High Entropy Alloy Analysis
CHEMENG 4H03: Project Technical Memo

Justin Covach – covachj – 400243271
Mark Fahmy – fahmym12 – 400319835
Karl Rizk – rizkk – 400316135

The objective of our model is to predict the material properties of high-entropy alloys, given the atomic composition and processing method of this alloys. This is useful because manually testing the material properties of high-entropy alloys is expensive and time consuming; creating a model that can predict material properties can significantly speed up and cheapen the cost of material development. To find this value in our dataset, we will implement neural networks as well as some clustering using K-prototypes. These two methods will provide predictive ability and insight into our data and potentially future data. Using Neural Networks to predict alloy performance has been attempted before, but has only become viable in recent years due to the improved availability of datasets and computational capacity [1].

Reference Materials

The full code used for the analyses outlined in this technical memo are available from the CovJus01/high-entropy-alloys-project GitHub repository. It is recommended to review the `README.md` for this repository, as it provides a more in-depth overview of the project structure and key files used. The `/src/` directory contains all Python scripts used to produce the models whose results are outlined in this memo, and the `/figures/` directory contains the results which aided in tuning model parameters. The `/data/` directory contains the original dataset which was obtained from Kaggle (kaggle.com/datasets/sethpointaverage/high-entropy-alloys-properties/data) and the parsed CSV which was used to train our model. The Kaggle data was obtained from a 2021 journal article by Machaka et al. [2]. Exclusion criteria and parsing methods are described in Section 1.1.

Contents

1 Methodology

The dataset contains 1545 entries with 16 columns in each entry. The first column ["Identifier: Reference ID"] contains a numerical value that identifies the research article which the alloy properties were obtained from, and can be cross-referenced with a GitHub repository which associates each Reference ID with a journal article. The following 4 columns are string formatted values that contain the chemical formula, microstructure, processing method and whether the alloy falls in BCC/FCC/other. The formula contains a set of numerical values that are associated with the corresponding element in the formula. The other string values contain no numerical values and are all categorical in nature. The following 11 columns describe various properties of high entropy alloys. All of these values are numerical in nature and require no additional preprocessing steps other than centering and scaling.

1.1 Data Processing and Parsing

Our dataset contains many empty values within each column. These null values can affect many types of analysis if not handled correctly. When using analysis methods that are unable to handle missing values, the empty values were assigned the mean of their respective columns, which allowed for the use of methods like K-prototypes.

Some entries were removed from the dataset as they did not add any meaningful information to the dataset (eg. no data except for alloy composition), or were extremely erroneous (containing values that deviated by more than 20% from their predicted values). Since processing method was a key parameter, any entries which had less than three columns of numerical data and no processing method were removed when the processing method could not be found in the reference articles. This resulted in 17 entries being removed, while approximately 15 were populated with information from their original journal article [3].

Additionally, the original dataset contained a number of columns which were removed entirely due to sparsity or redundancy. These columns were 'O Content (wppm)', 'N Content (wppm)', and 'C Content (wppm)' which each had only 57, 45 and 4 entries, respectively, and were therefore unusable.

Due to the various string values in the dataset, we first had to parse the different columns to extract the data in a way that would be usable for our analysis. The first parsing (and the most complicated) extracts the various unique elements from the string values, defines a column for each element, and then extracts the value for each element, setting ones that do not exist to 0, and ones that do exist to their corresponding value. This substantially increases our column count due to the sheer number of unique elements. Likewise, the microstructure column also adds many columns due to the number of unique microstructures. The microstructure as well as the remaining categorical columns we all one-hot encoded to represent the data in their columns numerically.

For the non-categorical columns, we also performed the preprocessing step of centering in scaling. This was performed on a per column basis and ignored missing values. The mean and standard deviations of the columns were often stored for later use in postprocessing to make sure the results match the expected values in their dimensions. These processes were implemented using NumPy's built in `nanmean()` and `nanstd()`.

Furthermore, certain analyses, such as t-distributed stochastic neighbor embedding, cannot tolerate `NaN` values; so, imputation was conducted such that every NaN value was replaced by the mean of the numerical values in that column. All categorical variables were present.

1.2 PCA

For our PCA analysis, we employed the regular PCA nipals algorithm. This method was chosen because it was necessary that the model ignore missing values. This again was implemented by using the NumPy methods that ignore the missing values within our data.

1.3 Clustering Analysis

1.3.1 K-prototypes

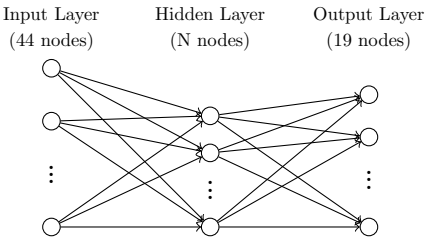
For the clustering analysis conducted, we elected to use K-Prototypes over K-Means or K-NN because K-Prototypes are natively equipped to handle categorical variables, without the need for one-hot encodning. The method was implemented using the Kprototypes Python library.

1.3.2 t-Distributed Stochastic Neighbor Embedding (tSNE)

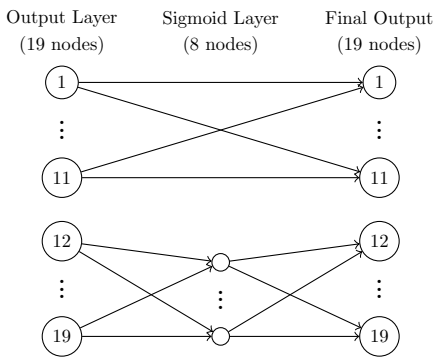
Coupled with the K-Prototypes analysis, t-SNE dimensionality reduction was used to give us a better understanding of the clusters. We chose to run this analysis alongside with PCA, because the nature of our data caused the PCA to output weak results; since t-SNE tries to preserve the distances between points, it was thought that it would give a superior visualization.

1.4 Large ANN Model

The main model was developed using TensorFlow's sequential model. The dataset was shuffled then split into training, validation and test sets. The split was 1000/400/129. Before entering the model, we separate the model inputs from the outputs. The input to the model was the parsed atomic formula, processing method and type of test done. The basic model structure was a set of hidden layers of width N using ReLU activation, followed by a linearly activated output layer. Since our dataset involves both categorical and continuous variables, we also have to pass the categorical values through a sigmoid activation layer before processing the cost. The base model structure and the handling of the categorical values is shown below.



(a) General Model Structure



(b) Categorical value handling

Figure 1: General model structure (left) and categorical output handling (right).

To back propagate effectively, we needed to build a hybrid loss function. To do this we sectioned off the outputs into categorical and regressive types, based on the expected output. Using this structure, we calculated the mean squared error for the regressive columns (Ignoring NaNs) and the binary cross-entropy loss for the categorical columns since they were all one hot encoded during preprocessing. This was all done using TensorFlow's loss functionality but by creating our own custom loss function. To determine the effectiveness of the model, we used this loss as a metric as well as a confusion matrix for the categorical columns. To find the most effective model, we iterated over different model structures, changing the number of nodes in each layer and the depth of the model. This sweep allowed us to determine which model we believed would handle our problem most effectively.

1.5 Divided ANN Model

Due to the sparse nature of the dataset, an alternative architecture was investigated. In this architecture, three submodels, were trained on rich, interrelated subsets of the data; their output was then in turn used as an input for another neural net which amalgamated the outputs of each net. The divisions were:

1. Mechanical Properties: Hardness Value (HV), Ultimate Tensile Strength (UTS), Yield Strength (YS), Elongation
2. Phase Properties: B2, BCC, FCC, Secondary, HCP, etc.
3. Density and Modulus: Experimental and Calculated Densities, Experimental and Calculated Young's Modulus

2 Results

2.1 PCA

The PCA loadings for the first three components are provided in Figure 2, and provide a valuable insight into some of the relationships between the variables. These results are discussed further in relation to the clustering analysis and ANN model discussion.

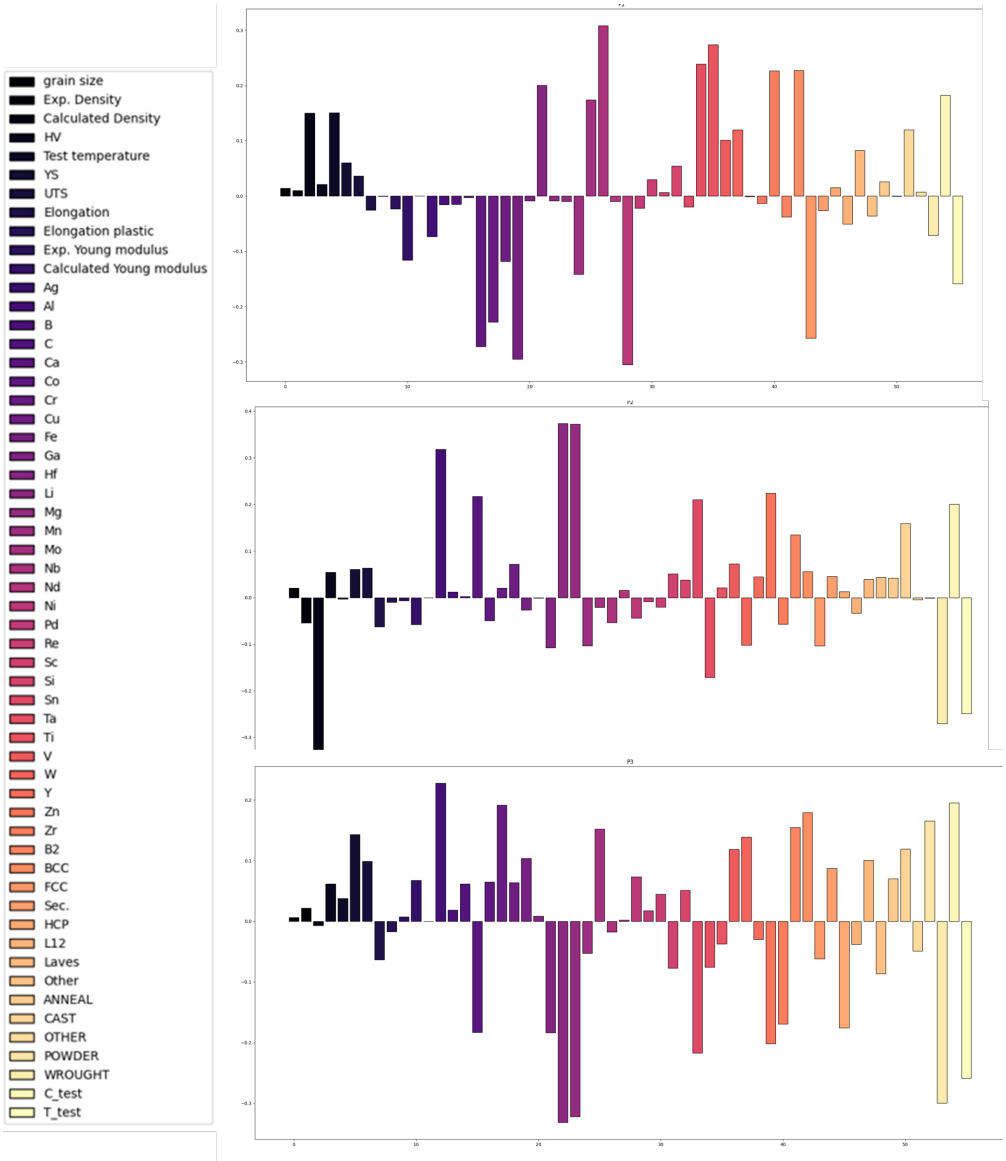
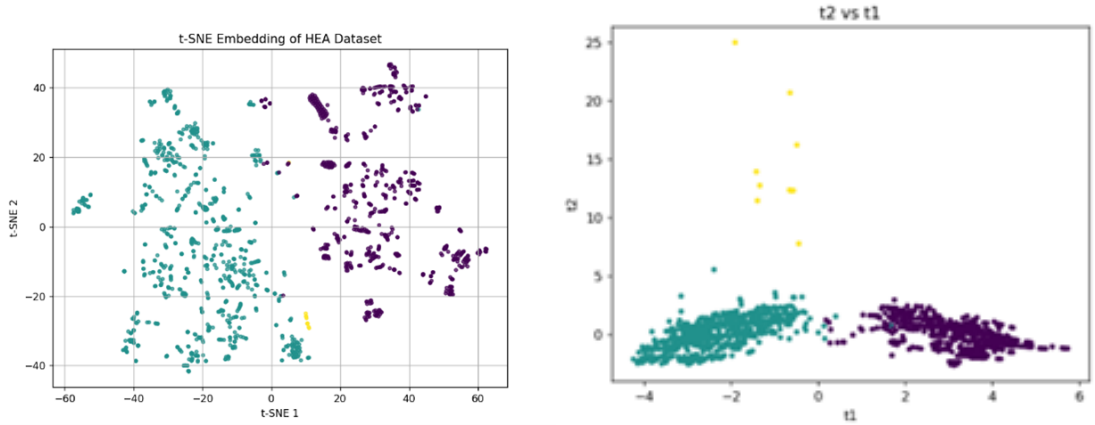


Figure 2: Loadings for first three principal components obtained from PCA (first at top, second in middle, third at bottom).

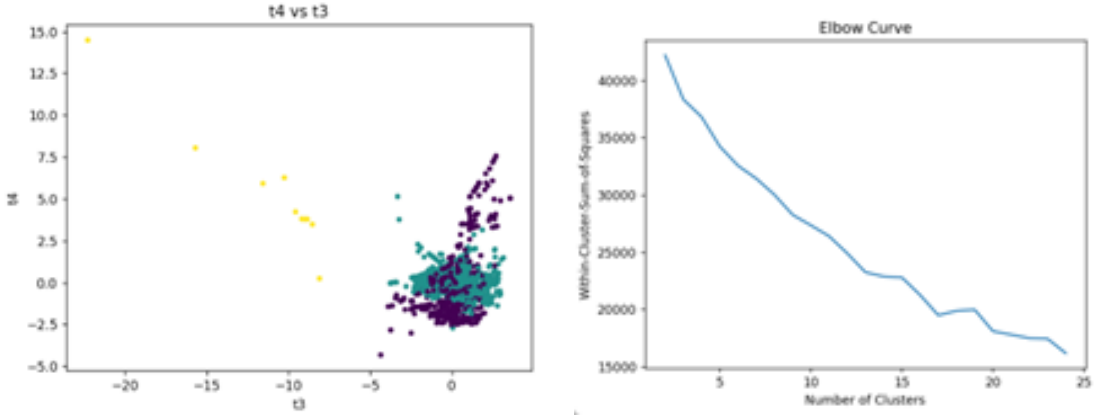
2.2 Clustering Analysis

The results of the clustering analysis method described in part 1.3 are provided below in Figures 3 and 4.



(a) tSNE Embedding of Dataset (b) t_2 vs t_1 score plot

Figure 3: tSNE Embedding (left) and score plot of first two components (right).



(a) t_4 vs t_3 score plot (b) Elbow curve.

Figure 4: Clustering results for principal components 3 and 4 (left) and elbow curve for K-prototypes (right).

2.3 Large ANN Model

After performing the sweep of the different ANN structures, the progression of the model was recorded and plotted below in Figure 5. On the left we begin with hidden layer width of 2 nodes and progress to our extreme case of 512 nodes.

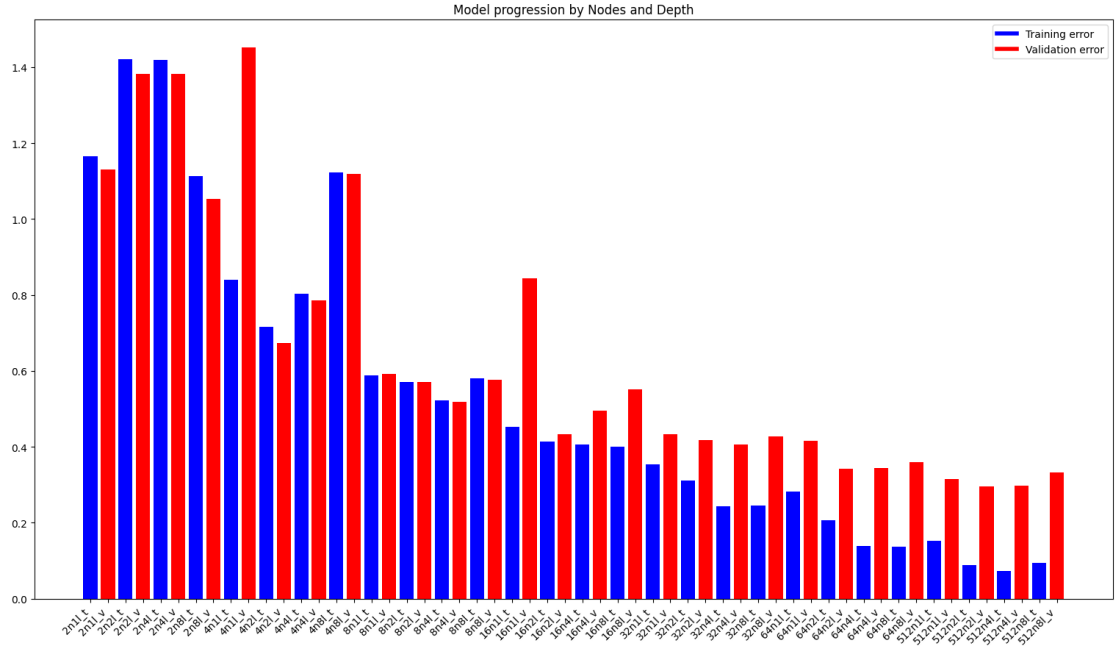


Figure 5: ANN model progression as width is swept between 2-512, and depth varies between 1-8.

From the above chart, we determined that the model that would perform best for us would be "512n2l" which corresponds to the model with 512 nodes in each layer, and a depth of 2. Below in Figure 6, a comparison of predicted outcomes and true values are shown, in addition to the confusion matrix values for the different outputs. This Figure indicates that the model performs very strongly and has the strongest predictive power for the more populated columns such as 'Calculated Density', 'Yield Strength', 'Ultimate Tensile Strength', etc., although the performance deteriorates substantially for less 'rich' columns such as 'Elongation' and 'Grain Size'.

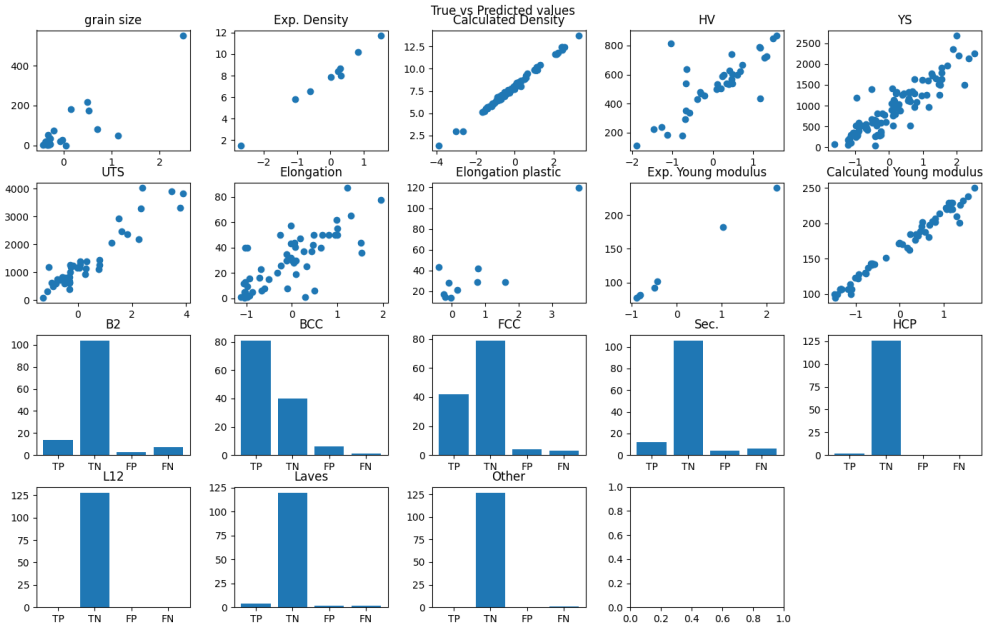


Figure 6: Predictive capability of model "512n2l".

2.4 Divided ANN Model

The divided model was tested with varied batchsizes of either 16, 32 or 64, and Epochs of 250 and 500 for the submodels and 1000 Epochs for the final model. The full results are available in the GitHub repository, but the training and validation loss revealed that the Epochs for the final model should be reduced to < 150 in order to avoid overfitting, as shown below in Figure 7. For this reason, the network was set to only use 125 Epochs in the final revision. Additionally, comparing the outcomes found that the ideal model outputs were obtained at a batch size of 32, as a lower batch size performed poorly, and larger batch sizes did not improve performance.

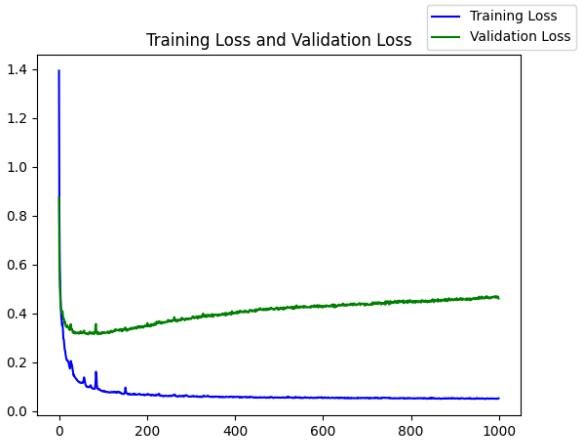


Figure 7: Training and validation loss for submodel batch size of 32, 250 Epochs and 500 final model Epochs.

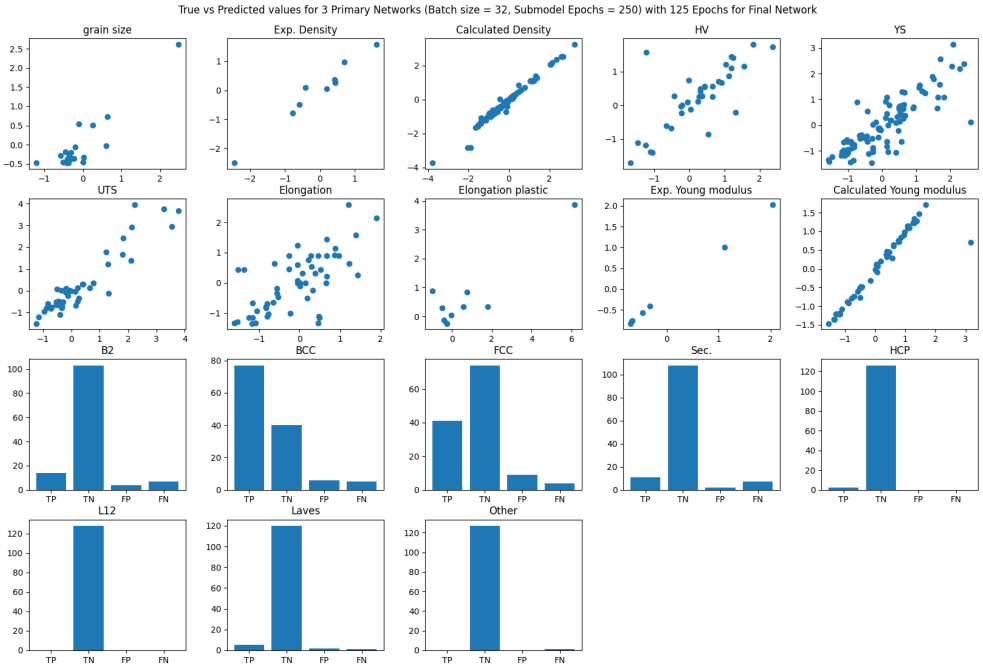


Figure 8: Model predictions for ideal model with 125 Epochs in final model.

3 Discussion

3.1 K-Prototypes

As can be seen from the elbow curve in 4b, there does not appear to be an elbow that would indicate an optimal number of clusters; three clusters seems to give the most coherent results, visually. It seems that the clustering works well for t-SNE and the first and second components, in the case of principal components one and two, the clustering accurately groups the two main bodies as well as a collection of outliers. However, the clustering seems to fail when looking at principal components three and four. The actual clustering seems to indicate that the main groups are in microstructure for the differentiation of T1, FCC & BCC, and the yellow outliers indicate high quantities of Li and Mg.

3.2 Large ANN Model

From our sweep of the different neural network structures, we can evidently see that the model is minimizing the loss in the outputs it predicts. In the initial stages where the model width is very small, the model struggles to effectively predict the data and begins to predict the same value every time. This is a strong indicator of a lack of capacity to interpret the problem. As we progress into more depth and width we can see the model begins to quickly decrease the loss in the training set as well as the validation set. At a model width of 64 the validation and training sets begin to seperate quicker as the validation loss decreases less and the model learns the training set better. We decided to pick the "512n2l" model since it decreased the validation loss the most, indicating that it is the best model to generalize well. In Figure 6, we see that the model is very effective at predicting the Exp. Density, Calculated Density, Exp Young's Modulus and Calculated Young's Modulus. We also see that the model has significantly more correct predictions for the categorical values. From our PCA, we can also see why the model is worse at predicting all the other values. Our P1-P3 loadings plot in Figure 2 displays a much higher correlation between the composition values and the material properties that we predict effectively. The magnitude or correlation for the poorly correlated properties only begins to be evident at our 8+ components. This seems to be why our model is not as effective, but still quite good, at predicting those properties. To increase the effectiveness it may be useful to collect more data corresponding to those properties and feed them into the model or potentially first predicting the high correlation variables and inputting them into a second model to predict the less related properties. Overall, the model performs quite well with a training loss of ≈ 0.07 and a validation loss of ≈ 0.32.

3.3 Divided ANN Model

Based on the figures above, we find that the model which has three primary networks and a final network (trained with a batch size of 32, 250 epochs for submodels, and 125 epochs for the final model) offers noticeably improved predictive performance across multiple output types. Figure 6, which compares predicted and true values for the single ANN model illustrates how the model can be effective in both regression and classification tasks.

This model architecture demonstrates particularly strong performance in well-populated outputs such as 'Calculated Density', 'Yield Strength', 'Ultimate Tensile Strength', and 'Calculated Young Modulus', where predicted values closely follow the diagonal and indicate minimal deviation from true measurements. Additionally, classification accuracy is enhanced, with confusion matrices for structural phase prediction (e.g., B2', BCC', FCC', HCP') showing high true positive and true negative counts, and low incidence of false classifications.

However, similar to other models, prediction accuracy deteriorates somewhat in less data-rich or more complex outputs such as 'Elongation', 'Elongation Plastic', and 'Grain Size', where a wider spread from the diagonal can be observed. Despite this, the combined network model clearly outperforms the simpler single-network approach shown in the first figure, due to its modular architecture and improved capacity for capturing complex feature interactions.

References

[1] M. V. Kamal, S. Ragunath, M. Hema Sagar Reddy, N. Radhika, and B. Saleh, "Recent advancements in lightweight high entropy alloys – a comprehensive review," en, *Int. J. Lightweight Mater. Manuf.*, vol. 7, no. 5, pp. 699–720, Sep. 2024.

[2] R. Machaka, G. T. Motsi, L. M. Raganya, P. M. Radingoana, and S. Chikosha, "Machine learning-based prediction of phases in high-entropy alloys: A data article," en, *Data Brief*, vol. 38, no. 107346, p. 107 346, Oct. 2021.

[3] Y. Lu et al., "Directly cast bulk eutectic and near-eutectic high entropy alloys with balanced strength and ductility in a wide temperature range," en, *Acta Mater.*, vol. 124, pp. 143–150, Feb. 2017.