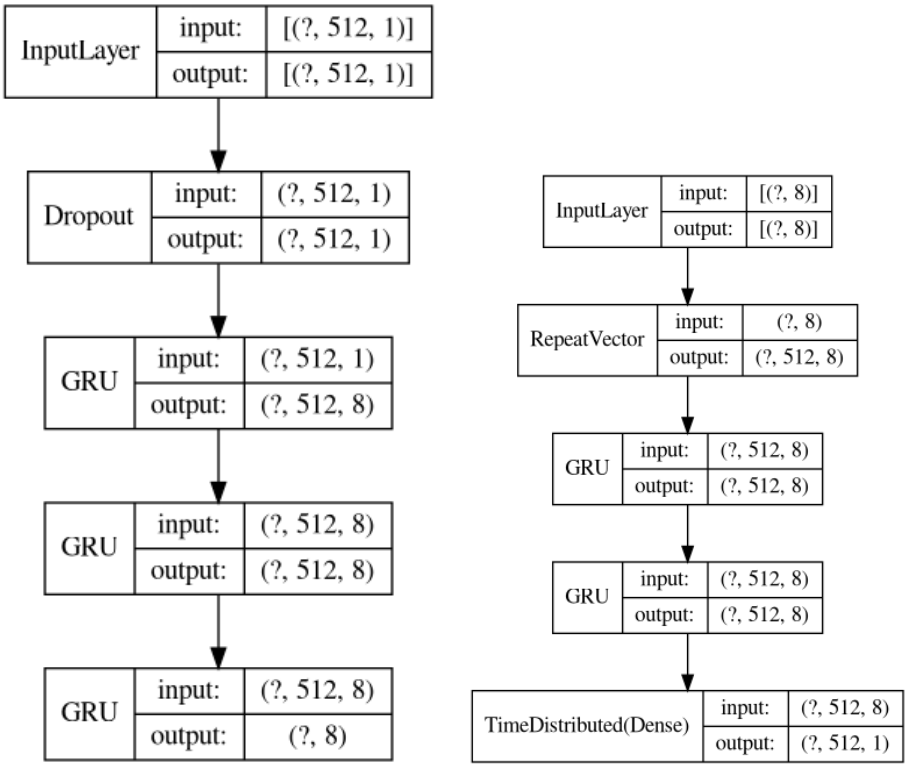
Whilst the part three models did show evidence of physically modelling the stellar systems, one probably wouldn't use them in a scientific context due to accuracy concerns. The part two autoencoder as discussed produced less noisy results, and this opens the possibility for perhaps exploring other values of α that might produce a model that can capture relevant physical features and produce a noiseless fit to the model. Such a model could be useful for astronomers attempting to identify candidate systems for transiting exoplanets.

This report is only a sample of the analysis and models built to analyse these datasets, with others excluded in the interests of report brevity. Given that both part three models appear to approximate physical systems we can speculate about what might be possible with a little extra effort. It seems clear that a small network on top of either network encoders could learn to replace the original transit least squares model by producing transit fits that are overall better than the original TLS method. Given a little more depth and a lot of training data one might conjecture the RNN model should be able match the performance of the CNN, it would be interesting to see how the RNN would represent things like transits in that case. The analysis of the latent space only really explains one variable, it would be interesting to see what might happen if one took a similar analysis to the remaining variables and figured out exactly how the CNN represents curves (a good conjecture might be that the CNN makes its representation spatially, hence why the transits in Figure 12 on page 13 centres the transit with $\mathbf{X}_4$).



(a) Encoder. **(b)** Decoder.

Figure 1: CNN Autoencoder model.



(a) Encoder. **(b)** Decoder.

Figure 2: RNN Autoencoder model.

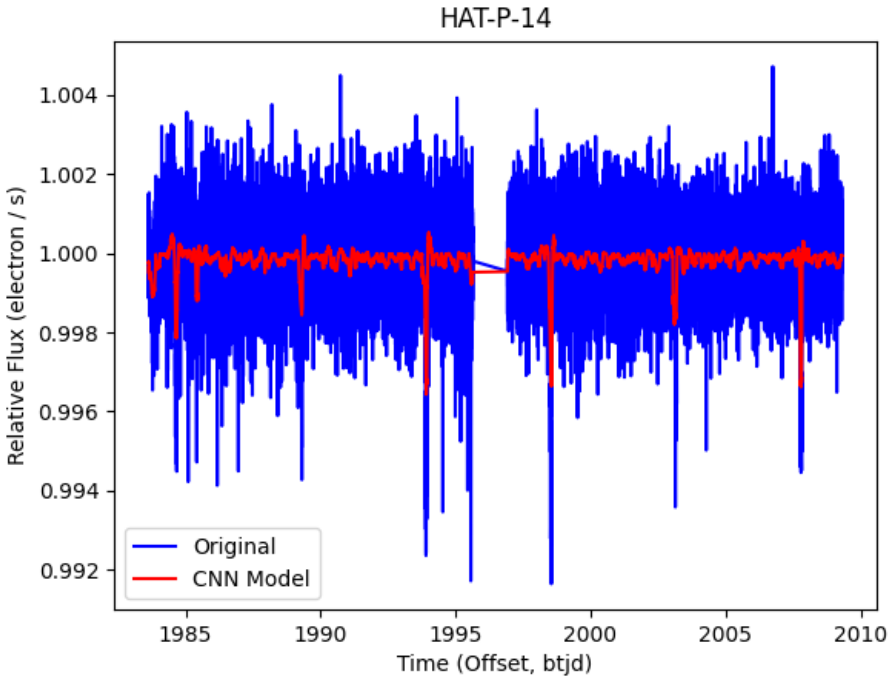


Figure 3: A comparison between HAT-P-14 on the original and RNN model output.

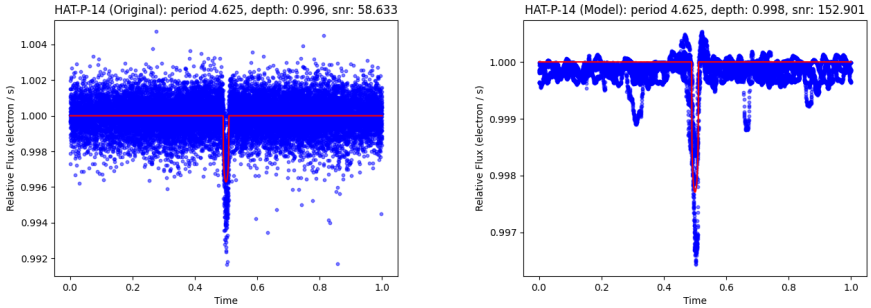


Figure 4: HAT-P-14 transit fold compared between original (left) and part two model (right).

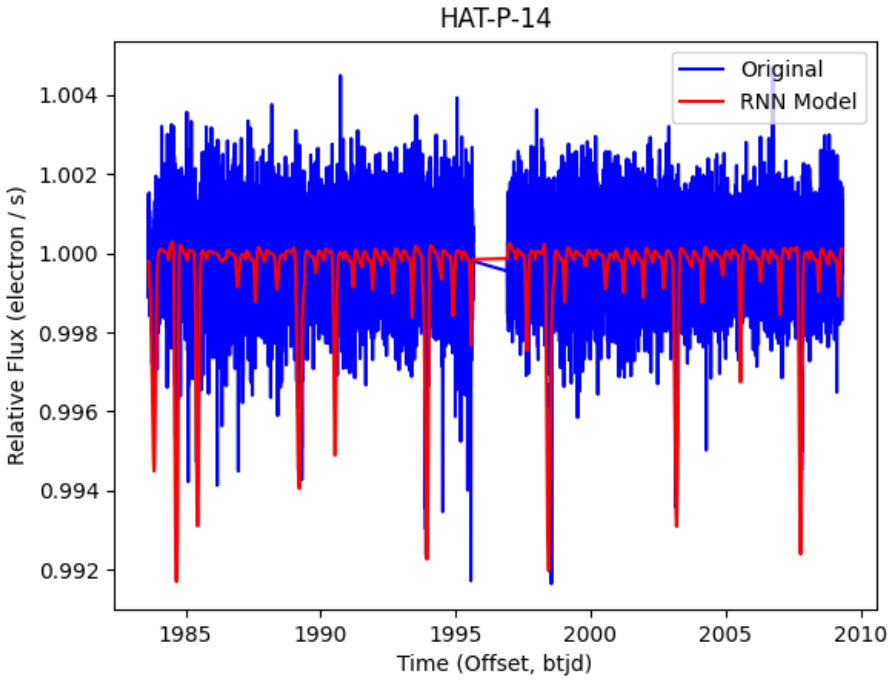


Figure 5: A comparison between HAT-P-14 on the original and RNN model output.

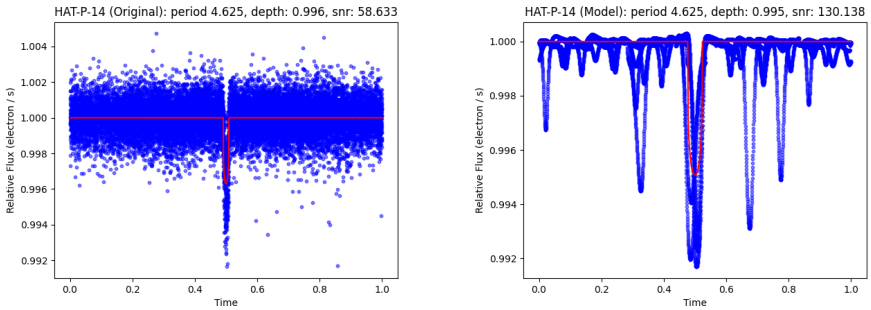


Figure 6: HAT-P-14 transit fold compared between original (left) and RNN model (right).

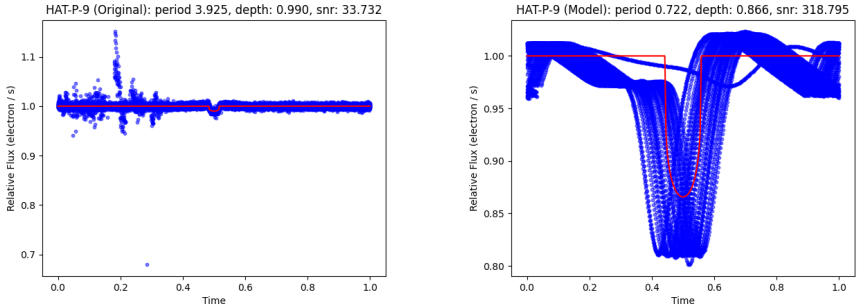


Figure 7: HAT-P-9 transit fold compared between original (left) and RNN model (right).

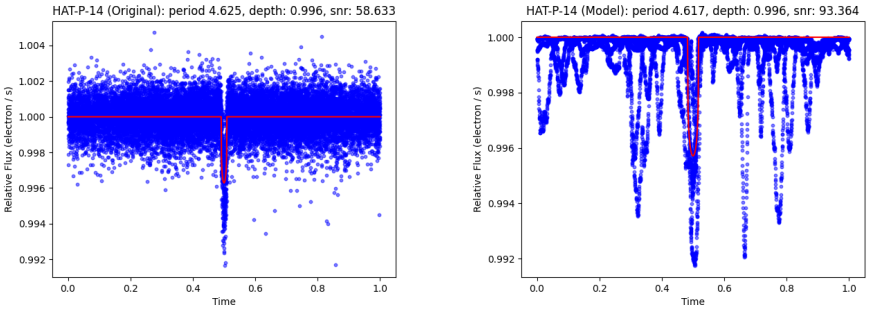


Figure 8: HAT-P-14 transit fold compared between original (left) and CNN model (right).

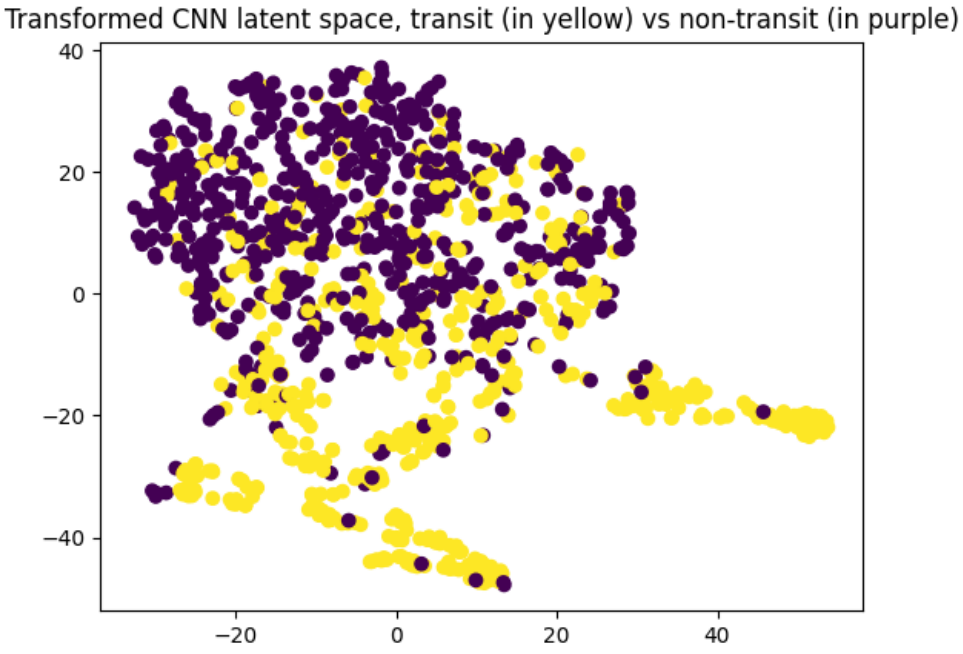


Figure 9: A t-SNE projection of the convolutional latent vector space.

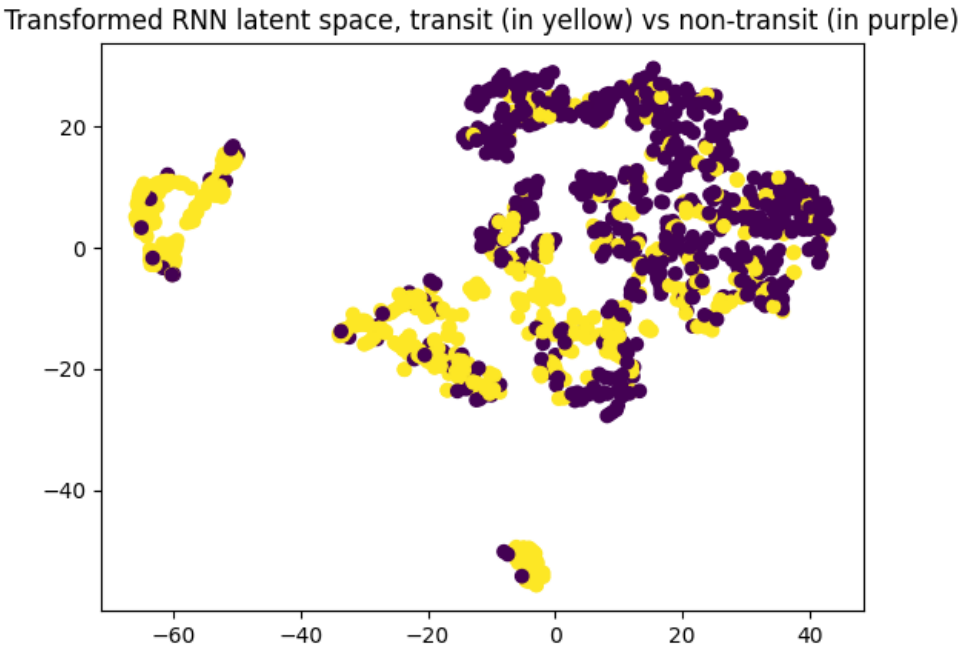


Figure 10: A t-SNE projection of the recurrent latent vector space.

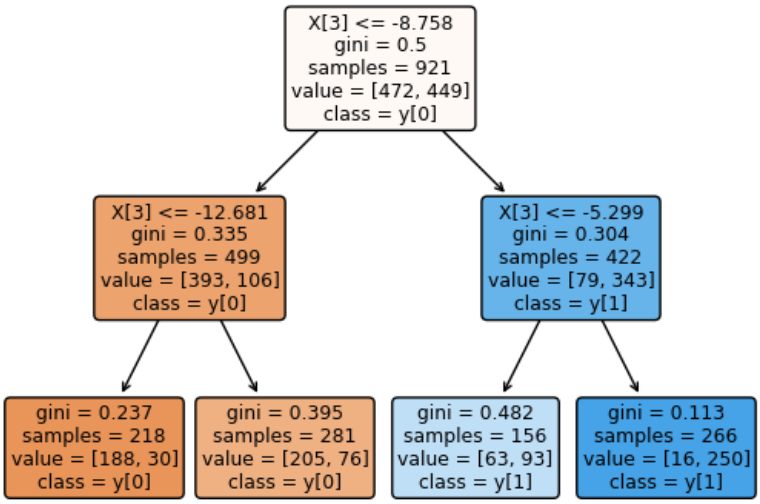


Figure 11: Decision Tree fitted to classify transits in the CNN latent vector space.

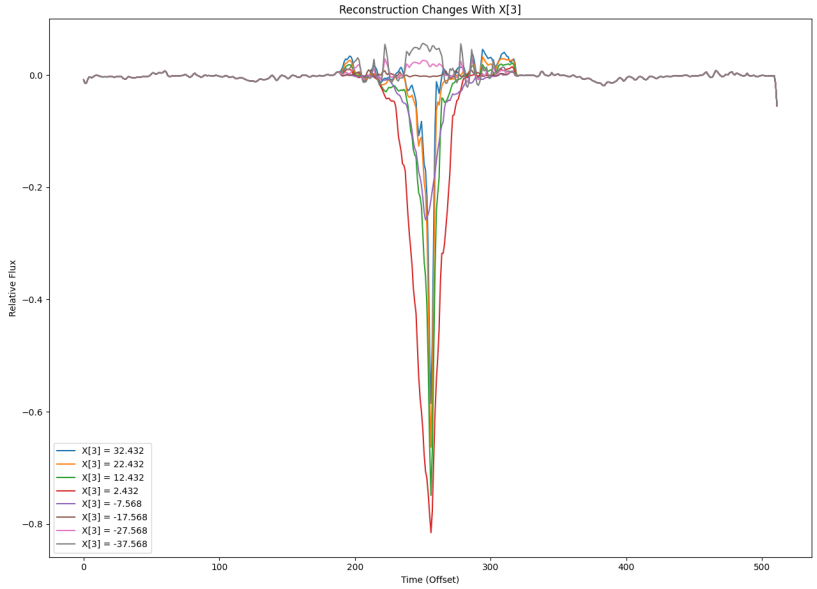


Figure 12: A comparison of the reconstruction curves for the latent vector space of the CNN.

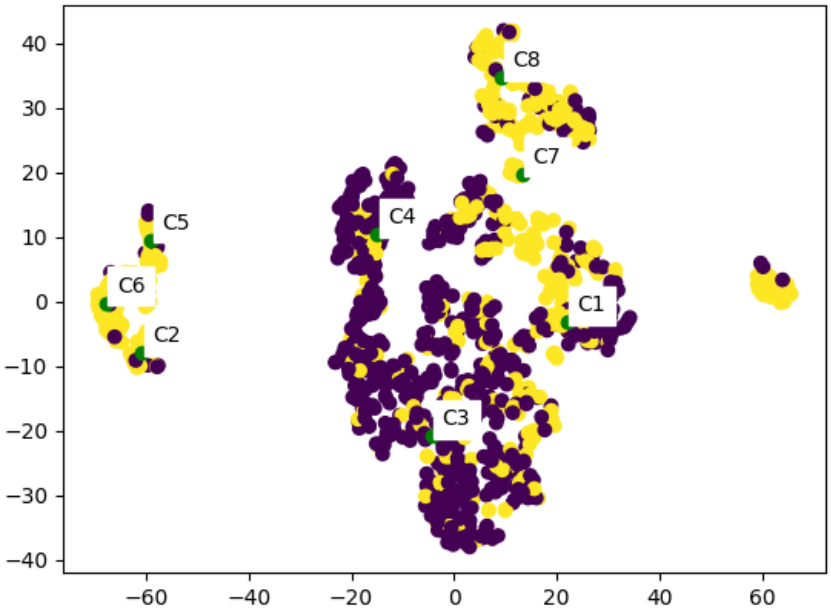


Figure 13: Projection of RNN latent space including clusters.

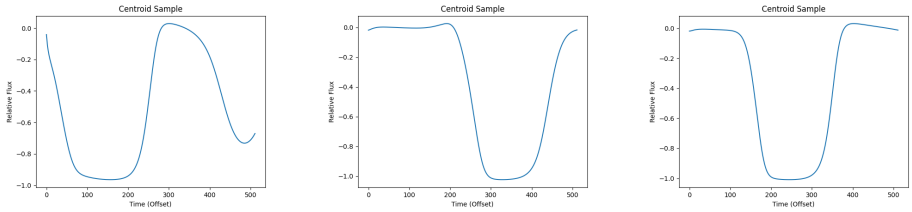


Figure 14: Reconstructed curves for $\mathbf{C}_2$, $\mathbf{C}_5$, and $\mathbf{C}_6$ (left, mid, right respectively).

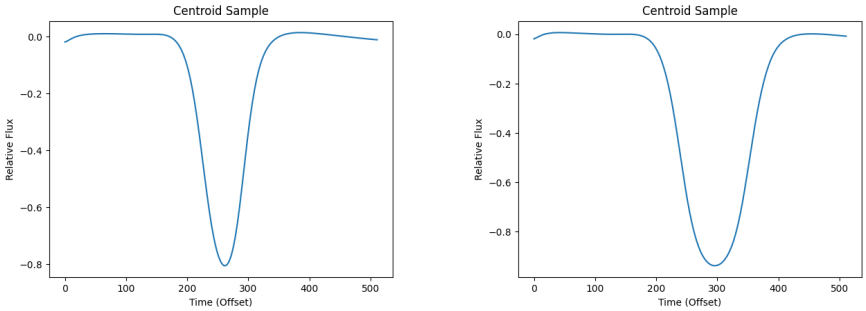


Figure 15: Reconstructed curves for $\mathbf{C}_7$ and $\mathbf{C}_8$ (left, and right respectively).

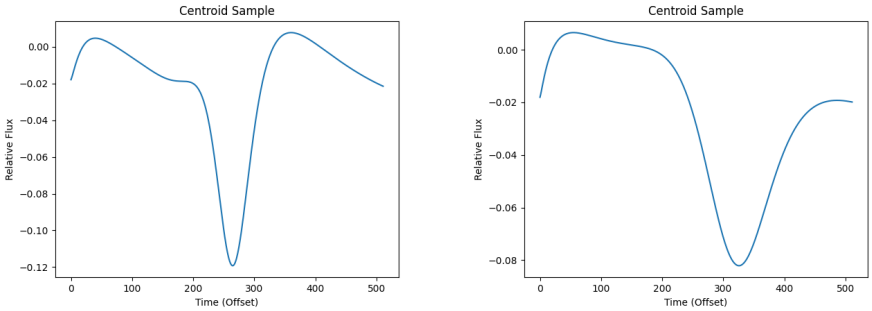


Figure 16: Reconstructed curves for $\mathbf{C}_3$ and $\mathbf{C}_4$ (left, and right respectively).