Assignment #3 – Normalizing Flow

https://github.com/Lemorita95/1FA006/tree/main/3_normalizing-flows

1 Written summary

Also available in https://github.com/Lemorita95/1FA006/blob/main/3_normalizing-flows/README.md

approach

1. hyperparameters:

   batch_size = 64

   learning_rate = 0.8e-4

   num_epochs = 10

2. Load data and stored in a CustomDataset class;
3. Data is normalized within CustomDataset class _*init*_ through z-score normalization;
4. Train, validation and test dataset is created with PyTorch Dataloader;
5. Training loss is defined as negative mean of log probability nf_loss();
6. Gives the user the possibility to load a model (if found at models/model.pth);
7. Default model is defined with 6 convolution layers (ReLU activation), 2 hidden layers (ReLU activation) and 1 output layers;
8. Output layer with variable size due to Normalizing flow;
9. Train and validate model train_validade_model();
10. Test model test_model();
11. Model outputs the mean and standard deviation of samples CombinedModel.forward();
12. Denormalize mean and standard deviation denormalize(), denormalize_std(), for std the mean for normalizing is not added;
13. Plot pull histogram, scatter of averages and residuals;

results

1. the training-validation loop for the hyperparameters took around 5 minutes to complete the diagonal gaussian, 5 minutes for full gaussian and 1:50h for full flow;

2. in terms of train and validation loss, the model only outperforms the TinyCNN for the full flow;
3. model does not improve validation loss (overfits) for full gaussian after 10 epochs.
4. model behavior is unstable for different flows. So the use of the same model and hyperparameter for different flows are limited.

challenges

1. tried two different models, CNNModel (from previous tasks) and TinyCNN - CNN runs faster but overfits easilier for some flows.
2. for this task, the models seem to be more sensitive to learning rate, for CNNModel i had to lower it for convergence for diagonal gaussian and full gaussian, for full flow i was able to increase the learning rate and batch size.
3. two different models for different flows presented a correlation between true values and residuals. The effect was not completedly understood whether its caused by the data processing, hyperparameter tuning or model architecture.
4. diagonal gaussian flow: CNNModel overfits after 10 epochs. changed the architecture of model (maxpool -> avgpool), included global_pool before fully connected layer and the model performed better, did not overfitted but the validation loss did not got much better after 10th epoch. TinyCNN runs slowly but performs good and consistently. given the difference at validation loss (CNNModel: 3.53 x TinyCNN: -0.04) and time to train 10 epochs (CNNModel: 11 min x TinyCNN: 19 min), TinyCNN is preferable.
5. full gaussian flow: after overfit adjustments at CNNModel, the same behavior as in diagonal gaussian flow was observed for both models with TinyCNN outperforming (in terms of validation/test loss) CNNModel.
6. full flow: computational expensive, even on google colab. CNNModel took 4 hours (20 epochs), the training and validation loss was continuously decreasing and the slope was constant, not decreasing, seems that if more epochs was performed, the model would be better. the mean predictions however, was not very meaningfull (predicting constant values). When increase the batch size (64) and learning rate (0.8e-4) for CNN model, the performance got better (from CNNModel with batch_size=32 and lr=0.8e-5 and from TinyCNN) but the computation time did not changed much (for 20 epochs, 1st CNNModel: 3.8h, TinyCNN: 3.78h, CNNModel2: 3.5h).
7. The predicted means for full flow, however, was kinda far from the perfect predictions (x=y). And the residuals for log_g presented a high correlation to the true values.
8. the final model of choice was CNNModel with batch_size=64 and lr=0.8e-4.

9. for full flow it was harder to tune hyperparameters due to the amount of time it took for training, so not much further investigation was made.
10. The same CNNModel for full flow does not work optimally for diagonal and full gaussian, starts to overfit after ~17 epochs and predicted means correlation to true values are low.
11. Tried softplus activation for hidden layers, but the model did not converged.
12. Altough percentile normalization could increase the performance of the model when handling extreme values, this test was not made for the model of this assignment.

2 Result plots

For this section, three different networks are discussed. The hyperparameters such as batch size, learning rate and epochs was preserved throughout different flows to access the fitness of the model for this different flow. The content present for each network are similar in order to possibilitate a comparison between them.

2.1 Diagonal Gaussian

The normalizing flow for diagonal gaussian have 6 parameters, 3 for mean and 3 for standard deviation, considering we have three labels.

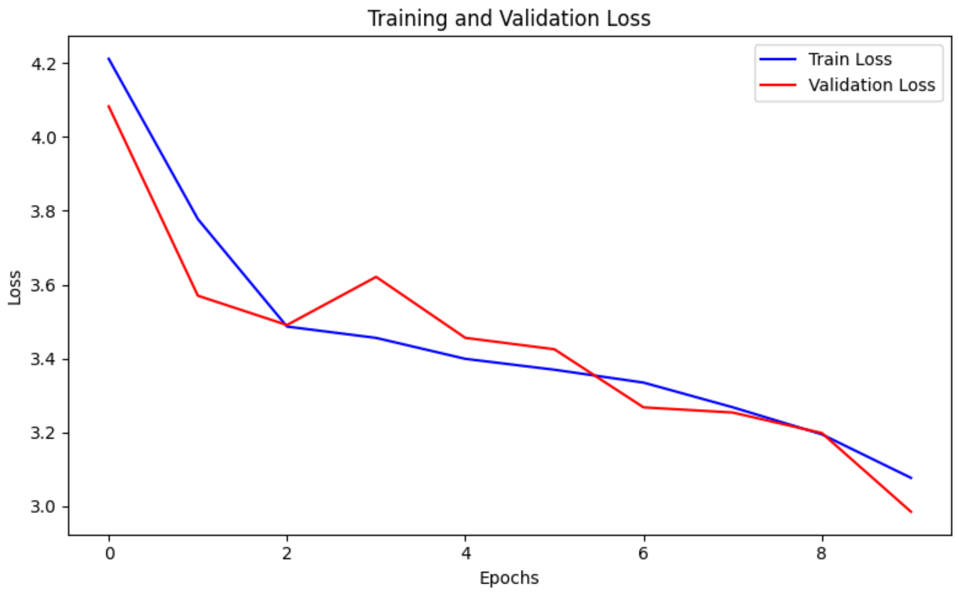

Figure 1: Train and validation loss per epoch for diagonal gaussian flow.

From Figure 1 we see a decrease in loss as the model goes through the epochs. This experiment used $Batch\ size = 64, Learning\ Rate = 0.8 \times 10^{-4}, Epochs = 10$. When using 20 epochs (not shown in the chart) the model starts to present overfitting at the 16th epoch as the train loss keeps decreasing and the validation loss started to increase.

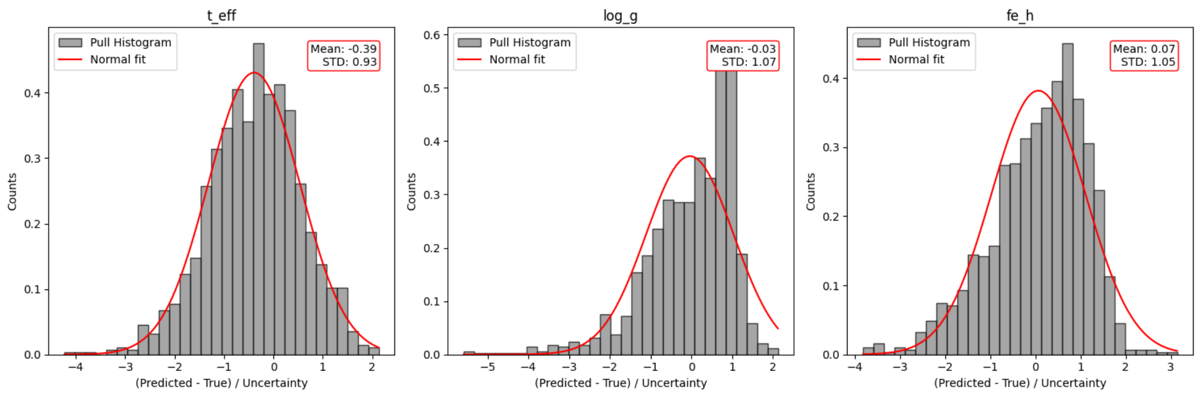

Figure 2: Pull histogram of predictions, distribution of residuals (predicted value – true value) normalized by the predicted standard deviation for diagonal gaussian flow.

From Figure 2, we see the model performance varies for each label. Since the desired reference value is a normal distribution with zero mean and unit standard deviation, this CNN performs better for surface gravity and metallicity (although negative skewness is observed in both) then as for surface temperature, where bias can be observed. The model performs worst for surface temperature as in addition to bias, a lower-than-expected uncertainty is seen.

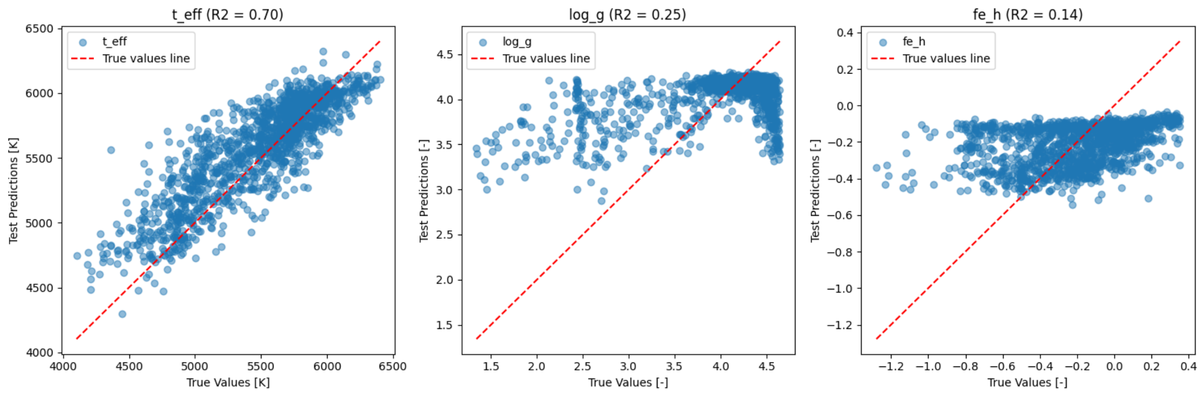

Figure 3: Scatter plot of predicted means for diagonal gaussian flow.

Figure 3 shows that the model prediction for means in more meaningful for surface temperature despite lower performance from Figure 2. For surface gravity and metallicity, to minimize the loss function, the model seems to place more importance in the standard deviation as then the mean, which can be seen as similar prediction for different true values (a horizontal line). Large values of surface gravity also seem to be an issue for this model.

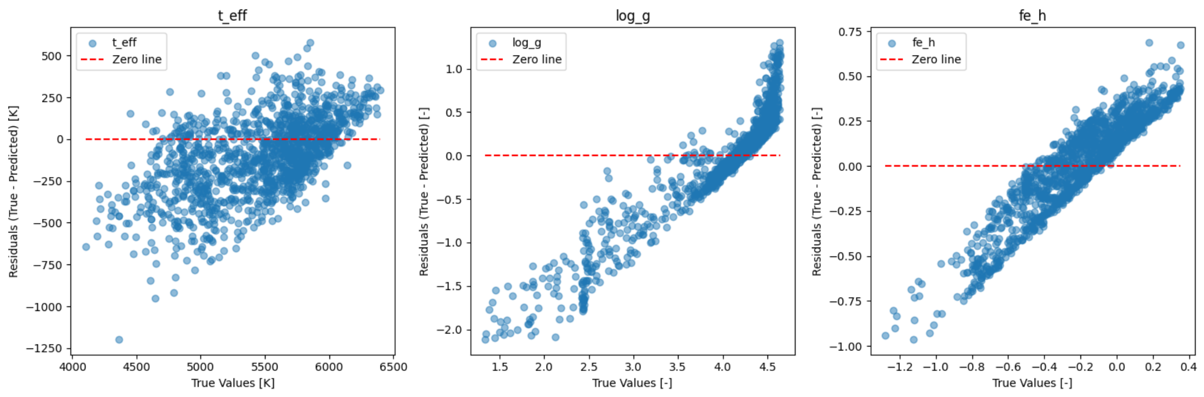

Figure 4: Scatter plot of predicted means residuals for diagonal gaussian flow.

Figure 4, as complement of Figure 3, clearly show a high correlation between the true values and the residuals, especially for surface gravity and metallicity, also showing the lower importance the model attributed to the predicted mean.

2.2 Diagonal Gaussian

The normalizing flow for diagonal gaussian have 9 parameters, 3 for mean and 6 for triangular covariance matrix, considering we have three labels.

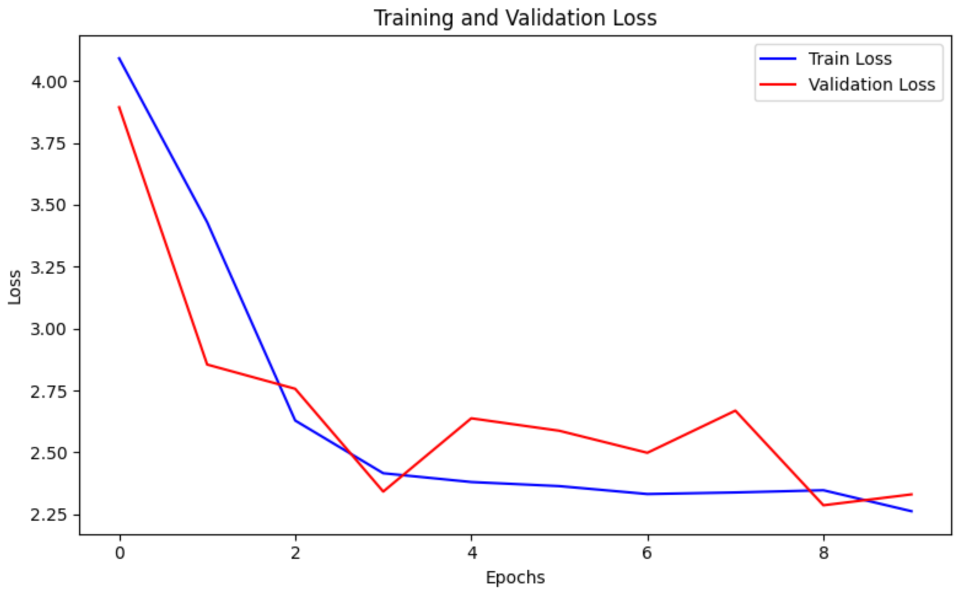

Figure 5: Train and validation loss per epoch for full gaussian flow.

From Figure 5 we see a decrease in loss as the model goes through the epochs. This experiment used $Batch\ size = 64, Learning\ Rate = 0.8 \times 10^{-4}, Epochs = 10$. The model does not present a good performance for this normalizing flow as the validation loss does not improve (overfitting) after the 3rd epoch.

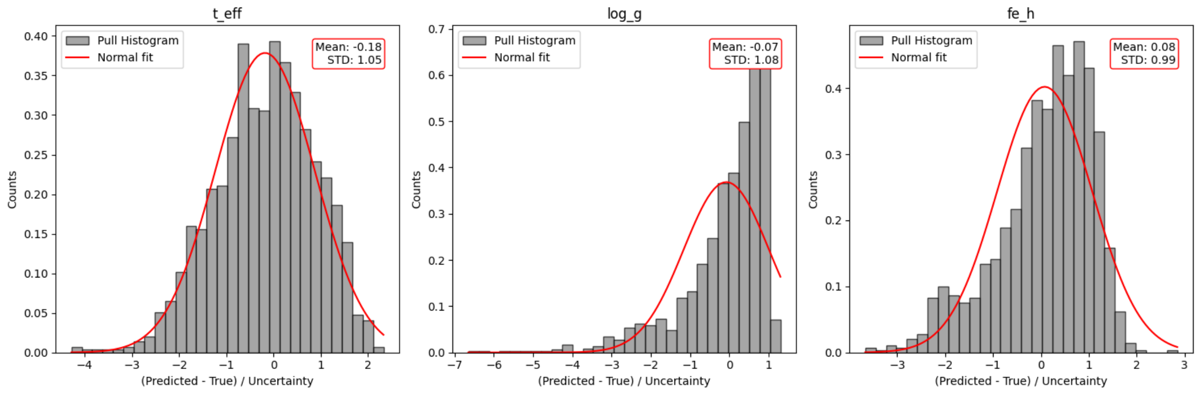

Figure 6: Pull histogram of predictions, distribution of residuals (predicted value – true value) normalized by the predicted standard deviation for full gaussian flow.

From Figure 6, we see the model performance varies for each label. Since the desired reference value is a normal distribution with zero mean and unit standard deviation, this CNN performs better than the previous one with diagonal gaussian flow. Negative skewness is observed one more for surface gravity and metallicity. This model, for surface gravity, does not predict mean values greater than a certain value, as observed in the sharp end of the histogram in the figure above.

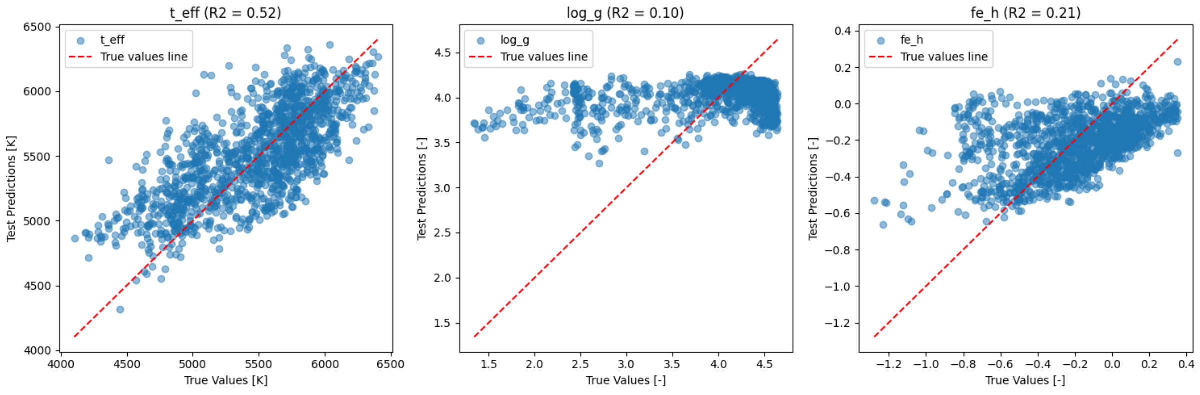

Figure 7: Scatter plot of predicted means for full gaussian flow.

Figure 7 shows that the model prediction for means in more meaningful for surface temperature despite lower performance from Figure 6. For surface gravity, to minimize the loss function, the model seems to place more importance in the standard deviation as then the mean, which can be seen as similar prediction for different true values (a horizontal line).

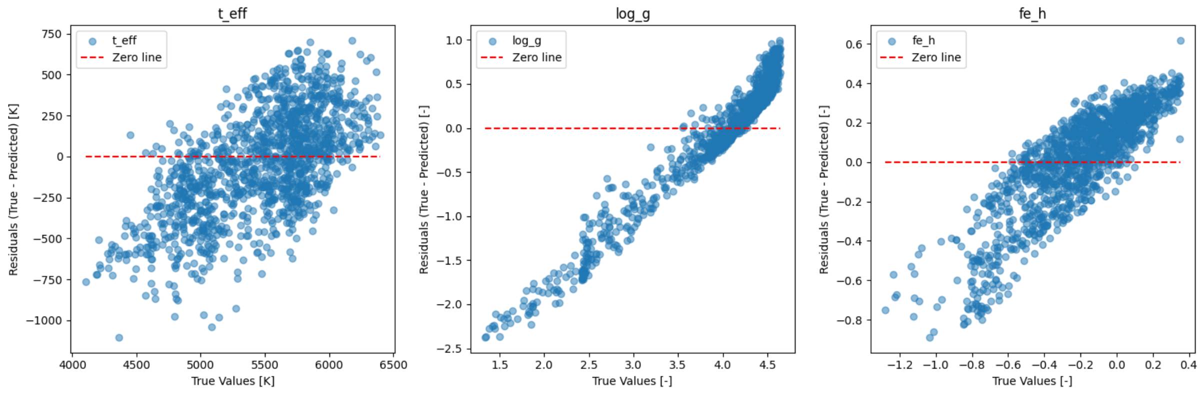

Figure 8: Scatter plot of predicted means residuals for full gaussian flow.

Figure 8, as complement of Figure 7, clearly show a high correlation between the true values and the residuals, especially for surface gravity and metallicity, also showing the lower importance the model attributed to the predicted mean.

2.3 Full Flow

The normalizing flow for full flow have 306 parameters.

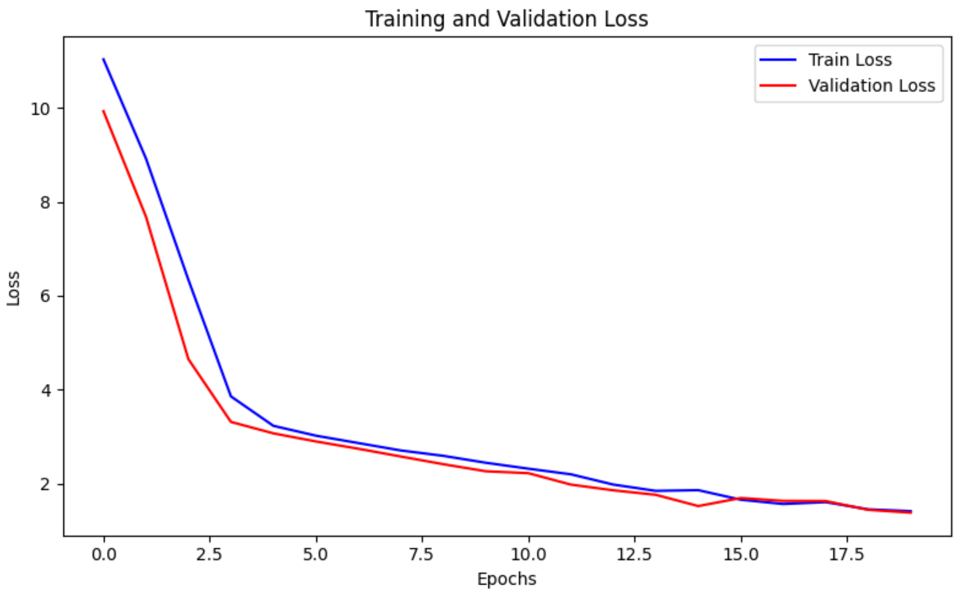

Figure 9: Train and validation loss per epoch for full flow.

From Figure 9 we see a decrease in loss as the model goes through the epochs. This experiment used $Batch\ size = 64, Learning\ Rate = 0.8 \times 10^{-4}, Epochs = 20$. For this flow, a larger number of epochs was used to test if the model would overfit, as seen on previous flows, and no sign of it was observed. The loss was continuously decreasing, and a larger epoch number would further decrease the loss. This test was not made high amount of time this model took to compile.

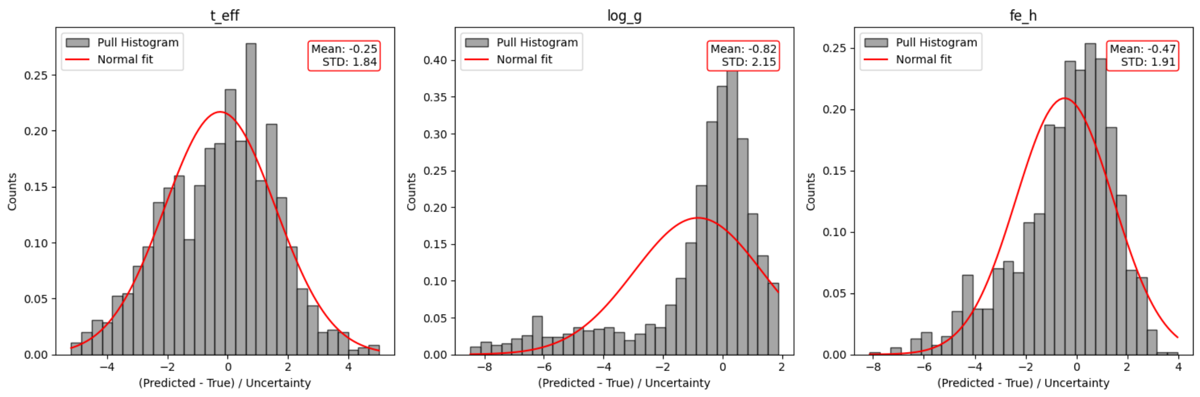

Figure 10: Pull histogram of predictions, distribution of residuals (predicted value – true value) normalized by the predicted standard deviation for full flow.

From Figure 10, we see the model performance varies for each label. Since the desired reference value is a normal distribution with zero mean and unit standard deviation, this does not perform well as its present significant bias and high uncertainty of the predictions.

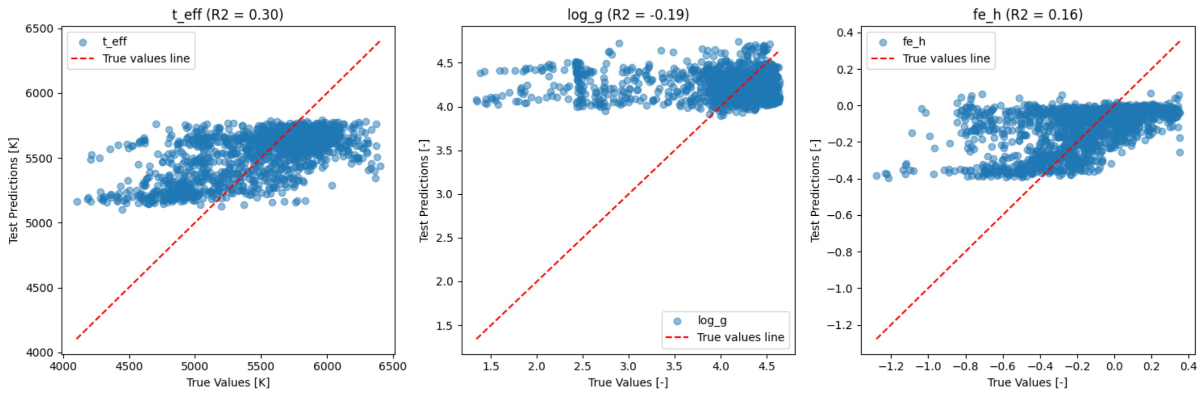

Figure 11: Scatter plot of predicted means for full flow.

Figure 11 shows that the model does not predict meaningful values for any of the three labels as to minimize the loss function, the model prediction of means is notably constant.

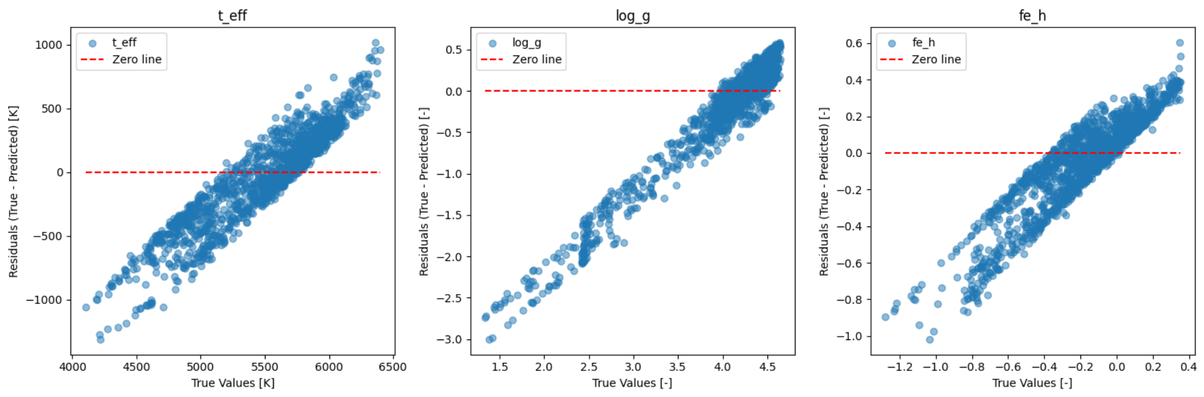

Figure 12: Scatter plot of predicted means residuals for full flow.

Figure 12, as complement of Figure 11, clearly show a high correlation between the true values and the residuals, showing the lower importance the model attributed to the predicted mean.

Since the result for CNNModel for full flow was not good, TinyCNN model was also used for full flow to access its performance. This model has low uncertainty on the predictions, as seen on the Figure 14 but the effect of high correlation of residuals are also observed. For this model, it was used $Batch\ size = 32, Learning\ Rate = 0.8 \times 10^{-5}, Epochs = 20$.

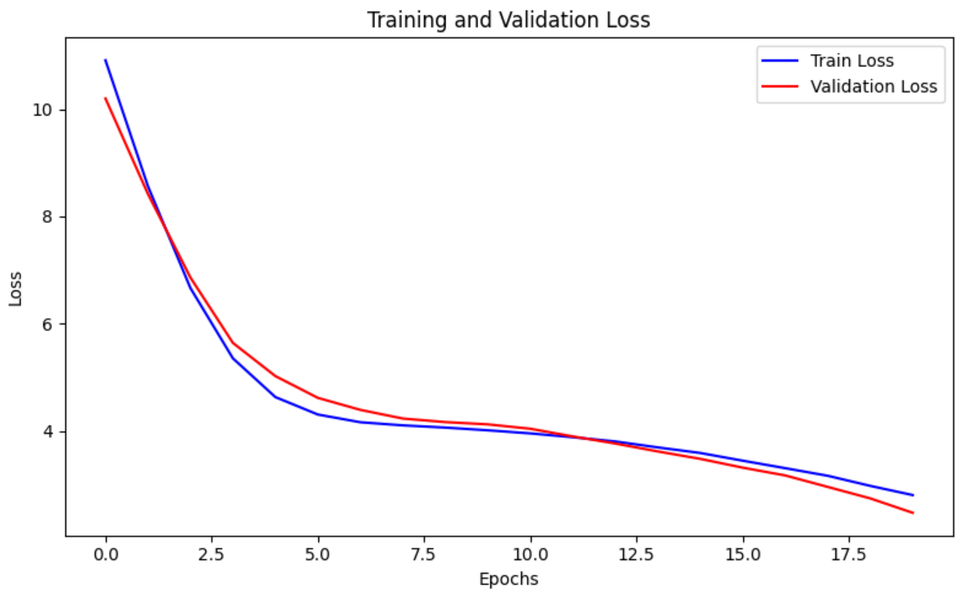

Figure 13: Train and validation loss per epoch for full flow. TinyCNN model.

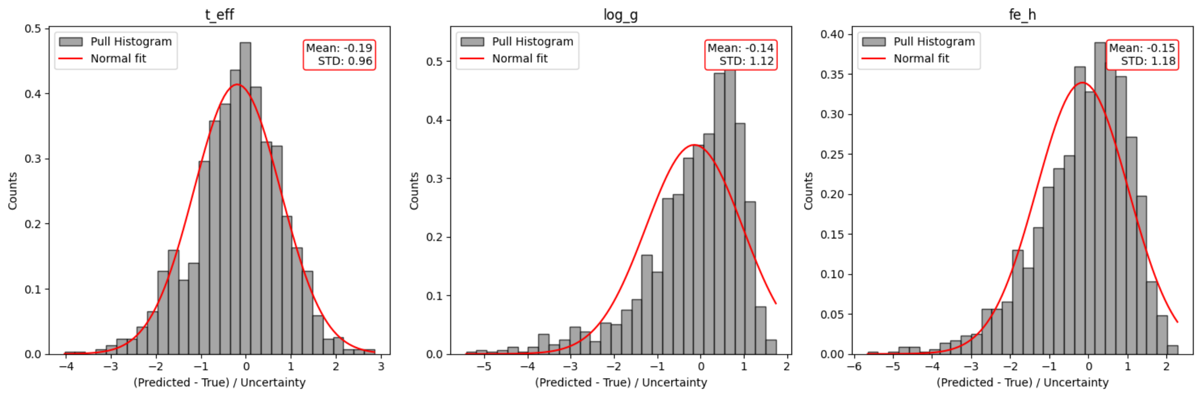

Figure 14: Pull histogram of predictions, distribution of residuals (predicted value – true value) normalized by the predicted standard deviation for full flow. TinyCNN model.

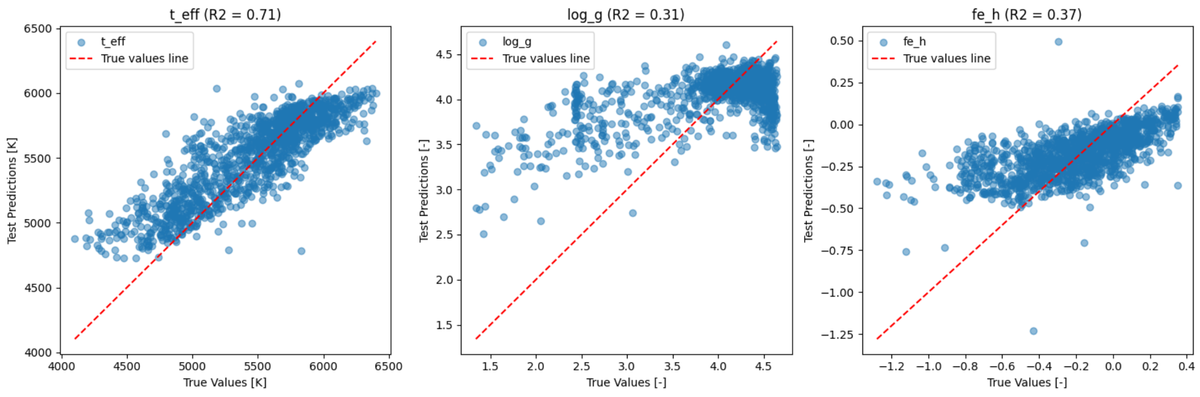

Figure 15: Scatter plot of predicted means for full flow. TinyCNN model.

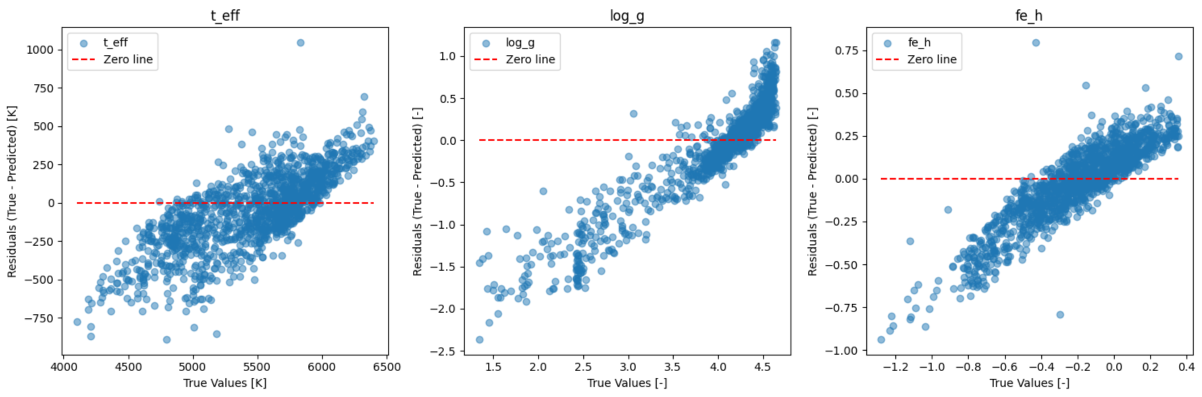

Figure 16: Scatter plot of predicted means residuals for full flow. TinyCNN model.