Predicting Pokémon Card Prices Based on Previous Sets

Zachary Lightcap
ztl1776@rit.edu

***Abstract*—The popularity of Pokémon has exploded over the last few years, and prices have followed suit. Although not every card is equally weighted, some have shot up in price, while many have not increased at all. We want to understand what features of a Pokémon card make it more expensive than others while also providing a way to generate base prices before the market settles on its preferred price. This paper presents a model that forecasts Pokémon card prices through a combination of structured data and unstructured image data. We will complete this by utilizing a multi-modal neural network architecture where a convolutional neural network (CNN) and an artificial neural network (ANN) are equally weighted. Then we will fit the model to the current average prices of cards on the market. Pokémon has been around a long time; therefore, some of the earliest cards will be significantly more expensive than later cards. Hence, age will be an important feature to monitor for this model, given that we want to predict the newest sets. Through this proposed architecture, we can train the network on previous Pokémon sets and attempt to predict the prices of the latest Pokémon card set.**

I. Introduction

The rise in Pokémon card prices in recent years has been widely observed, with some sources describing the general rise as being from a rise in disposable income, demand for mint cards, and a rise in high-profile traders. For individual cards, supply and demand become a heavy factor as people desire the rarest and most popular cards [1]. However, it is important to understand that these cards are used by a multitude of different people, whether they be collectors, competitive players, or just children who open packs for fun. Therefore, the prices represent a wide range of ideals as well as the features described above.

When a new Pokémon card set is released, prices fluctuate before stabilizing at values determined by consumer demand. This project will analyze the features of the past sets to understand their effect on price and build a model to predict the initial price of future sets. Such predictions can help buyers and sellers estimate fair starting prices and reduce exposure to early market volatility. Forecasting Pokémon card prices promotes fairness in the market and potentially removes a sense of missing out on potential savings or profit, depending on the early market volatility. This also provides insight as to why certain cards may become more expensive than others. Beyond this application, the broader significance lies in demonstrating the power of multi-modal neural networks. They are flexible deep-learning models that are capable of accurate predictions without requiring large amounts of computational resources. Using these flexible models, [2] has been able to create a multi-modal neural network to classify different human activities by looking at time-series sensor signals as their structured data and image data as their unstructured data. While our use case isn't the same, the combination of image and structured data allows for an increase in informed decisions.

Throughout this paper, we will explore previous work done in this field, the building of the three separate neural networks, the results of these networks in predicting the average price, and future work that can be done to explore and improve results. For prior work, we will focus more on multi-modal neural networks used in other fields, since not many sources have utilized them for Pokémon cards. Then, we will walk through the dataset as well as the CNN, the ANN, and the combined model's architecture. Afterwards, we will explore the results by reviewing evaluation metrics such as the mean absolute error, the median absolute error, and the R-squared of each model. This will also include a case-by-case analysis to see how the model performs on certain cards and why this may be the case. Finally, we will conclude our models and describe future work, as well as the potential limitations of this experiment.

II. Prior Work

Price prediction has been a long-sought-after goal as deep learning and machine learning processes have evolved. Although many publications focus more on price prediction of the stock market [3] [4], this can also be used to predict other forms of fluctuating prices. However, these regression and time-series forecasting models do not utilize image data as well as structured data. On the other side, sources have used just images of Pokémon to create CNN classification models [5]. Therefore, we explore other publications that utilize multi-modal deep learning neural networks, which combine both input types for other tasks. For one, [6] has successfully created a model to help with classifying multiple types of liver cancer utilizing computed tomography (CT) scans and other possible test data available, such as blood tests and biopsies. Now, [7] has successfully created a Pokémon card price prediction model where they employed a DenseNet-121 transfer learning model and a CNN implementation to get predictions with a low amount of test error (about $0.25 and $0.23) for each model, respectively. When reviewing previous multi-modal neural networks, [8] utilized the same technologies we will be using. However, they used structured,

image, and text data to predict housing prices, while we will be simplifying our model to just utilize image and structured data. We believe that with the current and updated datasets, we can create a model that can predict card prices to a serviceable range for new cards that have just hit the market.

III. Methodology

A. Data Exploration

This project revolves around pulling a dataset from the Pokémon TCG GitHub that has data about the card and the included images associated with each card [9]. From this dataset, we pull important data points, as well as add any data points that will help improve the mean-squared error of the model. The most important data point is the average price of the card, which is also included in this data set.

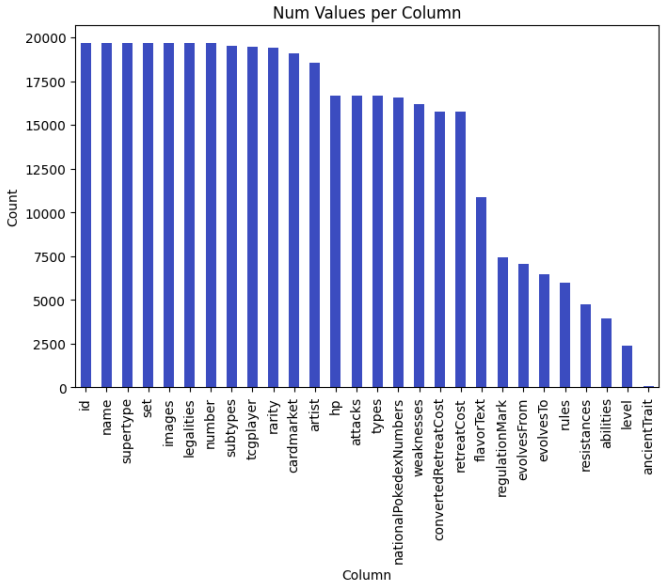

Fig. 1. Counts of Non-Null values for each column

Within the dataset, we see that the columns provided from the Pokémon TCG GitHub give us all of the necessary data. However, the format this data comes in is not entirely ideal. Pre-processing and feature engineering will be crucial to ensure our model is able to interpret the minimal differences between many cards. In Figure 1, we note that we have many columns that are not equal in size, with many null values included. Further exploration of the dataset reveals that there is a lot of textual information that needs to be removed or adapted to fit into a structured format. This includes rarity levels, special rules, and special traits. It is also important to note that we will be using the tcgplayer column to pull the average price from, which also has some null values. Other null values, such as hp, attacks, and types, all make sense, though, as cards can be split between being a Pokémon, a trainer, or an energy.

Examples of the cards are shown in Figure 2. Here, we can see immediate differences in how Pokémon cards are

Fig. 2. Normal Rare Zarude (left) versus Ultra Rare Rare Candy (right)

classified. First, we have a Rare Zarude card on the left, followed by an Ultra Rare Rare Candy card on the right. At this rarity, the images begin to take up more space, they are harder to get, and usually are rarer. Hence, this is what the convolutional neural network will have to make predictions based on. There are varying rarity types depending on the set, but there is an equivalence across sets, so they can be grouped into rarity buckets. There are also promo cards included, which look very similar to other cards in the set, but with slight changes, such as the rarity symbol. Here we are allowing them to stay to see if the combined structure will be able to understand this, since just the convolutional neural network may struggle to see a difference. Finally, we should mention that the dataset's prices are variable and can change from day to day, so all of the data comes from a single day to make sure they are as closely aligned as possible.

B. Pre-processing

The actual data was fairly clean and required only a small amount of pre-processing. Firstly, we needed to fix some of the card numbering. For special sets or promos, their order in the set's number would be something like "H1, H2, ..." instead of "1, 2, ...". To do this, we just removed all characters that weren't numbers and manually numbered a few labels, such as a set that was labeled from A-Z instead. Afterwards, we were able to convert the entire "number" column to an integer. Some columns needed to be split by using a Multi-Label Binarizer or through One Hot Encoding. The choice between the two depends on whether an instance can have 1 or more types of the features that are being split. An example of the Bianizer came from the Pokémon types as they can be one or two of: Colorless, Darkness, Dragon, Fairy, Fighting, Fire, Grass, Lightning, Metal, and Psychic 'Water'. However, the supertypes were done through the One Hot Encoder, since a card can only be one of the following: Pokemon, Trainer, and Energy.

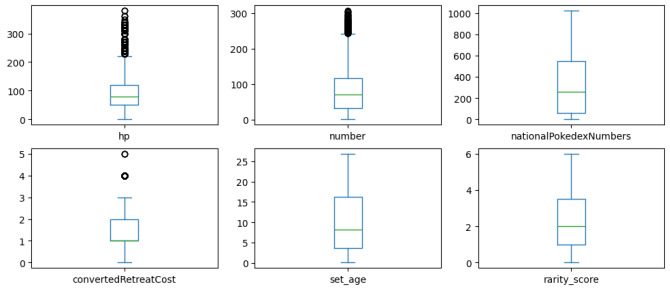

Fig. 3. Boxplots of Critical Features

Another thing we looked at was the boxplots of critical values (Figure 3). Here, we saw a large variation in terms of the spread of the values. This, along with the fact that we have added many binary indicators, pushes us to standardize the values. Therefore, all critical values were standardized between 0 and 1. We did look at outliers; however, it was decided not to remove them from the dataset as they were valid entries. Most card prices are just cents and maybe even dollars, but each set includes "chase" cards, which are rarer and thus can be up into the tens, hundreds, and sometimes thousands in terms of market price. We desire for this model to be able to predict all card types- even those of higher value, so the model needs to learn on these, so it does not always undershoot the price. Without this, the model may underperform significantly on anything over a few dollars.

Finally, we needed to pull out the image data. This included indexing the dataframe, pulling out the image data paths to online images, and downloading these images to create a new, separate dataset. Afterwards, the filepaths of each image were saved to the dataframe. All dataframes were stored both locally and on the drive as comma-delimited file types for future use and easy updates. Note that any features that did not have a market price were dropped as they couldn't be analyzed, and any value that did not have a clean image to use (only 4 of them) were also dropped. Before training the models, image data was resized to be (224x224) for better training on the EfficientNetV2B0 convolutional neural network. The pixel data was also rescaled to be between zero and one.

C. Feature Engineering

A lot of the data came in a way that was easier for cleaning, but harder for feature engineering. This is because there are a lot of variables that are stored as lists and nested into different sections. An example of this is the market price, which needed to be extracted from the tcgplayer column. Not only that, but cards can come in different versions. For almost every normal card, there is a reverse foil of that card, which is more expensive and has a shine to it. However, this shine cannot be seen through the image data. Therefore, only the normal version of cards was pulled if possible. Otherwise, the market price was marked as Null and dropped. Also, as seen before, some variables are in some sets but not others. Therefore, a large chunk of columns were turned into flags such as "has_rule", "is_regulation", and "has_ability". It was also important to create numeric columns for the textual data to include the information in the final dataset. Therefore, we counted the number of weaknesses and resistances a Pokémon had. Furthermore, we calculated the age of the set in years from the date the data was pulled.

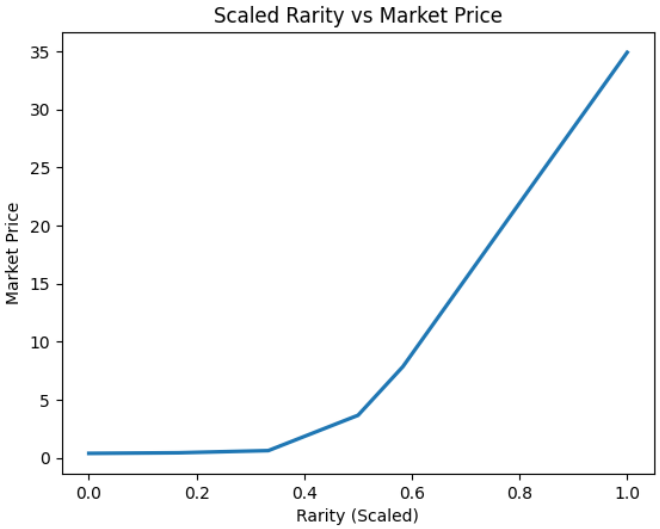

Fig. 4. Line plot of Scaled Rarity vs Market Price (excluding top 1%)

While the price of a card can't be predicted based on rarity alone, the rarity of a card is one of the most important features of a Pokémon card since the rarer a card is, the more expensive that card is (Figure 4). However, each set has its own version of a special card. For example, an ultra-rare card in one set could be called a "Rare Holo EX", and in another set the ultra-rare variant could be called a "Rare Holo VMAX". A mapping was created to map the rarities to other common rarity types. Then a new mapping was applied to assign a numeric value in order from least rare to most rare (1 to 6). The rarity mappings were chosen based on current knowledge of pull rates and groupings by the Pokémon company. Also, here we assigned promo cards and any other weird cards that were left out, a rarity of 3.5. This rarity level is the mean of all the rarities created and will ensure that they are included, but aren't over- or undervalued in training. Lastly, it was important to make sure promos were more distinguishable, so an "is_promo" flag was created.

D. Feature Selection

Before model training, a log transform was applied to the market price to counteract outliers and improve model training. After testing, the inverse log was applied to better calculate descriptive statistics such as the mean absolute error (MAE) and root mean square error (RMSE). Also, before testing, the dataset was split between training, validation, and a testing set. The testing set is the most recently released set that was released at the time called *Mega Evolutions*. The training and validation sets were compiled on the indices at random, with

feature	count	mean	std	min	25%	50%	75%	max
hp	19382	0.242424	0.185073	0	0.131579	0.210526	0.315789	1
convertedRetreatCost	19382	0.273842	0.206404	0	0.2	0.2	0.4	1
number	19382	0.263403	0.19814	0	0.101307	0.22549	0.375817	1
nationalPokedexNumbers	19382	0.321653	0.285186	0	0.059512	0.251707	0.538537	1
has_rule	19382	0.303684	0.45986	0	0	0	1	1
has_ability	19382	0.200031	0.400034	0	0	0	0	1
has_ancient_trait	19382	0.003044	0.05509	0	0	0	0	1
is_regulation	19382	0.382571	0.486027	0	0	0	1	1
weakness_count	19382	0.825766	0.384186	0	1	1	1	2
resistances_count	19382	0.244815	0.436302	0	0	0	0	2
num_attacks	19382	1.382675	0.761297	0	1	2	2	4
Colorless	19382	0.107419	0.309654	0	0	0	0	1
Darkness	19382	0.061242	0.239781	0	0	0	0	1
Dragon	19382	0.026004	0.15915	0	0	0	0	1
Fairy	19382	0.011712	0.107589	0	0	0	0	1
Fighting	19382	0.093386	0.29098	0	0	0	0	1
Fire	19382	0.076721	0.266155	0	0	0	0	1
Grass	19382	0.119131	0.323951	0	0	0	0	1
Lightning	19382	0.074244	0.262175	0	0	0	0	1
Metal	19382	0.047518	0.21275	0	0	0	0	1
Psychic	19382	0.115107	0.319159	0	0	0	0	1
Water	19382	0.121195	0.326362	0	0	0	0	1
set_age	19382	0.376311	0.284058	0	0.124833	0.304807	0.602849	1
is_promo	19382	0.062068	0.241285	0	0	0	0	1
rarity_score	19382	0.422927	0.216909	0	0.166667	0.333333	0.583333	1
marketPrice	19382	14.217246	80.959512	0.02	0.18	0.67	4.54	6502.49

Fig. 5. Summary Statistics

an 80-20 split between the rest of the cards left. Image data and textual data were kept apart, but shared an index to ensure that each dataset was equivalent and no leakage between datasets occurred.

Later on, a TensorFlow dataset had to be created to combine the structured table data and the image data for the combined model only. This was because the RAM of the Google Colab session would run out when trying to convert so many images to numpy arrays to combine the datasets normally, which would cause the session to abruptly end.

Our final dataset, shown in Figure 5, includes the 25 selected structural features and the target feature (marketPrice). Any numerical columns that could be pulled were kept, such as the HP, retreat cost, number in the set, national Pokédex number, age, rarity score, and market price. Otherwise, the rest of the columns that were kept were either counts, such as how many attacks, weaknesses, or resistances a Pokémon has, or they were flags. This includes all of the types and whether a Pokémon has an ability, a special rule, an ancient trait, and whether it is a regulation card or not. Being in regulation means the card can be played in competition. Of course, it is also important that the ”is_promo” flag accounts for the fact that they look very similar to the original cards. Overall, features cover many different aspects of the Pokémon cards that may contribute to the price or uniqueness of a card. The uniqueness is important because with so many images being fed to the convolutional neural network, cards have the potential to blend with the model because their overall design features are similar. Then, when the models are combined, the artificial neural network can overcome any of the challenges the CNN may have faced.

As for the image data, they were paired with their structured counterparts through their ID, but stored as a separate testing array for ease of access during training, validation, and testing.

E. The Neural Networks

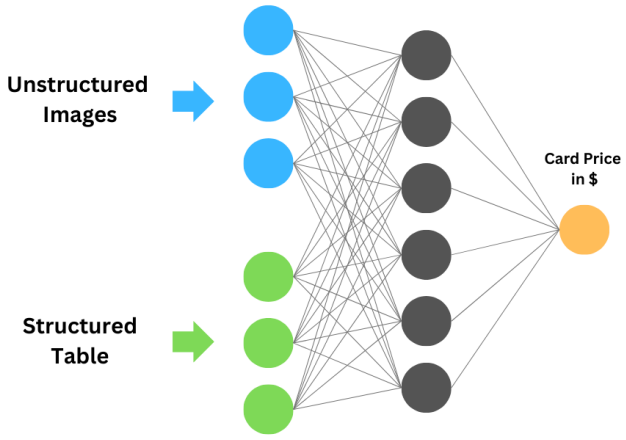

Fig. 6. Overview of the Multimodal Model

We will now present a high-level overview of the model’s architecture (Figure 6). We create two separate neural networks, one utilizing unstructured images and the other utilizing

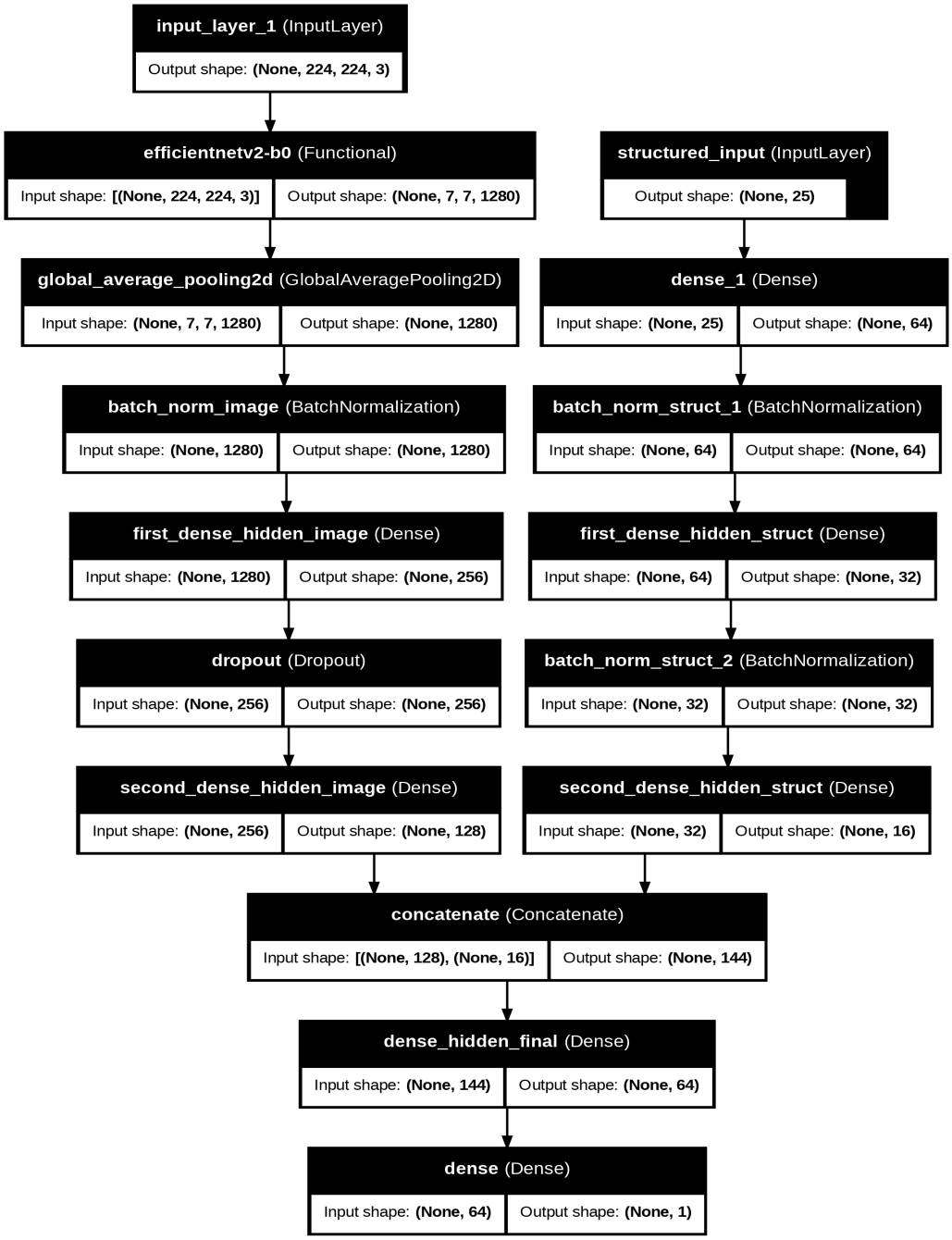

Fig. 7. Full model architecture

structured table data. The unstructured image data is sent through an EfficientNet convolutional neural network created by Google AI, which gives strong, accurate results while utilizing fewer layers and computations as compared with older models [10]. Therefore, it is perfect for our use case, given the faster turnaround on results. The structured data is sent through a multilayer perceptron (MLP) artificial neural network, which is a model architecture type where the nodes are fully connected with nonlinear activation functions. The last hidden layer before the output layer is saved from both of these models. Afterwards, these layers are concatenated and used as the input layer of the combined model. Then, the combined model is a series of dense hidden layers, learning from both models, to output the predicted card price in dollars.

To delve deeper, we have Figure 7, which reveals each layer of the neural network. The left side that leads to the concatenation layer describes the CNN, and the right side describes the ANN. For the convolutional neural network, we take our input of images, which were shrunk to (224x224), and send them through an EfficientNetv2-b0 model architecture.

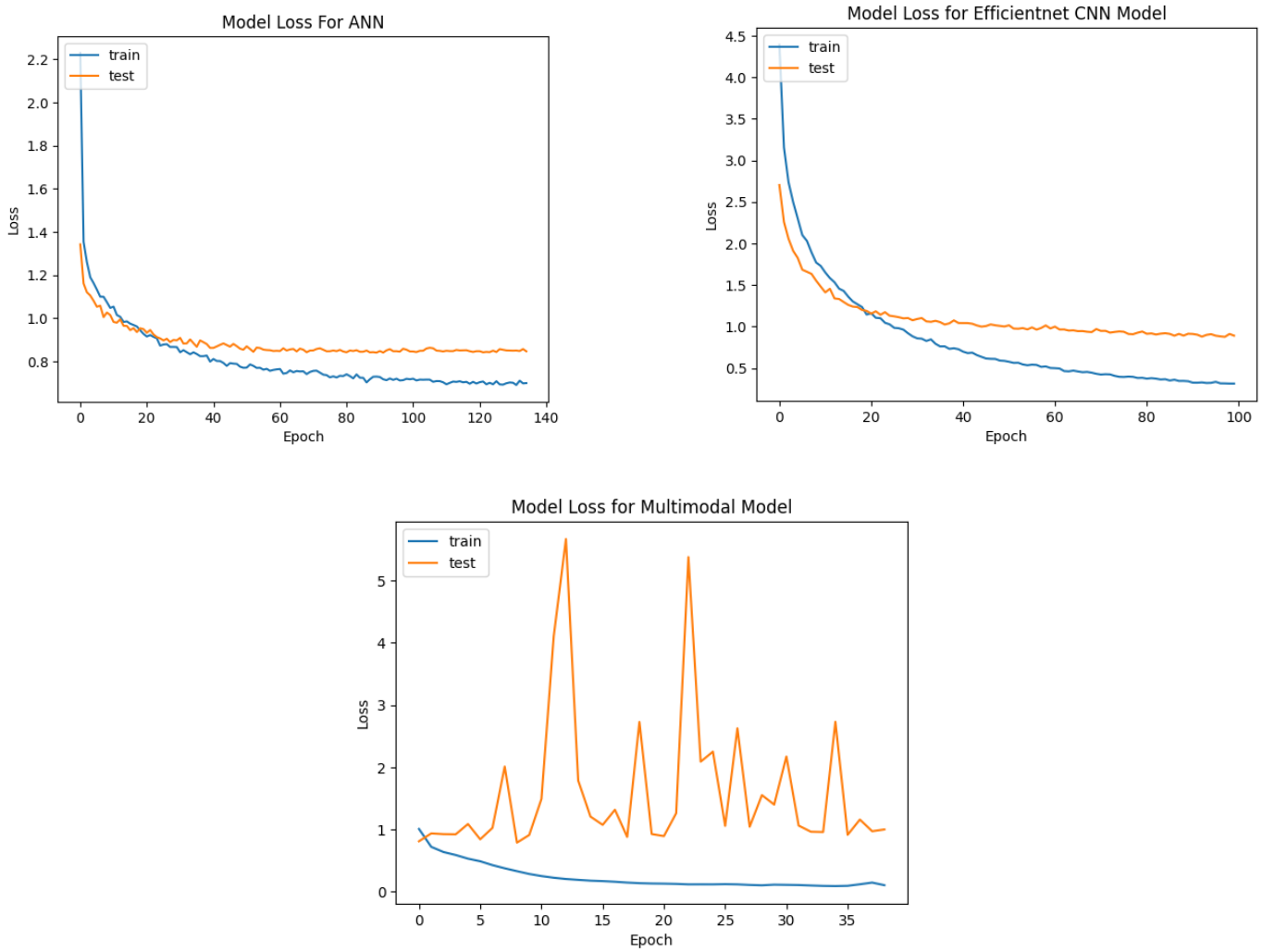

Fig. 8. Training and validation loss curves for the ANN, CNN, and Combined models

The rest of the structure includes structures that we added on to help improve model training and accuracy. This includes multiple dense hidden layers, a batch normalization layer to make learning based on the training data harder, and a dropout layer to prevent overfitting of lower-valued cards. The final hidden layer is a dense layer that reduces the dataset to 128 nodes. As for the artificial neural network, we use our 25-node structured input layer and add a dense hidden layer with 64 nodes. Then, we use alternating batch normalization and hidden layers to reduce the number of nodes to 16. Using batch normalization layers thereby improves training by normalizing the values and reduces overfitting on the large amount of lower-priced cards. We concatenate these two final hidden layers as the input of our combined model, resulting in a single 144-node layer. Then we train through a final dense hidden layer, reducing the number of nodes to 64. Lastly, our 64 nodes are reduced to the model's single output of the card's predicted price.

IV. Experiments and Results

During model training, we observe that the ANN and the CNN follow a similar loss structure (Figure 8). They both have an exponential downward curve where the loss of the training and testing lines flattens out. Early model stopping was based entirely on the validation loss, which means that even though the dataset was improving on the training set continuously, as it flattened out for the testing set, we found the model was not going to improve much further. Another observation of these two graphs is that the loss of testing flattened out much quicker than the loss of training. We must also note that the loss of the ANN versus the CNN is lower for testing, and higher for training. This indicates that the ANN generalizes better than the CNN. Although the ANN does not fit the training data as tightly (higher training loss), it performs better on unseen data (lower testing loss). The CNN, on the other hand, fits the training set more thoroughly but does not generalize as well, suggesting that it may be overfitting. As for the combined model, the loss didn't have a noticeable pattern. It appears to

have struggled to take meaningful steps towards improving the model from the first few steps for the testing loss, even though the training loss consistently took steps closer to 0. Thus, the combined model is also potentially overfitting the dataset and struggling to generalize.

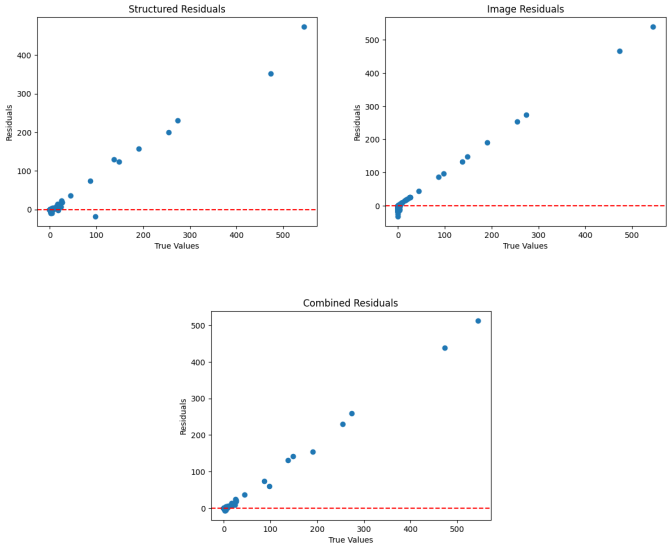

Fig. 9. Residual plots for the ANN, CNN, and Combined models

Our testing dataset consisted of 188 cards, all from the *Mega Evolutions* set. We use this type of set to simulate the goal of this model in a practical setting, which is to determine the price of a card upon launch to set a good starting price. Reviewing the residual plots (Figure 9), we notice the same linear pattern across all three models. This linear residual pattern indicates a systematic error, suggesting the models are underfitting and failing to capture a linear relationship present in the data. All three models are struggling to capture this relationship, but the structured model captures the relationship the best out of the three, as seen by its residuals being generally smaller than the image and combined residuals.

Another relationship revealed from our residual plots can be shown through our scatter plot of the predicted values versus the true values in Figure 9. That relationship, being at the models, does not predict very expensive cards very well. However, they do perform well at the cents and dollar levels, though the image model on its own does struggle at the lower level as well. Starting with the weakest model according to the graph, the image model's predictions are consistently off. For cards that cost cents, it predicted dollars in some cases, and even went up to 32 dollars for a card that was only worth 2 cents. For comparison, the structured model predicted 7 cents, and the combined model predicted 8 cents. This indicates to me that the card has features that resemble a higher-selling card, but not enough to warrant a $32 prediction. However, the image model was the worst at predicting higher-valued cards. For the $500 card, the image model only predicted $4.50 while the combined model predicted about $31, and the structured model predicted at least $71. Therefore, while all of the models

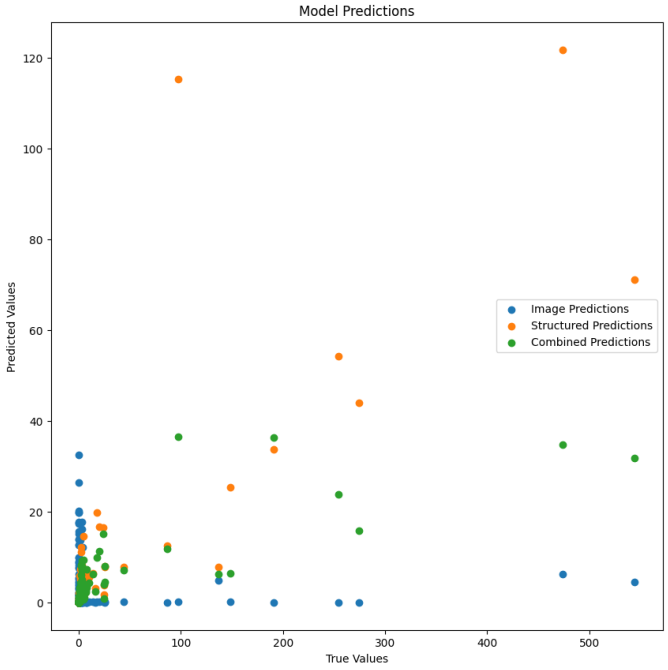

Fig. 10. Scatter plot of predicted values versus the true values

struggled as the prices grew to be in the hundreds, the image model would predict even higher on super cheap cards and lower on the high-end cards. We can also see, based on the scatter plot, that the image model usually predicted the lowest on high-valued cards, followed by the combined model, then the structured model, which was typically the closest to the actual price. While the models struggled to predict a price close to the heights an actual high-valued card was worth, the combined and structured models were still effective in predicting which cards would be more expensive.

MODEL	R2	RMSE	MAE	MAPE	MedAE
IMAGE	-0.04	63.44	16.09	54.71	1.52
STRUCTURED	0.30	51.96	10.85	0.92	0.05
COMBINED	0.10	58.82	12.18	0.91	0.05
MEAN	-0.05	63.67	13.88	0.68	0.06

Fig. 11. Statistics about the performance of all three models

The structured model performed the best out of all three models. We note that it had an R-squared value of .3, which indicates that 30% of the variation in the model can be explained by the structured model (Figure 11). This isn't great, but reviewing the scatter plot from Figure 10 again, we note that it was also good at predicting lower card values while being the closest to the highest-valued cards. The mean absolute error of this model was 10.85, which means that on average, the structured model was off by about $10.85. However, due to the large outliers, the median absolute error (MedAE) is an important statistic for comparison. This is because it shows that half the predictions are closer than 5 cents, and half are further than 5 cents, which, in terms of card prices, is a strong prediction. Being closer to 5 cents

off most of the time would rarely cause issues when setting your price, but it does further enforce the model's struggles predicting outliers. The combined model has similar values, but is more off since the image portion is realistically dragging it down. The combined model has an R-squared of .10, a mean absolute error of 12.18, and a median square error of .05 as well. This means the combined model is performing worse overall against the structured model, but when looking at smaller card prices, it has equivalent results. Although having only 10% of the model's variation explained by the model is less than ideal. The image model is hindering the combined model than helping it. This can be seen by the combined model being a little worse than the structured model. Then, looking at the image model's R-squared of -.04, mean absolute error of 16.09, and median absolute error of 1.52, it is the only model that is performing worse than just guessing the mean most of the time. However, only being off by an average of $16.09 and a median of $1.52 is still valuable.

V. Conclusions and Future Work

The structured model performed the best with fairly equivalent results from the combined model. It is clear that since the combined model is taking into account the image and the structural model equally, it is performing worse since the image model is performing poorly. Even if the weights of which one was weighted more were changed, the fact that the image model is performing worse among both high and low-valued cards indicates that future work should focus mostly on that first. Even if the structured and combined model struggled on high-valued cards, they are still much better indicators than just guessing a price at random or attempting to go by just the mean. Another thing to note is that card prices fluctuate constantly outside of the common, uncommon, and rare cards, and Pokémon card prices for "chase" cards are constantly on the rise and pulling further away from the price of an average card. This is caused by the rising rate of scalpers, the increasing popularity of cards, and an increase in willingness to pay for these cards. In fact, just recently, there has been an increase in the price of a normal 8-cent card up to $45, just because a person was trying to collect as many of them as possible, so people raised the price to get more out of it. Therefore, in the dataset that was created, unless the card was a part of the first few sets, the "chase" cards in the middle sets are less expensive. Thus, the dataset will struggle to account for these exploding prices. However, age was factored into the structured dataset, which may have had more of an impact on that model's performance and why it did the best. Currently, the best use of these models would be to pick the best one, in this case, structured, then utilize this to set your less expensive Pokémon card prices. Then, analyze which cards have larger spikes in the model and analyze those cards on the market to adjust the card price effectively, since the model was able to accurately predict which would likely be considered "chase" cards.

Overall, each of the models' performance was lacking, but I do believe utilizing data balancing techniques such as SMOTE (Synthetic Minority Oversampling Technique) or undersampling parts of the dataset would likely improve the model's performance. In addition, increasing the resolution of the images may allow the CNN to capture more subtle visual differences between cards. Future work could also experiment with incorporating textual card information; while text alone may not be sufficient, its nuanced details may enhance the fully combined model. Nevertheless, the most impactful next steps are likely refining the dataset across different time periods and improving the image-based model.

References

[1] D. Hall, "The Economics of Pokémon Card Pricing." The Local Economist. https://www.thelocaleconomist.com/articles/the-economics-of-pokemon-card-pricing [Accessed Sept. 19, 2025].

[2] J. Yun, S. Ha, and S. Choi, "Multi-Modal Convolutional Neural Networks for Activity Recognition." *2015 IEEE International Conference on Systems, Man, and Cybernetics*. https://ieeexplore.ieee.org/abstract/document/7379657 [Accessed Sept. 19, 2025].

[3] X. Ding, Y. Zhang, T.Liu, J. Duan, "Deep Learning for Event-Driven Stock." AMC Digital Library. https://dl.acm.org/doi/10.5555/2832415.2832572 [Accessed Sept. 20, 2025].

[4] R. Akita, A. Yoshihara, T. Matsubara, K. Uehara, "Deep learning for stock prediction using numerical and textual information." IEEE Xplore. https://ieeexplore.ieee.org/document/7550882 [Accessed Sept. 20, 2025].

[5] F. Campos, "Building a Pokémon Image Classifier using Deep Learning in PyTorch." Medium. https://medium.com/@filipesampaiocampos/building-a-pok%C3%A9mon-image-classifier-using-deep-learning-in-pytorch-bfa2ce385994 [Accessed Sept. 20, 2025].

[6] R. A. Khan, M. Fu, B. Burbridge, Y. Luo, F. X. Wu, "A multimodal deep neural network for multi-class liver cancer diagnosis." ScienceDirect. https://www.sciencedirect.com/science/article/abs/pii/S0893608023003246 [Accessed Sept. 19, 2025].

[7] E. Albarran and P. Keyes, "PokeNet: Predicting Pokemon Card Features Through Deep Learning." Stanford. https://cs230.stanford.edu/projects_spring_2018/reports/8290290.pdf [Accessed Sept. 20, 2025].

[8] D. Cote, "Hybrid (multimodal) neural network architecture: Combination of tabular, textual and image inputs to predict house prices." *Medium*, Nov. 18, 2022. https://medium.com/@dave.cote.msc/hybrid-multimodal-neural-network-architecture-combination-of-tabular-textual-and-image-inputs-7460a4f82a2ef. [Accessed November 19th, 2025].

[9] PokemonTCG, "pokemon-tcg-data," GitHub. https://github.com/PokemonTCG/pokemon-tcg-data. [Accessed Sep. 6, 2025]

[10] P. P., "What is EfficientNet? The Ultimate Guide." *roboflow*, Aug. 19, 2023. https://blog.roboflow.com/what-is-efficientnet/ [Accessed Dec. 1, 2025]