Exercise 2: Uncertainty Prediction

Erik Stolt

What you did and how

I plotted the spectra for three stars and after that implemented a CNN (the problem stated NN but since we did CNN in the previous exercise I assumed it is fine here too. CNN is in the end a type of neural network) structure to predict the three labels effective temperature (T_{eff}), surface gravity ($\log g$) and metallicity (Fe/H) and their corresponding uncertainties. I trained my network only on the flux-wavelength spectra to obtain these results. Furthermore, I implemented the more efficient way of handling the large dataset, i.e. by using a class called `SpectraDataset`.

What results you obtained

The final network was trained on 30 epochs (using early stopping, it stopped at 26) and achieved a best loss of ≈ -2.6 on the test data set. I also printed out some values to see that they looked reasonable, one print out looked like this

```
Example 1:
Predicted mu: [ 4.8344077e+03  2.6435940e+00 -2.5412348e-01] Predicted std: [0.05350459 0.09784
Actual:      [ 4.7878350e+03  2.4332738e+00 -3.2542446e-01]
--------------------------------------------------
Example 2:
Predicted mu: [5.9847109e+03 4.4565301e+00 1.9503428e-01] Predicted std: [0.02229325 0.02859625
Actual:      [5.856437e+03 4.413752e+00 1.594904e-01]
--------------------------------------------------
Example 3:
Predicted mu: [ 4.7090605e+03  4.5497613e+00 -2.2964796e-01] Predicted std: [0.02402881 0.04223
Actual:      [ 4.7619224e+03  4.6158781e+00 -2.2512627e-01]
```

in the order $T_{eff}, \log_g$ and Fe/Hh and their corresponding uncertainties.

What challenges you encountered and what could be improved

The major challenge was getting good values for the uncertanties. At first I forgot to scale the labels data which resulted in poor results in the sense that the pull distribution did not follow a gaussian. However, when I rescaled the data, the results became instantly better.

Plots

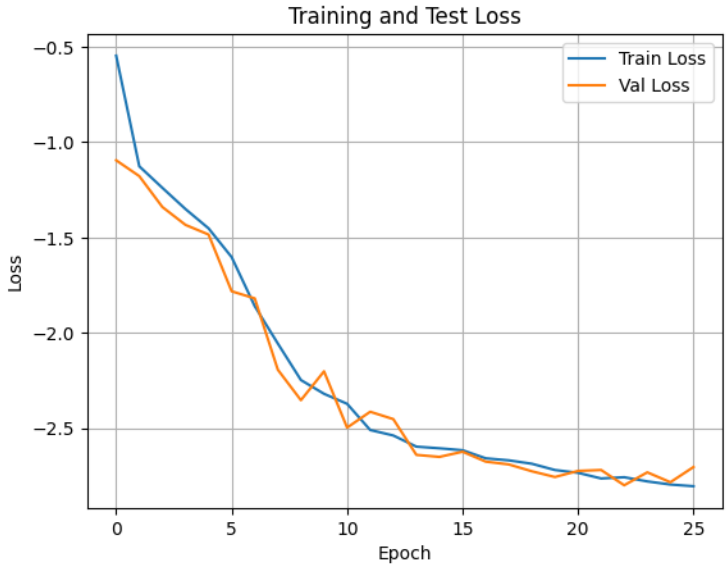

Figure 1: Loss for the training data and the test data over 30 epochs.

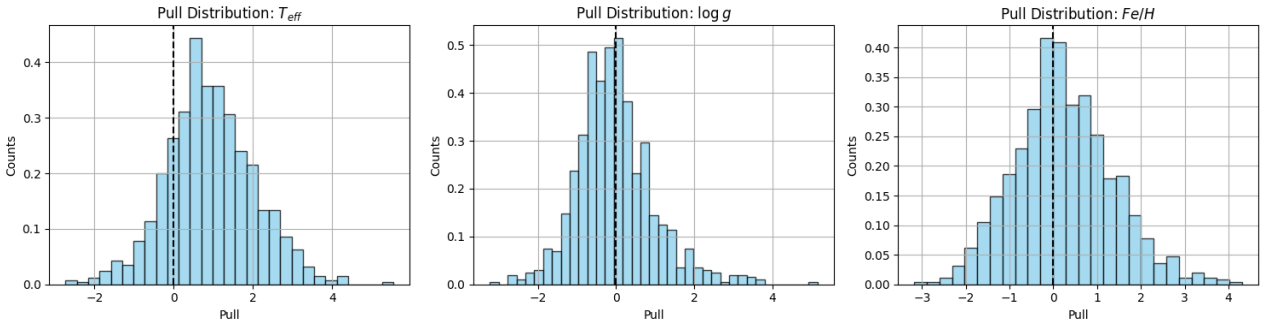

Figure 2: Pull distributions for the three uncertainties corresponding to T_{eff}, surface gravity ($\log g$) and metallicity (Fe/H).

We see that the standard deviations all follow a gaussian with a mean zero (or close to). Since they are not particularly wide or thin, we conclude that the the prediction are resonable. We can summarize this as: If the pull distribution:

* is wider than standard normal (e.g., std > 1) –> the uncertainties are too small (underconfident)

* is narrower (e.g., std < 1) –> the uncertainties are too large (overconfident)

* is shifted from 0 –> the model is biased. This can be argued to be the case for T_{eff}, however, this is a relatively small effect.

Link to Colab

Link: `https://colab.research.google.com/drive/14ndkHlyFq3ObjzLSyYcE9HMkoNdqAWPA?usp=sharing`

Final thoughts

Final thoughts

Training We see that the training, early stopping is triggered after 26 epochs. At this point we have obtained good result which we see at the prints. The prints looks relatively close to the true value, indicating the model works appropriately.

Pull Distrubutions We see that the standard deviations all follow a gaussian with a mean zero (or close to). Since they are not particularly wide or thin, we conclude that the the prediction are resonable. We can summarize this as: If the pull distribution:

* is wider than standard normal (e.g., std > 1) âĘŠ the uncertainties are too small (underconfident)

* is narrower (e.g., std < 1) âĘŠ the uncertainties are too large (overconfident)

* is shifted from 0 âĘŠ the model is biased. This can be argued to be the case for T_{eff}, however, this is a relatively small effect and the std looks really good! So overall the results are good I would argue.

Scaling Notice that we have trained and tested the model on the scaled data. This process can always be inverted via

$$y_{\text{original}} = y_{\text{scaled}} \cdot (y_{\text{max}} - y_{\text{min}}) + y_{\text{min}}, \tag{1}$$

for minmaxscaler for *labels* and for the *spectra* via spectra inverted = np.exp(spectra transformed). This I have easily done by labels unscaled = scaler.inverse transform(labels scaled) as implemented.

Negative Loss We notice that we have negative loss during training and testing. This we are not used to when using *mse*. However, this is not a problem since we have used a modified loss-function Negative log-likelihood (*NLL*). We can always add a constant to this to keep it positive, however, the negativity is not an inherent problem. This we can back up by the following observations:

* The predictions make sense (plot a few Îij Âś ÏČ intervals)

* The validation loss continues to track training loss

* There's no overfitting or instability

Then this is fine. The negative loss just means the model is improving in *log-probability space*.