Assignment #1 – Astronomy CNN with PyTorch

Github repository

Written summary

Also available in link

approach

1. hyperparameters:

   batch_size = 32

   learning_rate = 0.001

   num_epochs = 10

2. the device used to alocate pytorch tensors was MPS.
3. data was stored locally (due to its size) in a folder called data/ at a .npy extension. the data can be found at https://huggingface.co/datasets/simbaswe/galah4/tree/main.
4. a CustomDataset class (CustomDataset()) is created to handle the dataset, it inherits the Dataset class from torch.utils.data.
    1. features and labels normalization was handle during the class instantiation. features (wavelength) were logarithm normalized and labels (surface temperature, surface gravity and metallicity) were normalized using standard score.
    2. as the class take data path and load files as numpy arrays, this class also convert the arrays to tensors only when an element is called using the *_getitem_* method. the tensors is sent to device during each batch at train, validation and test phases.
5. with the help of torch.utils.data Dataloader() train (70%), validation (15%) and test (15%) datasets are handled.
6. the model (model.py) consists of 3 convolution-maxPooling layers with ReLu activation function, one fully connected hidden layer with ReLu activation and the output layers with 3 neurons. data is flattened before passing to the fully connected layer.
    1. the loss function used (criterion) is the Mean Squared Error (MSE).
    2. the chosen optimization method is Adam.
7. training, validation and test is made and the values of interest recorded and a python variables and also logged using tensorboard. (tensorboard --logdir=runs)

results

1. the training-validation loop for the hyperparameters took around 2.5 minutes to complete.
2. training loss has a monotonic decrease throughout the epochs.
3. validation losses decreases throughout epochs.

challenges

the experiments below are made changing only the informed hyperparameter keeping the others constant.
when talking about performance, it is refered to: train and validation loss.

1. Learning rate: using a higher value (0.01) leads to overshoot and worst model performance.
2. Batch size: lower batch sizes leads to lower train and validation losses at the initial epochs but for values as 16 and 64 it converges to similar performance at the 10th epoch.
3. Epochs: from the 10th to the 15th epoch, there is no significant increase in the model performance and the validation losses presents a movement to increase (overfitting).
4. Model loading: for further assignments, a function to load a model (if any) will be implement to save redundant computations, expecially when acquiring images and plots outside tensorbord.

Result plots

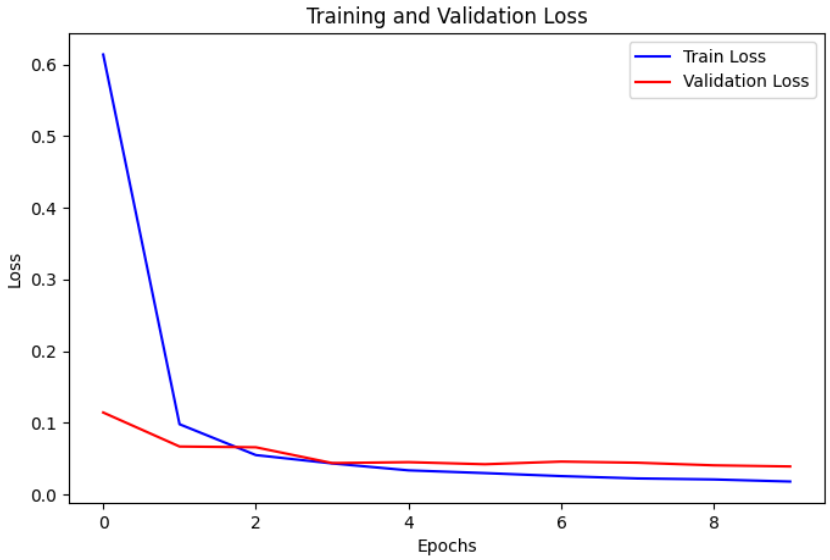

Figure 1: Train and validation loss per epoch.

From Figure 1 we see a decrease as the model goes through the epochs. This experiment used $Batch\ size = 32, Learning\ Rate = 0.001, Epochs = 10$. When using 15 epochs (not shown in the chart) the model starts to present overfitting as the train loss slightly decreases and the validation loss started to increase.

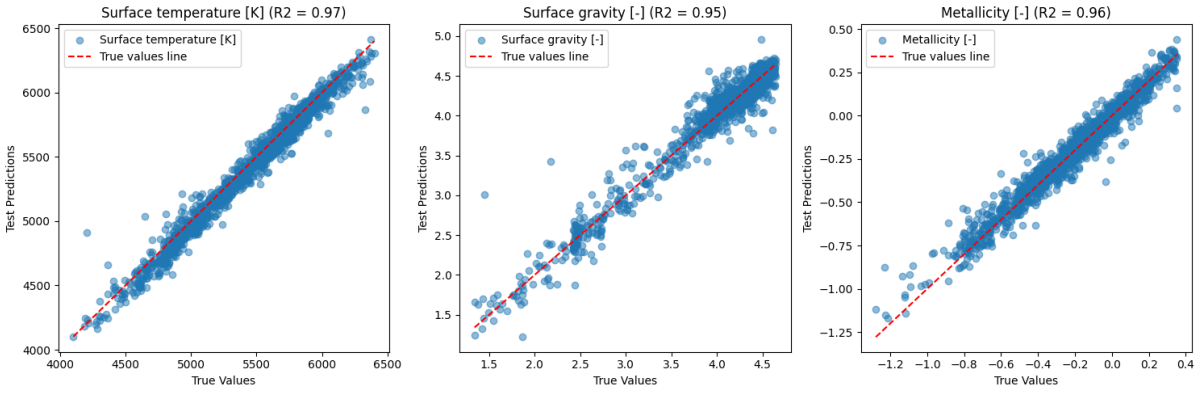

Figure 2: Predicted and true values for each label

From Figure 2, we see the model presented a good performance, using the R^2 metric, in predicting the three labels.