DS 4420

Final Project Write-Up

April 29, 2022

Predictive Power of Images for House Prices

Nick Bagley & William Conti

Abstract - **Common attributes of a home that are used to evaluate its price include square footage, number of bedrooms, and other numerical features. However, there is a visual factor involved in a home's price that is difficult to quantify. In this project we aim to transform images of houses into a format that can be used by artificial intelligence to predict prices. We then use several different models to determine if these images provide meaningful information for predicting house prices. The image features are extracted using a Convolutional Neural Network, and then these features are passed through Feed Forward Neural Networks to generate a final prediction for a home's price. From our project we have found that images of a house can have a positive impact on a model's predictive power of the house's price.**

I. INTRODUCTION

Accurately predicting the price of a home is something that can benefit buyers, sellers, investors, and developers. There are many different companies such as Zillow and Trulia that build complex predictive models in order to evaluate home prices as accurately as possible. [1] Historically real estate prices have been evaluated using different attributes of a house such as square footage, distance to transportation, and number of bathrooms. These are all quantifiable features of a home that are easily passed into models. However, these factors are not perfect in predicting housing prices. There has always been a visual factor when evaluating a home as well, which has typically been done by an appraiser visiting a property and subjectively measuring the attractiveness of a home. With the modern ability of deep learning models and the increased access to housing data, the visual aspects of a home can now be evaluated by models. [2] Our goal for this project is to see how much predictive power can be added to housing price models by including images of the house in the feature set.

II. DATASET AND FEATURES

A. *Initial Dataset*

The dataset that we used was housing data from Southern California containing over 15,000 different houses. It was sourced from Kaggle, created by user ted8080. [3] Each row contained several numerical features, as well as a column that references a specific image file stored in a separate folder. Each row is associated with a single image file that shows one exterior image of the respective house. The columns of the dataset are shown in Figure 1 below, with the target variable highlighted in green.

Image_id	Street	City	N_city	Bed	Bath	Sqft	Price

Figure 1: Image_id references an image in another folder, and N_city encodes the City column

The dataset contained a good distribution of prices. Most of the houses had prices between 250,000 and 750,000, with frequency significantly declining past 1,000,000. The upper range of prices was around 2,000,000. While prices in this range were much less frequent, they were not solo outliers so there was no need to remove them from our dataset. The distribution of the housing prices are shown in Figure 2. Similarly, there were no real outliers found in any of the feature columns that needed to be removed. The different cities were also evenly represented across the dataset.

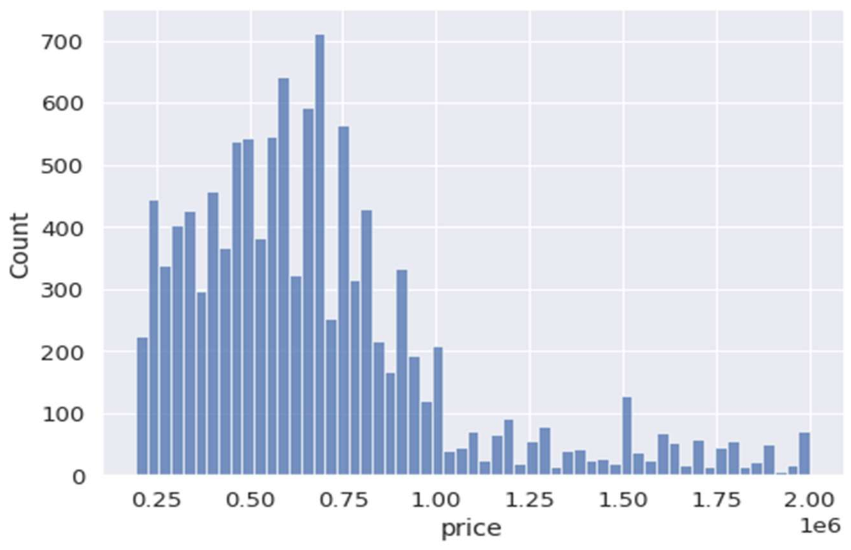

Figure 2: Price Distribution

B. Data Cleaning

The creator of the dataset also provided a list of the image files that were invalid. These included blurry images, duplicates, and incorrect image files. Figure 3 below shows an example of a valid image, and Figure 4 shows an example of an invalid image.

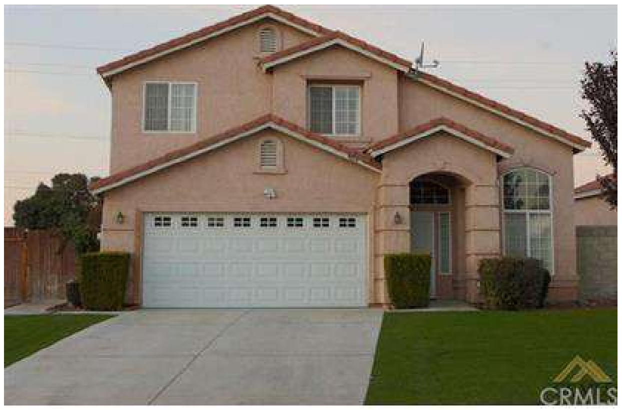

Figure 3: Valid Image

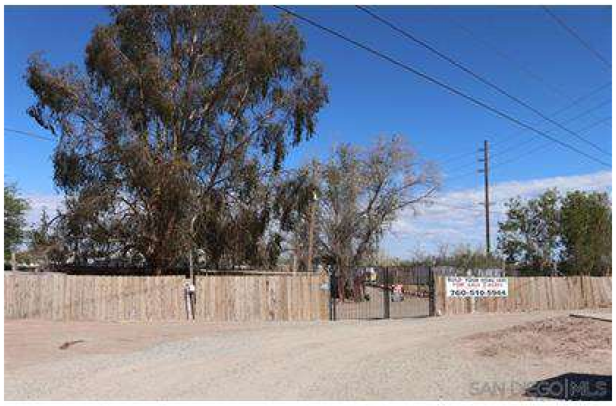

Figure 4: Invalid Image

We removed all the invalid images and their pertaining rows in the dataset. In addition to this, we removed all rows in which that row's city occurred less than 20 times in the dataset. This was to prevent the model from applying too heavy of a weight to the cities the underrepresented cities in the dataset. Once we applied this data cleaning we were left with 11,375 rows of data.

C. Final Features

Both the CITY and the N_CITY columns represented location. Since the encoded column was easier to work with than the textual column, we dropped the CITY column. We also dropped the STREET column since it was not able to provide any extra meaningful location information to our model. The CITY column had 347 unique values, which we one-hot encoded into 347 separate columns. This left us with 351 features in our final dataset consisting of BED, BATH, SQFT, IMAGE_ID, and the one-hot encoded N_CITY.

III. MODEL TRAINING

A. Baseline Model

First we created a baseline model for comparison. We removed the IMAGE_ID column from the dataset to measure the model performance without any image features. To create our neural networks for this project we used Pytorch. Our neural network has three linear layers and uses ReLu as its activation function. The input layer has 350 units, one for each feature. This layer then reduces this to 8 units which it passes to the hidden layer. The hidden layer reduces this to 4 which it passes to the output layer. Finally, the output layer converts this into a single value representing the house price. The structure of our network is visualized below in Figure 5.

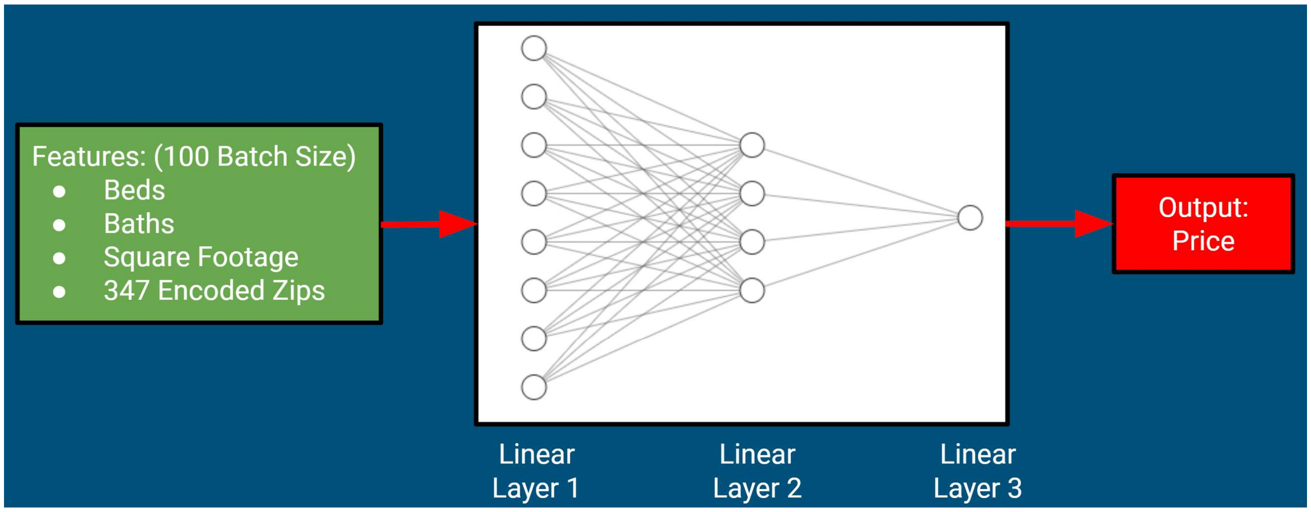

Figure 5: Baseline Model Structure

We tried running the model with different numbers of layers and nodes, but we found that this simple structure provided the best performance. More complex structures improved the results by negligible margins while increasing the cost of training the model.

To train our model we created a dataset class using Torch's Dataset class. We used a DataLoader in order to train our model in batches of 100. [4] We then ran our model through a training and validation loop using gradient descent and the Adam optimizer. We chose to use mean absolute error as our loss function to record across each epoch.

B. *Image Feature Extraction Using CNN*

To train our final model we first extracted feature vectors from our images. We used a pretrained Convolutional Neural Network called VGG16 for our images. This is a very popular architecture that is used mainly for different image classification tasks. The model uses a Softmax function to obtain its final output. However, since we are not using the model for classification, we instead freeze the network before this final layer and extract a vector of 4,096 features for each image. [5] The architecture of the VGG16 CNN is shown in Figure 6. Where we froze the model and extracted the feature vectors is indicated by the red arrow.

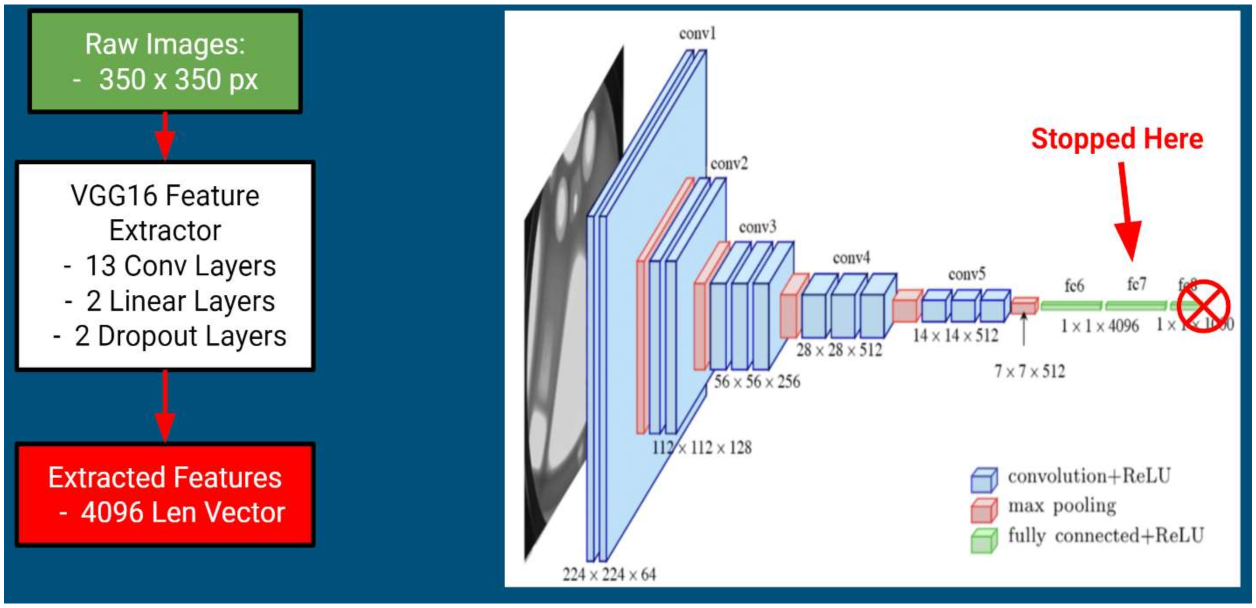

Figure 6: VGG16 Architecture for Feature Extraction

C. Final Image Model

Our final model used both the tabular features from the baseline model and the image features extracted from the Convolutional Neural Network. The image features were loaded from a CSV into a data frame. Then we merged the data frame containing our numerical features with the image data frame, using the IMAGE_ID column as a key. Once we had our final dataset we created another Dataset class, separating each row into its respective numerical and image features.

The model structure that we used ran separate sequential layers for the numerical and image features, using ReLu activation for both. For the numerical layers we started with an input layer of 346 features. This is because we dropped rows that did not have a corresponding image, which caused four cities to vanish from the dataset, ultimately lowering the number of one-hot encoded columns by 4. This gets reduced to 16 units and passed to our hidden layer. This layer reduces it to 8 units and passes it to our final linear layer, which reduces it down to 4 units.

For our image layers we start with our 4,096 image features that we extracted from the VGG16 model. This gets reduced to 32 units, which then gets passed into a batch normalization layer to standardize the inputs to the layers for the current batch. [6] We then pass this into a dropout layer to help mitigate overfitting and improve the generalization error of the model. [7] The model then fed into the next linear layer, taking the 32 units as input and outputting 16 units. This was passed to the final sequential layer which reduced this to 4 final output units.

Once the data passed through each of these sequences separately we were left with 4 units for the numerical data and 4 for the image data. We concatenated these into a single tensor of size 8. The model contains a final output layer, taking these 8 units as an input and outputting a single price value. Structuring our model like this yielded the best results, opposed to combining the numerical and image features at the beginning and feeding them through the neural network together. A visual representation of our model's structure is shown in Figure 7.

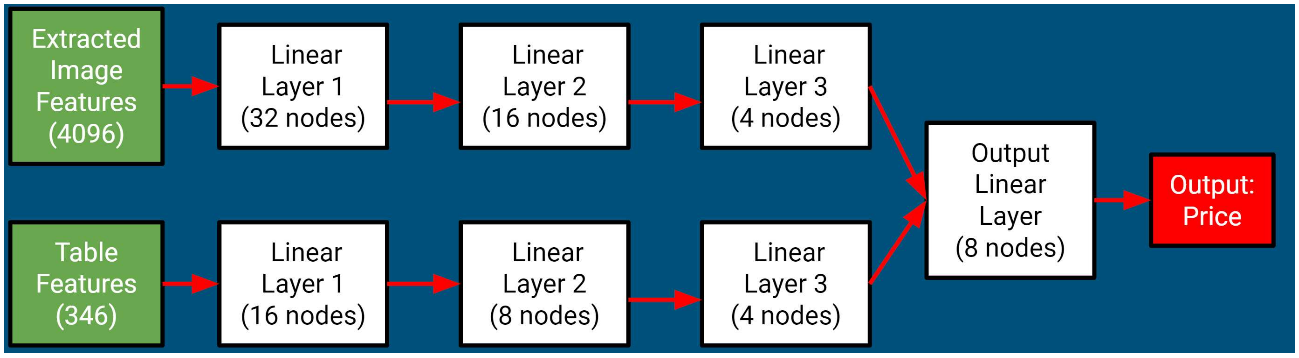
Extracted Image Features (4096)
Linear Layer 1 (32 nodes)
Linear Layer 2 (16 nodes)
Linear Layer 3 (4 nodes)
Table Features (346)
Linear Layer 1 (16 nodes)
Linear Layer 2 (8 nodes)
Linear Layer 3 (4 nodes)
Output Linear Layer (8 nodes)
Output: Price

Figure 7: Final Architecture for Image Neural Network

IV. RESULTS

A. Our Findings

For our evaluation metric we chose to use the mean absolute error (MAE). To get a baseline performance we started by finding the MAE of predicting the mean price of our dataset. Once we normalized our target variable, we got an MAE of 0.131 when only predicting the mean. If our model obtains a better value than this it shows that it has some predictive power and does not only predict the mean for each input.

We have three other models to evaluate. We have both our baseline and our image model, as well as another model in which we only used the image features without adding the numerical features back in. This image-only model is set up the same as our image-combination model, just without the numerical layers. The purpose of this model is to give a better insight into the raw predictive power of images. Shown in Figure 8 are the results of our models after running each for 20 epochs rounded to 3 digits.

Model	**Training MAE**	**Validation MAE**
Predicting the Mean	0.131	0.131
Baseline Numerical Model	0.050	0.053
Image + Numerical Model	0.045	0.052
Image Only Model	0.104	0.114

Figure 8: MAE of the Different Models

The base model already performs extremely well without any image features. The model avoids overfitting, performing very similarly across the training and the validation sets. Figure 9 shows the loss across both the training and the validation loops. The lines follow a similar curve, with validation only slightly starting to move above training towards the final epochs. Figure 10 is a chart that depicts the model's performance after 20 epochs in respect to its range of predictions. The predictions are very similar to the actual, with the underrepresented upper values being the spot where the model struggles most.

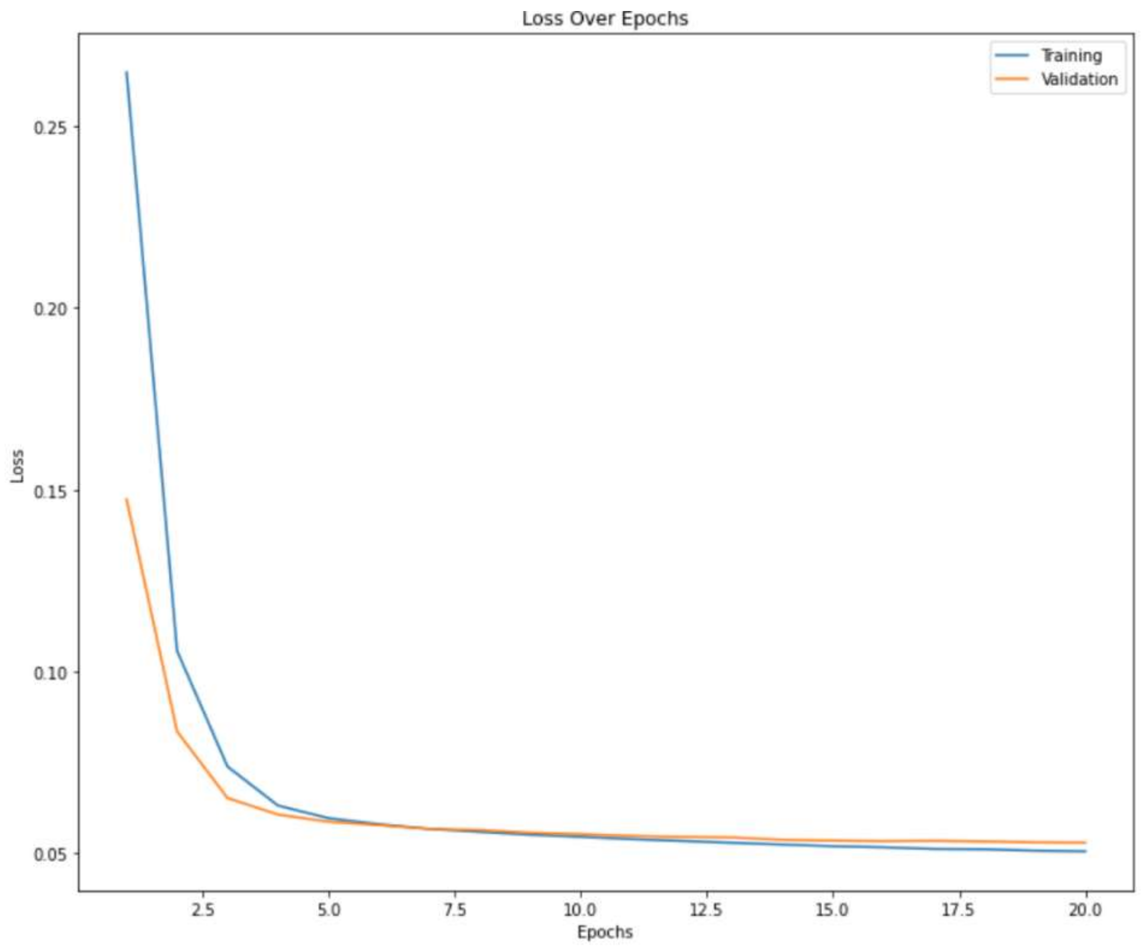

Figure 9: Loss for Baseline Numerical Model

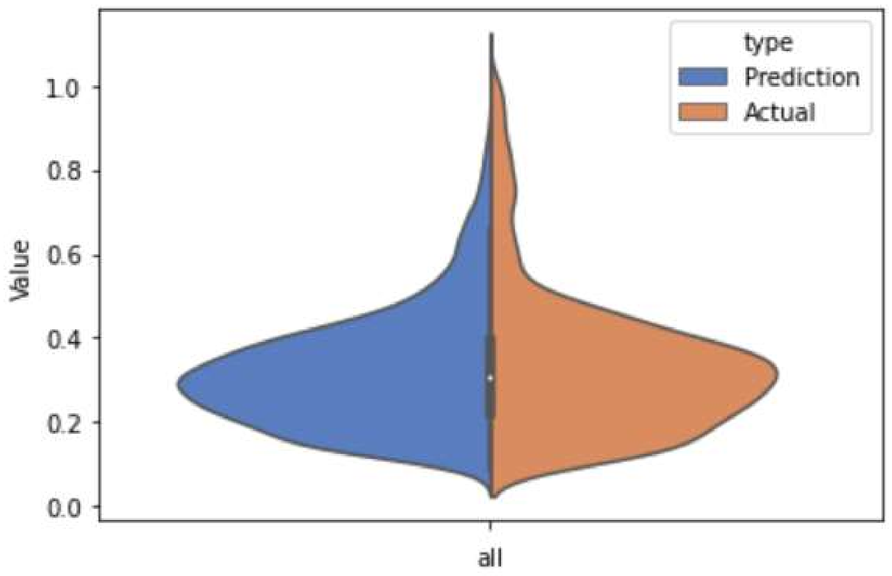

Figure 10: Prediction Chart for Baseline Numerical Model

The image-numerical combination model performs extremely similar to the baseline numerical model. Both the training and the testing MAE are below the baseline model, but only by marginal amounts. There was a greater improvement in the training loss, which indicates that the images cause more overfitting in the model. Figure 11 shows the training and validation loss plotted for each epoch. While the lines follow a similar curve as the baseline model, but the validation loss begins to rise above the training loss much earlier in training. Figure 12 is a chart that depicts the model's performance after 20 epochs in respect to its range of predictions. This chart looks very similar to the baseline model too. This model seems to be slightly more accurate in the range of 0.1 – 0.5, but still struggles in the underrepresented upper range of values.

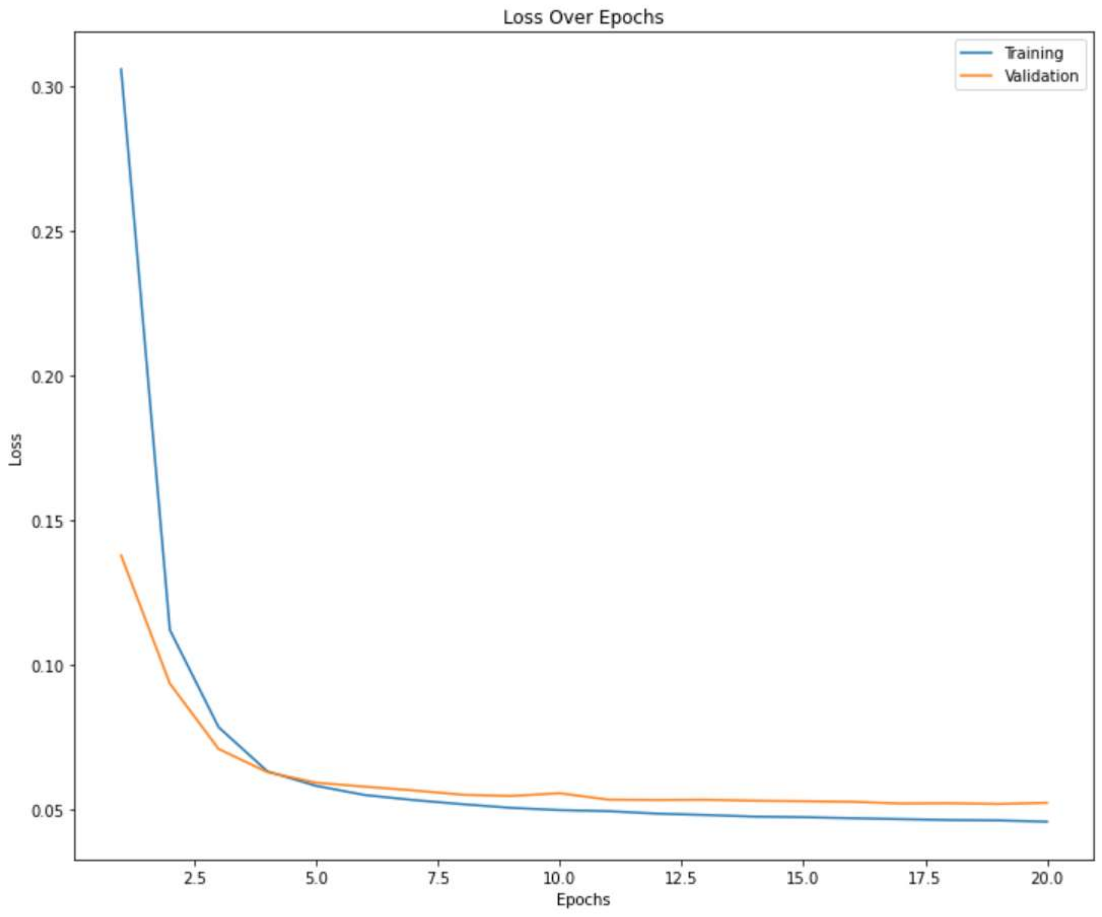

Figure 11: Loss for Image-Numerical Combination Model

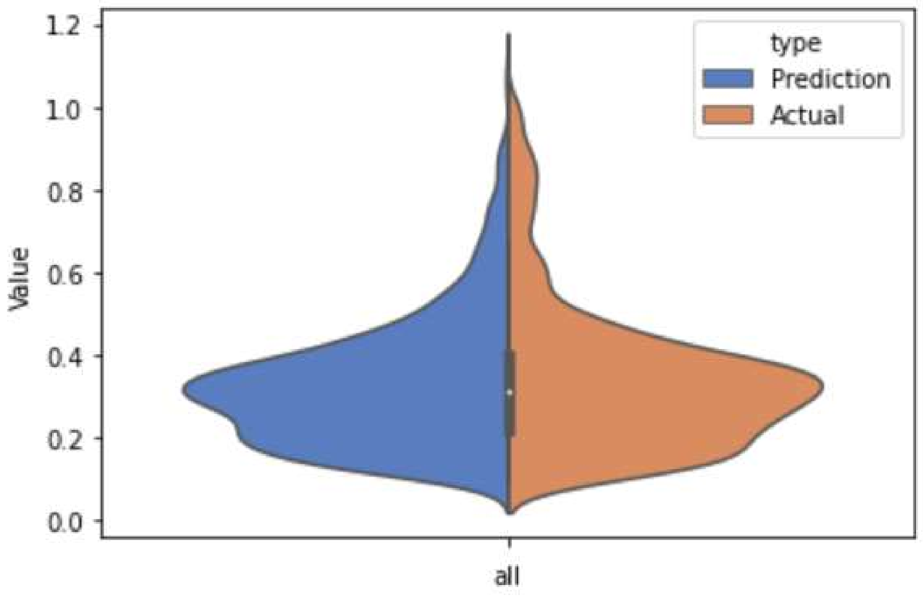

Figure 12: Prediction Chart for Image-Numerical Combination Model

The image-only model performs the worst out of our neural networks. However, it does still perform better than predicting the mean, which shows that the images do hold some predictive power on their own. This model also has the largest gap between the training and validation loss. The trend of this gap across the models indicates that the more a model focuses on images, the more the model overfits to the training data. Figure 13 Shows the training and validation losses plotted across each epoch. The training curve steadily decreases, but the validation curve is much more erratic. After 9 epochs the validation loss begins to increase as a result of model overfitting. The loss continues to rise and fall, eventually hitting a low of 0.112 after epoch 18 and then again spiking. Figure 14 is a chart that depicts the model's performance after 20 epochs in respect to its range of predictions. The almost-normal distribution of the predicted values suggest that this model relies much more on predicting the mean of the dataset than the other models. While the images do add some predictive power, the model is still not able to match the true range of price values.

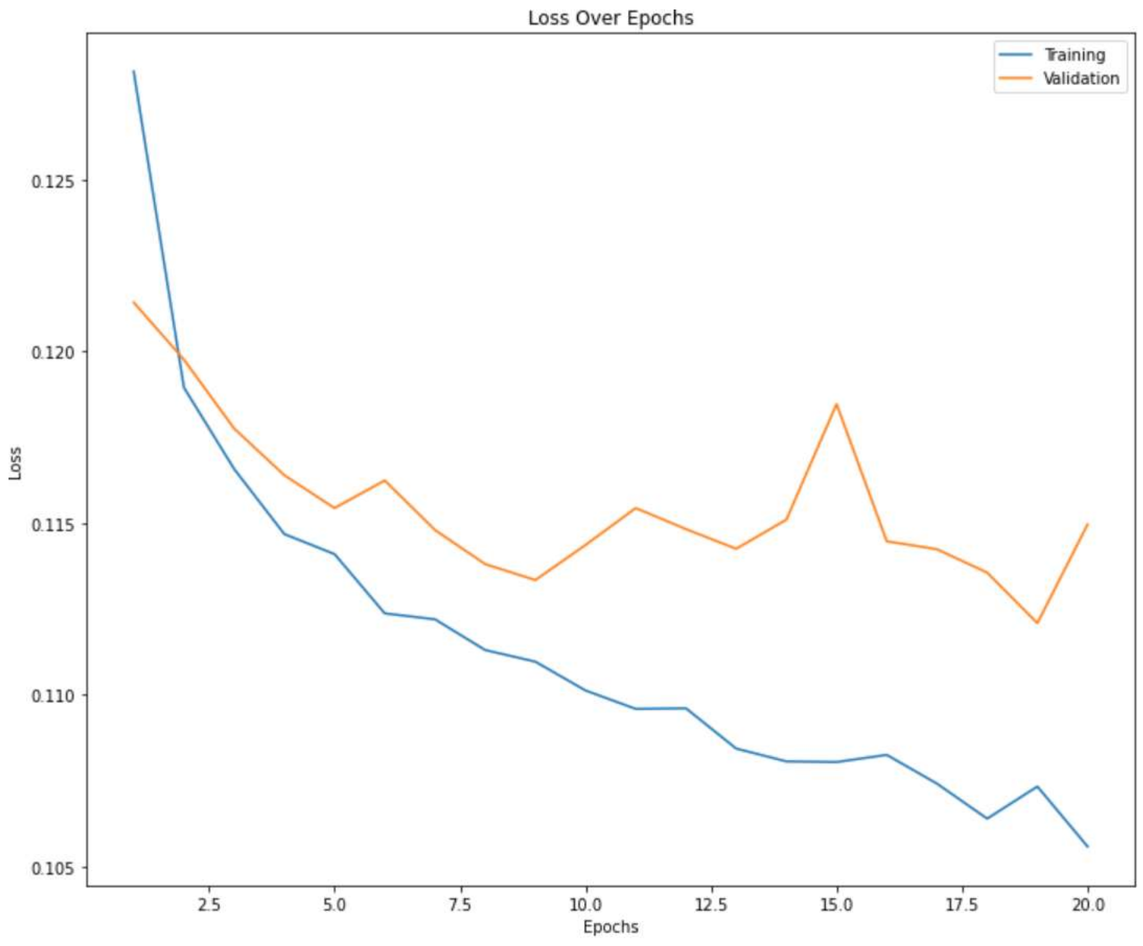

Figure 13: Loss for Image-Only Model

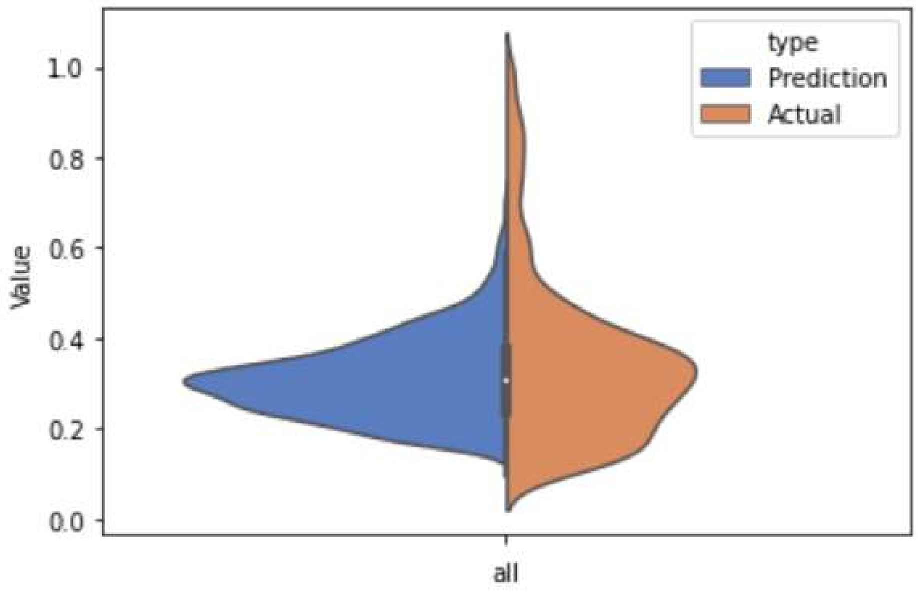

Figure 14: Prediction Chart for Image-Only Model

B. Previous Research

In addition to our own experiments, we also looked at previous research on this topic. Zona Kostic and Aleksandar Jevremovic also did research on predicting house prices using images. [8] The dataset that they used contained interior, exterior, and satellite images of every house, usually having between 15-30 pictures per row. Then, instead of extracting a feature vector from a single CNN, they passed these images through a pipeline of models to obtain more specific features from the images. For example, one model would extract an ‘attractiveness’ feature, while another would extract a ‘size’ feature. Once these features were obtained, they would combine them with the numerical features that they already had (beds, sqft, etc.) and pass it through a final model. Like us, they evaluated the MAE of the models before and after the image features were added to the inputs. Depending on the model they were using they saw an improvement in the MAE by between 0.01 and 0.05. From these results they concluded that images boost the predictive power of price models for real estate.

V. CONCLUSION

In this project we prove that using images is useful when predicting house prices. The model that combined images with numerical features outperformed the baseline numerical model, even though it was only by a very small amount. While there was slightly more overfitting when images were introduced, it did not result in a worse loss against the validation set. Additionally, the model trained with only the images performed noticeably better than the predicting the mean. This proves that a model is able to gain predictive insight from images to evaluate a house's price.

The main downside that we faced when using images was the cost of extracting the feature vectors. Running our images through the VGG16 CNN took nearly five hours. This increase in cost is the main trade-off for the improvement in performance. However, we were running our networks on very limited computing power. Large companies who are looking to build the most accurate price prediction models have access to much better computing power. Any increase that they can achieve in accuracy is worth the cost trade-off.

We did not obtain as significant results as seen in the research done by Kostic and Jevremovic. The main reason for this is likely the input data. In their experiment they had multiple exterior, interior, and satellite images for every row of data. This allows the model to gain a significant amount of information. In our project we only had access to a single exterior image for each house. In addition to not giving as much information, it also made the model much more sensitive to any abnormalities in the images. For example, if most of the images are taken from the front of the house, the model would perform poorly on a row in which the image was taken at a diagonal. When there are more images per house the abnormality found in a single image does not have as much effect. However, even using only one image per row we obtained results that showed the predictive power of images in house prices. To obtain the highest possible performance in a pricing model it is beneficial to combine images to the numerical dataset.

Citations

[1] Rose, J. (2022, March 7). *10 awesome websites who let you check your home's value for free*. Good Financial Cents®. Retrieved April 27, 2022, from https://www.goodfinancialcents.com/online-home-appraisal/

[2] Asaftei, G. M., Doshi, S., Means, J., & Sanghvi, A. (2021, March 30). *Getting ahead of the market: How big data is transforming real estate*. McKinsey & Company. Retrieved April 27, 2022, from https://www.mckinsey.com/industries/real-estate/our-insights/getting-ahead-of-the-market-how-big-data-is-transforming-real-estate

[3] ted8080. (2019, December 4). *House prices and images - socal*. Kaggle. Retrieved April 27, 2022, from https://www.kaggle.com/datasets/ted8080/house-prices-and-images-socal

[4] PyTorch. (2019). *Torch.utils.data*¶. torch.utils.data - PyTorch 1.11.0 documentation. Retrieved April 27, 2022, from https://pytorch.org/docs/stable/data.html

[5] Thakur, R. (2020, November 24). *Step by step VGG16 implementation in Keras for beginners*. Medium. Retrieved April 27, 2022, from https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

[6] Brownlee, J. (2019, December 3). *A gentle introduction to batch normalization for Deep Neural Networks*. Machine Learning Mastery. Retrieved April 27, 2022, from https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/

[7] Brownlee, J. (2019, August 6). *A gentle introduction to dropout for Regularizing Deep Neural Networks*. Machine Learning Mastery. Retrieved April 27, 2022, from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

[8] Zona Kostic, Aleksandar Jevremovic. 2021. *What image features boost housing market predictions?.* arXiv:2107.07148v1. Retrieved from https://arxiv.org/pdf/2107.07148.pdf