Team Discount GPT: Large Binocular Telescope Misalignment Project Final Report

Ryan Luu[1], Bowman Brown[1], Alfianto Widodo[1], Mahmoudreza Dehghan[1], and Deqing Fu[1]

[1]Department of Computer Science, University of Southern California, 941 Bloom Walk, Los Angeles, CA 90089 , USA

ABSTRACT

The Large Binocular Telescope (LBT) observatory team aims to accelerate the telescope misalignment correction process by incorporating machine learning into its workflow. As a novel initiative within this domain of optical science, astronomy, and machine learning research, this project endeavors to investigate various machine learning models suitable for real-world applications on the LBT telescope. Our study proposes a machine learning model that performs precise regression analysis by leveraging image data and Point Spread Function (PSF) subsystem data to predict control parameters for correcting misalignment issues, thereby replacing the existing LBCFPIA system that is currently being used to correct misalignment on the telescope. We experimented with basic regression algorithms, Convolutional Neural Networks, and Vision Transformers, to adequately explore different model options for this specific problem. Despite dataset limitations, we achieved a promising Mean Squared Error (MSE) result of 0.33 on one of our trials with the ResNet18-MLP model, indicating potential for success for this project.

1. INTRODUCTION

The Large Binocular Telescope is a state-of-the-art observatory situated on Mount Graham, Arizona that features two 8.4 meter wide mirrors instead of the traditional single optic seen on most modern giant telescopes. However, the utilization of two primary optics has led to unique misalignment issues that may result in sections of blur on the captured images. The LBT team is currently grappling with pinpointing the origin of this misalignment, which could stem from optical, mechanical, or software-related issues due to the complex nature of the system. To address these challenges, the team has developed a custom interface program called the Large Binocular Camera Focal Plane Image Analysis (FPIA) which performs geometric analysis of extrafocal pupils to determine focus and wavefront corrections via third-order spherical aberrations.[1] Despite the effectiveness of this system, its time-consuming nature, requiring anywhere from 5-40 minutes per iteration over multiple iterations that depends on specialized human input for accuracy, necessitates an alternative approach.

An essential aspect of effective data analysis necessary to develop our machine learning models is to first comprehend the source of the problem at hand. As such, our research on the Large Binocular Telescope's (LBT) active optics system began with a review of the telescope's workings. The paper "Prime Focus Active Optics with the Large Binocular Telescope" authored by J. M. Hill[1] provides details on the development and implementation of the current active FPIA optics system, including tests conducted to assess the LBT's performance. The study established that FPIA enhanced image quality by reducing the effects of atmospheric turbulence, correcting mechanical issues, and was able to successfully compensate for telescope misalignment.

A component that directly interacts with FPIA to fix the misalignment is the Point Spread Function (PSF) subsystem. It receives correctional data from FPIA and characterizes the imaging system's response to a point source such as a star, describing the pattern that results when the point source is projected onto a detector. It measures the telescope's image quality, influenced by various factors such as optical design, atmospheric conditions, and aperture size and shape. The paper goes on to outline how implementing FPIA on the LBT enhances the PSF and thereby improves the telescope's image quality. The active optics system adjusts the primary and secondary mirrors' positions in real-time, compensating for changes in the environment and reducing incoming light distortion, which leads to sharper and clearer images. Our simulated data will incorporate these ideal PSF conditions as fundamental parameters in training the model.

2. METHODS AND APPROACHES

2.1 Dataset

The objective of our research is to develop a preliminary model capable of identifying misalignment information in an optical system through analyzing the Point Spread Function (PSF) subsystem. However, due to the intricate nature of the data parameters and the current scope of the project, this research utilized synthetically generated data based around the physical specs of the LBT as opposed to actual data and imaging taken from the telescope. For the purpose of our investigations, we will consider a simplified PSF, defined as a two-dimensional (2D) image that represents the spot shape generated by the optical system. The input parameters for our algorithm comprise positional and tilt coordinates (d_x, d_y, d_z, t_x, t_y), PSF position (p_x, p_y), field coordinates (field_x, field_y), and the image of a cropped out star shown in table 1 and figure 1.

Table 1. Example Line from CSV of Generated PSF Dataset

	d_x	**d_y**	**d_z**	**t_x**	**t_y**	**p_x**	**p_y**	**field_x**	**field_y**	**data_img**
0	-2.0	-2.0	-2.0	0.0	0.0	52.026...	51.736...	0.104	-0.104	”[[0. 0. ... 0. 0.]

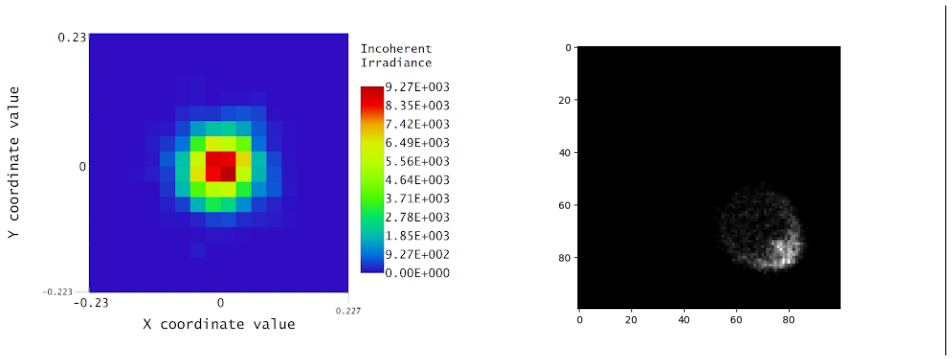

Figure 1. Mimicking our star image using Gaussian about the light source contained within our field coordinates (left). Image of Ray-Traced Generated Star Image produced by Zemax (right).

However, the PSF position coordinates (p_x , p_y) initially generated by the ray-tracing program cannot directly be used. Those coordinates indicate the cropped-out zone position since the entire detector plane was too large for our simulation, with only a small section containing the PSF being used in the generated data. To address this and enable applicability to actual telescope data, we determined the true PSF positions using centroiding. The process is described in Equation 1 of the paper by Wan et al.[2] We added the centroiding result to the initial generated PSF positions (p_x, p_y) to derive the true PSF position, as expressed in the following equation:

$$p_xTrue = \frac{\sum(I_i \cdot x_i)}{\sum I_i} + p_xinital, \quad p_yTrue = \frac{\sum(I_i \cdot y_i)}{\sum I_i} + p_yinital \tag{1}$$

As previously noted, this study restricts its focus to determining misalignment attributable solely to the three freedoms of movement along the x, y, and z axes of the telescope as shown in Figure 2. Although there could be other parameters contributing to misalignment, this project's scope is limited to movement and tilting of the telescope across these three axes. Hence, our model aims to predict the low order aberration misalignment parameters d_x, d_y, d_z, t_x, t_y, given the input image and p_x, p_y from our PSF subsystem.

2.2 Loss Functions

In our preliminary experiments, we have employed the Mean Squared Error (MSE) as the loss function for our baseline models. However, we acknowledge that MSE may be less resilient to the presence of outliers in our training dataset. To address this limitation, our research aimed to explore alternative loss functions that can facilitate more accurate regression analysis, while maintaining robustness against outliers. One potential

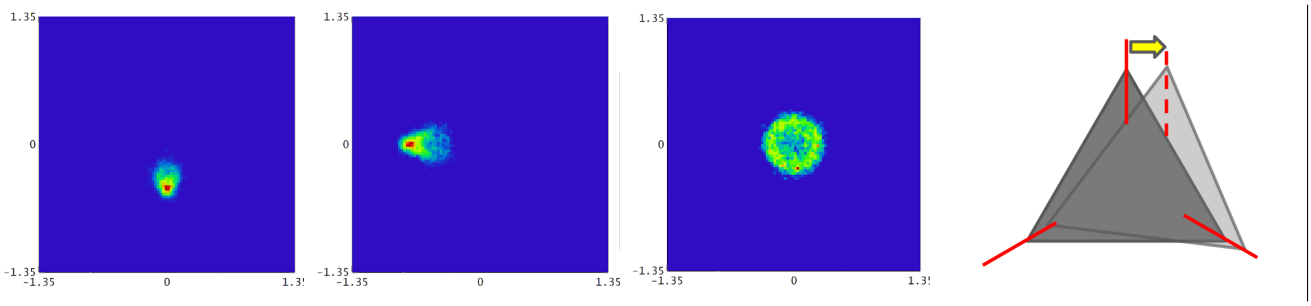

Figure 2. Depiction of the impact of transitional movement and tilting of the telescope around the x(left), y(center left), and z(center right) axis on the star image, resulting in misalignment. The present study focuses solely on the problem of misalignment attributable to these particular variations in orientation, with the three freedoms of movement being visualized on the (right).

candidate is the Huber loss 2, which amalgamates the advantageous properties of both Mean Absolute Error (MAE) and MSE.

$$L_\delta = \begin{cases} \frac{1}{2}(y-\hat{y})^2 & if\ |(y-\hat{y})| < \delta \\ \delta((y-\hat{y}) - \frac{1}{2}\delta) & otherwise \end{cases} \tag{2}$$

Huber loss applies a quadratic penalty when the difference between the true and predicted value is less than the hyperparameter delta. Otherwise, it applies the absolute value of the difference as penalty. Our research aims to explore and potentially combine variations of the standard Huber loss as presented in the following studies: "Generalized Huber Loss"[3] and "Alternative Probabilistic Interpretation of the Huber Loss".[4]

The paper on Generalized Huber Loss proposes an algorithm to efficiently find the parameters that lead to minimizing the loss for robust learning and efficient minimization. The paper on Alternative Probabilistic Interpretation of the Huber Loss relates minimizing the loss to minimizing an upper-bound on the Kullback-Leibler divergence between Laplace distributions, where one distribution represents the noise in the ground-truth and the other represents the noise in the prediction. This approach is also tested on vision tasks such as object detection with RetinaNet[5] and R-CNN[6] which goes along with the objective of this research. In our experiments, our simple CNN model in 3.1 with a tuned Huber loss performed better than MSE loss. The detailed results can be seen on table 2.

Table 2. Huber Loss vs MSE Loss results

	MSE	**Huber** $\delta = 1.0$	**Huber** $\delta = 0.95$	**Huber** $\delta = 0.9$
MSE score	0.6273	0.5512	0.5490	0.6143

Despite the promising results obtained with Huber loss in our Convolutional Neural Network (CNN) model, we opted to use Mean Squared Error (MSE) loss for our future experiments to ensure standardization across various models. The challenge of tuning Huber's δ in addition to the model architectures themselves within the scope of our research prompted us to choose the simpler MSE loss going forward.

2.3 Regression Algorithms

We employed fundamental regression algorithms without the use of the image as baseline models, given the nature of the problem. The Large Binocular Telescope Misalignment issue presents a unique and intricate challenge, and we anticipated that the performance of these basic models would be constrained on the proposed dataset. Nevertheless, our objective was to investigate the behavior of these models on our dataset to gain insights into their applicability and limitations.

We used R-Squared (R^2) as our evaluation metric for the regression algorithms. The best R^2 score was obtained by the Random Forest algorithm with a value of 0.484. Results of the other methods can be found on

Regression Algorithm	R-Squared Score
Gradient Boosting.	0.4685
SVM.	0.4678
Random Forest.	0.4642
Linear Regression.	0.4402
KNN.	0.3735

Table 3. Results of our regression experiments

Table 3. Overall these R^2 scores confirm our expectation that these simple models could not explain a significant amount of the variance in the target variable without a more robust dataset.

One potential explanation for the restricted performance of these models is their limited capacity to capture the various factors contributing to the misalignment. Furthermore, the dataset may have exhibited noise and high-dimensionality, posing challenges for these models to effectively address with just regression alone. To mitigate these limitations, it is essential to investigate the more sophisticated deep learning algorithms outlined in the subsequent sections. In summary, although the results were anticipated, examining the performance of these foundational models offers a valuable baseline for future comparisons and a starting point for enhancing prediction accuracy.

2.4 Deep Learning Algorithms

Due to the specific nature of our task, our deep learning approaches to solving the problem were focused on exploring different combinations of architectures, model types, and hyperparameter configurations that produced the best results.

2.4.1 Convolutional Neural Network

To establish a baseline for our deep learning methods, we constructed a 'simple' Convolutional Neural Network (CNN) based model. CNNs have been shown to perform well on image processing tasks,[7] but have had limited success in high-precision regression domains. We utilize the CNN-based architecture as a baseline for our project to provide valuable insights that are used to refine the structure of our more advanced deep learning approaches.

The 'Simple CNN' used a two-part structure. In the **image-architecture**, the input image is processed with a CNN architecture consisting of the sequence: conv layer, max pooling, conv layer. This produces an output vector that is concatenated with the two input continuous scalar values. The concatenated input is fed into the second part of the model, the **combination-architecture**, consisting of a fully connected layer with ReLU activation functions applied followed by another fully connected layer as the regression head to predict our five continuous output variables.

2.4.2 ResNet-MLP

We expanded on the Simple-CNN by replacing the image-architecture with a ResNet[8] model and adding additional layers with ReLU activation functions to the combination-architecture. We used the ResNet architectures provided by the TorchVision package.[9] Our approach explored leveraging larger models to improve the the information content of the image-architecture output feature vector. We compared ResNet-18 and ResNet-50 models and found that Resnet-50 models performed worse than ResNet-18 models on average.

Furthermore, we varied the size of the image-architecture output feature vector as powers of two in the range $2^3 - 2^{10}$. We found that a feature vector size of $2^6 = 64$ lead to the best performance for both ResNet models, and was consistently able to outperform the Simple CNN. We concluded that processing the image was more difficult that processing the continuous variables. Additionally, based on our analysis of the image-architecture output feature vector size, the images contain few informative features.

2.4.3 Vision Transformer

Transformers have recently gained popularity for image processing with the introduction of the Vision Transformer.[10] Vision Transformer is an architecture that is specifically configured for image processing tasks, and has been shown to perform better than CNN models.[10] In an attempt to improve the informativeness of the image-architecture output feature vector, we used the ViT-16 small model to process the images. We used the Hugging Face Transformers library[11] to implement the Vision Transformer (ViT). Since ViT has a large computational cost, we performed hyperparameter tuning using experiments of only 25 epochs.

We conducted experiments using both pre-trained weights provided by Hugging Face and random weight initialization to compare the results. The pre-trained Hugging Face weights demonstrated better results and faster convergence of the loss function. Nonetheless, it is important to note that further in-depth experimentation is necessary to ascertain the extent to which pre-trained weights improve the final results.

The ViT model was unable to achieve a better top-1 or mean accuracy than either of the ResNet models. We hypothesize that is due to the limited dataset as ViT architectures have been shown to perform worse under low data constraints.[12] For our task, where data is scarce and no similar open-source dataset exist, transformers do not appear to be the most viable option.

2.4.4 Pre-training

The results of our transformer-based model lead us to implement a pre-training strategy in an attempt to leverage existing open-source datasets. Pre-training has been found to improve performance for tasks with limited data, even when the pre-training task is different from the training and testing tasks.[13]

We chose to construct a new pre-training dataset of 12,000 examples based on MNIST for our task. Similar to our dataset images, MNIST images are grey-scale and contain small, dense regions of pixel values that inform the results. To create pre-training images, we first created a 100x100 black 'canvas' of pixels. Five random MNIST images were selected and then aligned in the top left, top right, bottom left, bottom right, and center of the canvas. The label was constructed as the five real values corresponding to the five MNIST image digits.

We expected that pre-training would improve results by aligning the initial weights of the model to leverage informative connections and features in the input images. Our pre-training dataset did not incorporate the continuous-value px and py inputs, and was only directed towards improving the information content of the image-architecture output feature vector.

2.4.5 Ensembles

We explored three different ensemble architectures, each of which used 10 randomly initialized sub-models. 'Simple-CNN Ensemble' used an ensemble of the Simple-CNN architectures described in 2.4.1 with results produced by averaging the output of each model, 'ResNet-18 Average Ensemble' used an ensemble of Resnet-18 models and also averaged the outputs to create the results, and 'ResNet-18 Attention Ensemble' used an ensemble of Resnet-18 models with a self-attention layer to combine the outputs of each model.

2.4.6 Attention Combination

To take advantage of the interaction between the image-architecture output vector and the continuous input scalars, we applied self-attention to the concatenated vector before it was passed into the combination-architecture. The attention layer was used to enable the model to 'attend' to the only most important features, which we expected to be the scalar inputs and a small subset of the image-architecture output features (based on our analysis in section 2.4.2).

Additionally, we experimented with using a self-attention layer to 'attend to' the output of different sub-models in one of our ensemble models. The principle behind this approach was similar to above, where the attention layer would dynamically learn to attend to more important ensemble sub-models, thus increasing overall performance.

3. EXPERIMENTS

For all trials, we trained for a total of 100 epochs. We used a learning rate of 0.0001 for all experiments except for the ViT architecture, which used a learning rate of 0.00001. Models with a '-A' flag in the name indicate that self-attention was applied to the concatenation of the image-architecture output and the input scalars. Models with a '-HD' flag in the name indicate that the input scalars were mapped to a higher dimensional space before being concatenated with the image-architecture output.

Each model is run for 20 total trials, 10 trials using 'normal' training with no pre-training and 10 trials with pre-training. Pre-training iterates over the custom dataset described in section 2.4.4 for 2 epochs.

Results are calculated by re-running the iteration of the model that had the highest validation score on the unseen test dataset in order to limit issues with over-fitting. Our results can be found in tables 4 and 5.

	Simple-CNN	ResNet18-MLP	ResNet50-MLP	ResNet18-A	ResNet18-A-HD	ViT
Normal Acc	0.5993	0.4178	0.4611	0.4246	0.4204	0.9012
Normal Std	0.0482	0.0208	0.0492	0.0267	0.0234	0.1345
+Pretrain Acc	0.8906	0.3744	0.4333	0.4108	–	0.9292
+Pretrain Std	0.0837	0.0436	0.0415	0.0157	–	0.1084

Table 4. Individual Model Results

	Simple-CNN Ensemble	ResNet-18 Average Ensemble	ResNet-18 Attention Ensemble
Normal Acc	0.4091	0.4204	0.4579
Normal Std	0.0215	0.0148	0.0274
+Pretrain Acc	0.6322	–	–
+Pretrain Std	0.1018	–	–

Table 5. Ensemble Model Results

3.1 Simple-CNN

Training was performed with Early Stopping in order to account for over-fitting using Adam optimizer and Huber loss. We managed to achieve a MSE Score of 0.5490. Results of training can found in figure 6 below. The Simple-CNN model served as an effective baseline against which our other deep learning approaches were compared.

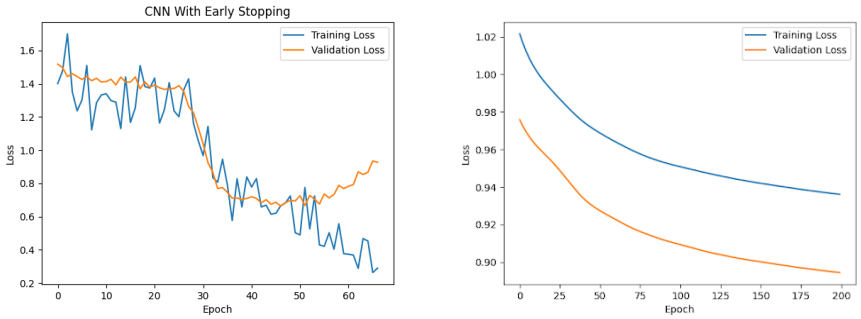

Figure 3. Results of first CNN Trials using loss functions Adam (left), and SGD(right). For the CNN, the Mean Squared Error (MSE) for Adam was 0.5490 and SGD was 0.9741

From figure 6's CNN training chart, we observed that the model was overfitting with Adam which led to experimentation with SGD and its variations. However, the model achieved a higher accuracy with Adam and converged considerably quicker.

3.2 ResNet-Based Models

Single ResNet-based models outperformed all of the other architectures we explored. Surprisingly, the ResNet-18 model using pre-training without attention showed the best performance, beating the larger ResNet-50 by 0.0433 MSE. Our original expectation was that pre-training would provide more benefit for the larger models such as ViT and ResNet-50. However, ResNet-18 showed the greatest difference in performance, 0.0434 MSE, when pre-training was applied.

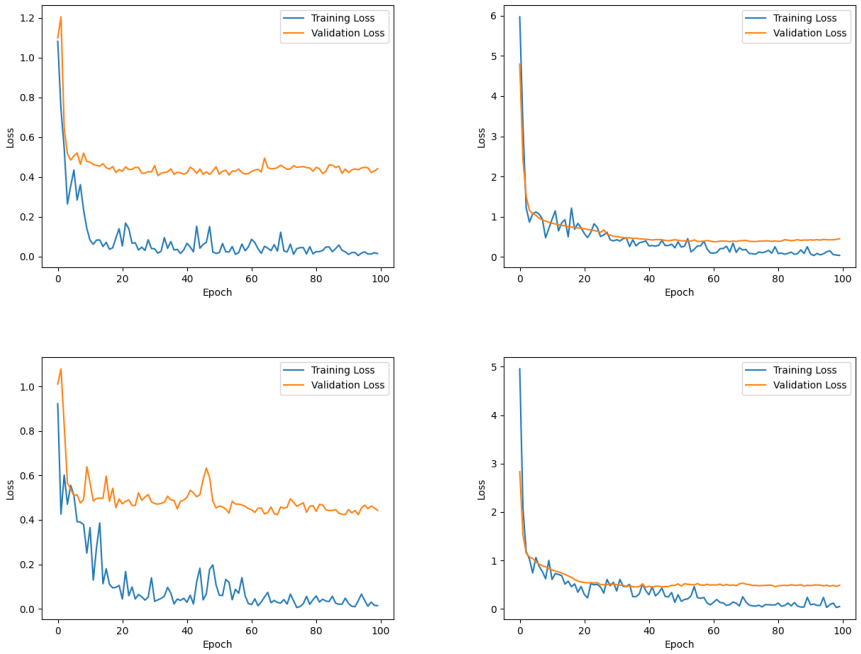

Figure 4. Training Results of the ResNet models. ResNet-18 (top) outperformed ResNet-50 (bottom). The training and validation loss are shown for models without pre-training (left) versus with pre-training (right).

3.3 Vision Transformer Based Models

The ViT-based model was the worst performing model. This may have been a user-issue with the experimental setup, caused by sub-optimal hyperparameters, or a result of known performance issues with ViT when the dataset is small.

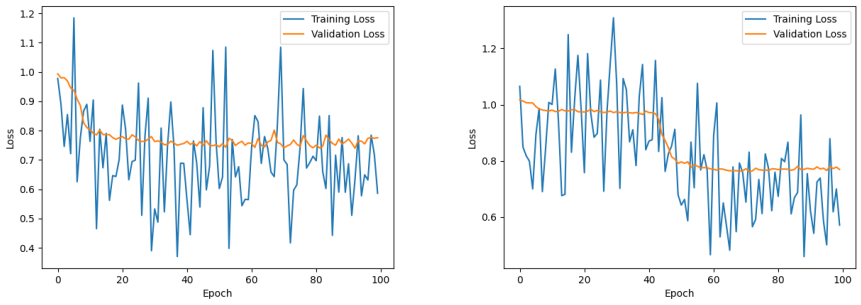

Figure 5. Training Results of the ViT models. The training and validation loss are shown for models without pre-training (left) versus with pre-training (right).

3.4 Ensemble Models

Ensemble models showed stable performance, but could not beat the single ResNet-18 model with pre-training. The additional cost of executing multiple ResNet-18 networks contributed to a high cost of tuning. As such, the ensemble models used the optimal hyperparameters found for the single models, which may have contributed to the slightly worse performance.

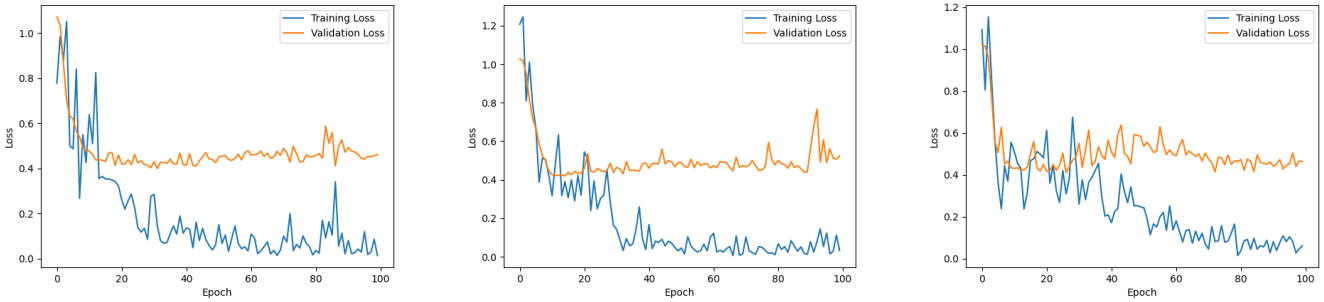

Figure 6. Training Results of the Ensemble models. Training and validation loss are shown for Simple-CNN Ensemble (left), ResNet-18 Average Ensemble (center), and ResNet-18 Attention Ensemble (right).

4. CONCLUSIONS AND FUTURE WORK

With the best performing MSE score of 0.3744, averaged over 10 runs, on our limited dataset using ResNet18, we have demonstrated the potential for the success of the project. We fully expect that this approach using Machine Learning, with some further tweaking to the dataset and model, will yield successful and practical results in the near future.

Moving forward, new approaches to combining the image and scalar input information need to be explored. Our initial analysis had focused on deep learning techniques for regression. However, given the multi-modal nature of our input data, more advanced methods for data combination may provide significant improvements to our results. The more advanced Deep Learning models have not yet shown good results, likely due to issues with tuning or implementation.

Future improvements for the project include the acquisition of additional data, either through real examples collected from the LBT, variation on the current data to gather more input parameters, or through improved pre-training datasets that more closely mimic the task. Furthermore, future Deep Learning models should include components of the current manual method that are cheap to pre-compute as potential input variables, similar to existing work on physics-informed modelling.[14] This reduces the need for the Deep Learning model to learn easily-computed rules or components of the function from scratch, and can reduce the size of the model needed to solve the task.

Finally, more fine-tuning of hyperparameters may also be necessary. Our work so far has consisted of prototyping models and implementing baselines from previous research. Results may be strongly influenced by the hyperparameter selection, and as such, tuning hyperparameters is an important step in delivering useful models. With all these considerations, we fully expect the problem at hand can be solved in the near future.

REFERENCES

[1] Hill, J. M., Ragazzoni, R., Baruffolo, A., Biddick, C. J., Kuhn, O. P., E. Diolaiti, D. T., and Rakich, A., "Prime focus active optics with the large binocular telescope," in [*Astronomical Optics: Design, Manufacture, and Test of Space and Ground Systems II*], **11116**, 63–72, SPIE (2008).

[2] Wan X, Wang G, W. X. L. J. and G, S., "Star centroiding based on fast gaussian fitting for star sensors," **2836**, 9–18 (August 2018).

[3] Gokcesu, K. and Gokcesu, H., "Generalized huber loss for robust learning and its efficient minimization for a robust statistics," (2021).
[4] Meyer, G. P., "An alternative probabilistic interpretation of the huber loss," (2020).
[5] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P., "Focal loss for dense object detection," (2018).
[6] Girshick, R., Donahue, J., Darrell, T., and Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," (2014).
[7] O'Shea, K. and Nash, R., "An introduction to convolutional neural networks," (2015).
[8] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," *CoRR* **abs/1512.03385** (2015).
[9] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., "Pytorch: An imperative style, high-performance deep learning library," *CoRR* **abs/1912.01703** (2019).
[10] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N., "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR* **abs/2010.11929** (2020).
[11] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J., "Huggingface's transformers: State-of-the-art natural language processing," *CoRR* **abs/1910.03771** (2019).
[12] Zhu, H., Chen, B., and Yang, C., "Understanding why vit trains badly on small datasets: An intuitive perspective," (2023).
[13] Hendrycks, D., Lee, K., and Mazeika, M., "Using pre-training can improve model robustness and uncertainty," in [*Proceedings of the 36th International Conference on Machine Learning*], Chaudhuri, K. and Salakhutdinov, R., eds., *Proceedings of Machine Learning Research* **97**, 2712–2721, PMLR (09–15 Jun 2019).
[14] Karniadakis, G., Kevrekidis, Y., Lu, L., Perdikaris, P., Wang, S., and Yang, L., "Physics-informed machine learning," 1–19 (05 2021).

5. CONTRIBUTIONS

Everyone so far has been contributing to both the process of this research project, including both the presentations, previous reports, as well the final report here. No complaints from any of the team members in regards to contributions.

Ryan Luu - Worked on Generating the Dataset, Regreession Models, and the CNN Model.

Bowman Brown - Worked on Deep Learning Models such as Vision Transformer, Resnet-18 Ensemble, Resnet-MLP, etc.

Alfianto Widodo - Worked on research and experiments for CNN with SGD and Huber loss

Mahmoudreza Dehghan - Worked on researching basic regression algorithms and implementing them.

Additionally, it is worth noting that all team members have participated actively in editing each other's sections and the overall report as a whole.

Our github repo can be found at https://github.com/rluuy/lbt_alignment