Real Estate Property Price Prediction Using Convolutional Neural Networks with a Pattern Recognition Approach

A novel approach to an ancient task

Evan Ting I Lu
SEAS,
Columbia University.
New York, NY, USA
tl3098@columbia.edu

Abstract

Real estate is one of the oldest and largest industries in the world. The US housing market grew by $6.9 Trillion in a single year in 2021[1]*. Such paramount economic importance makes it naturally one of the hottest sectors where people have strived to wrap their heads around market behaviors and predict property prices. This paper proposed a novel way for real estate property price assessment that incorporates feature engineering and convolutional neural networks (CNN). The contribution of this paper is two-fold. First, it verifies the concept of transforming the sales price prediction task into a pattern recognition problem using a proven feature engineering process and leveraging CNNs to produce effective results. Second, it explores adopting Super-Resolution Generative Adversarial Networks (ESRGAN) [1] to up-sample low-resolution images to improve model performance. The proposed method achieved 83.5% accuracy on the 5-class price prediction task and an RMSE of 13880 on the regression task, showing that such a novel approach is applicable and possesses much potential in the future.*

Keywords

real estate, housing market, investment, price prediction, price assessment, feature engineering, convolutional neural network, low-resolution image, pattern recognition, super-resolution

1. Introduction

Real estate, easily a juggernaut in the world's economy, has been a field where people yearn to understand and, as a more audacious goal, to make predictions. Not unlike the stock market, countless professionals have strived for decades for such aspirations. With more powerful tools thanks to the rise of machine learning and deep learning, avid practitioners have been trying to leverage these new aids to reshape the landscapes of these markets. Traditional machine learning methods have been applied to stock and real estate markets. Contrastingly, while much research has been done on leveraging deep learning for stock analysis, very little work has been done in the real estate sector. I consulted industry professionals and summarized possible reasons. First, the real estate industry is by and large old and slow-changing. The huge profit margin and high entering barrier create a moat for it to resist the disruption of new technologies. Most industry players either lack the required skills or the incentive to adopt new technologies. Second, compared to the stock market, the real estate market is more reflective and reactive to geographical factors. This causes uncertainties and challenges for quantitative analyses. Thus, despite the popularity of deep learning, relatively little research has been done in this lucrative domain.

Despite these difficulties, I believe progress comes from trial and error, and revolutions start with tiny sparks. The motivation for this project is not to generate state-of-the-art results but to explore a novel way for an ancient task and hopefully could be that first spark.

The key concept of this paper is to transform the typical property price assessment task into a pattern recognition problem, which modern convolutional neural networks excel at. Each transaction of a housing property was transformed into an 18 x 18 x 1 grayscale image. Subsequently, CNN models were trained on these images to learn to predict (or, in real estate terminology, assess) the property price given its sales variables. A crucial premise to the success of this strategy is the number and quality of features. Images applications using CNNs usually contain tens of thousands of pixels, if not more. It is an arduous, if not impossible, task to obtain that many features for datasets in real-world contexts. To overcome this challenge, this project references an existing feature engineering process to augment 79 raw features to 321 – a number that is nowhere near a "real" image but is good enough to verify the concept, which is the primary purpose of this paper.

The second part of this paper experiments with whether up-sampling images prior to training helps with model performance using a super-resolution GAN (ESRGAN) model. This approach came to interest as it was proven in prior work by Cai et al. [4], who delivered promising results in low-resolution image recognition tasks. However, lacking proper data to fine-tune the ESRGAN model, this project was limited to concept-verification instead of striving for optimal results.

2. Previous work

Pattern recognition has been a popular subdomain in machine learning and deep learning. Low-resolution image recognition, a branch in the pattern recognition domain, is also not an untouched realm, and there has been some decent prior work. For instance, Massoli et al. proposed a CNN-based system for low-resolution facial recognition [2]. Rutia et al. [3] focused on leveraging CNNs to conduct hierarchical identification using low-resolution images of insects.

Rachmatullah et al. introduced in their research using CNNs to classify low-resolution fish images, stating such a task has been a challenge in the marine domain because of (1) fish size and orientation variance, (2) feature variance, (3) environment variance, and (4) low image quality. They conducted a series of testing and developed a two-layer CNN model that achieved 99.73% accuracy on the Fish CLEF 2015 dataset. The proposed model incorporated dropout and data augmentation techniques. The authors suggest these two measures to be effective for small and low-resolution image applications. On the other hand, Cai et al. proposed using then-state-of-the-art super-resolution techniques to enhance image quality before training the models [4]. The images were cropped and down-sampled to 50 x 50 and then up-sampled to 227×227 with bicubic interpolation. Their proposed "resolution-aware CNN" model demonstrated a promising result for this approach on three commonly used datasets.

These previous studies, by and large, share two common traits. First, there's no clear-cut definition when speaking of "low-quality" or "low-resolution." In most work, a 100 x 100 image would be considered low-resolution, with some studies going further to as low as 700 pixels [11], while others have a looser definition, such as 128 x 128 [3]. Regardless of the definition, the "images" used in this paper are much smaller and, thus, of much lower "quality." It remains a question whether CNNs can maintain their reputation under such constraints. Second, all the previous works were built upon "real images." That is, the images' substructures were mainly guaranteed; the question left was how to effectively train the model to harvest and learn from the structures. On the other hand, the data used in this paper has no such guarantee, which places more uncertainty and challenges on the task.

While in the real estate domain, it is not new to adopt traditional machine learning or statistical methods such as correlation tests, SVM, decision trees, etc. for market analysis and property assessment tasks [5], it remains a largely untouched realm as to applying modern deep learning methodologies to these problems. Most works involve primitive network designs such as a simple one-layer perceptron or three-layer dense neural networks [6] and can't speak much about the full potential of deep learning. Contrastingly, stock market enthusiasts seem to embrace these new tools more. M. Leung introduced in his article [12] using feature engineering to transform time series stock data and train with CNN/GRUs. This shares a similarity with the idea of this paper, i.e., transform a piece of data via feature engineering and develop deep learning models with the generated "images."

In sum, while deep learning has been a hot realm in recent years and has proven itself to be a potential revolutionizer for many domains and applications, real estate, with its notorious slow technology adaptation pace, leaves much virgin land and hence opportunities to be explored.

3. Method

In this section, I first describe the data used in this project. Then introduce the feature engineering process, which is not invented by me but is a crucial cornerstone to the success of the main concept. Then, I describe several CNN architectures that I experimented with. Finally, I explain how I applied the super-resolution model to enhance image quality in the hope of boosting model performance.

3.1. Data

3.1.1. Source

The data was obtained from the Kaggle competition: *Housing Prices Competition for Kaggle Learn Users* [7]. Training and testing datasets were provided in .csv format, each containing 1460 and 1459 samples, respectively. Data is preprocessed and cleaned. Each data entry has 79 variables describing a house sale in Ames, Iowa. The goal is to predict the sales price for each home sale. Despite its relatively small size, this dataset offers a great starting point for this project. It's a well-test and proven dataset with more than 39,000 teams competing over eight years; top performers deliver consistent and robust results, which shows there may be underlying structures among the samples. Some top-ranking teams are kind enough to share their methodologies publicly, including the feature engineering process, which is the critical assumption for my concept to work. The lack of scale in this dataset is a limiting factor for this paper, but as the main objective was to verify my idea, this dataset provides a decent place to start from.

3.1.2. Train/test split

I combined the original train and test datasets and re-split them into two portions with a ratio of $85\% : 15\%$ for the actual train and test dataset for this project. The samples were randomly shuffled before splitting to mitigate potential bias in the original train/test datasets. After splitting, 2475 samples were used in training, and 436 were reserved for final testing.

3.2. Feature Engineering

For years, convolutional neural networks have proven themselves in the pattern recognition domain. They are known to be experts in picking up abstract structures in images. For this to work, however, there's an underlying assumption that there are enough features for the network to learn from. In terms of images, this means the number of pixels. A 224 x 224 image, as in ImageNet, has more than 50,000 pixels; a 32 x 32 image in CIFAR-10 has 1024 pixels; even a 28 x 28 MNIST image has 784 pixels. Moreover, research has shown that CNN performance quickly decreases or collapses when image resolution declines [7] [8]. For any real-world sales context, it's almost impossible for a dataset to encompass tens of thousands of variables for each data sample. Thus, in order to turn sales prediction into a pattern recognition task with CNN, proper feature engineering is of paramount importance.

The feature engineering process for this project was taken from Naya et al., one of the top performers in the Kaggle competition [4] [8], with minor adjustments. In his Medium post, Naya shared how they augmented 79 raw features to 321 total features, including correlation tests, outlier eliminations, new variable creation, etc. I adopted the same process to generate all 321 features. Afterward, I appended three dummy features so that during the training, each sample could be reshaped into 18 x 18 x 1 and treated as if they were grayscale images. The order of pixels follows the correlation between each variable and the target variable, selling price, i.e., the top-left pixel represents the variable that has the highest correlation to the selling price, and the bottom-right pixel has the lowest. Before training, each pixel was scaled to 0~1, as is the case in most image classification tasks. I made three versions of datasets for the three classification tasks by splitting the samples into 3, 5, and 10 categories according to their target values, i.e., selling price. For instance, in the 10-class classification, label = 1 stands for the highest selling price group, and label = 10 for the lowest. The splitting points were decided by quantiles to make sure each category had roughly the same number of samples. In the regression task, I used raw price for each sample, which differs from Naya et al.'s approach of converting prices to log scale. This allows intuitive results without needing to refer to the log values.

Figure 1 shows a few "sales images" after processing. Each image looks like a low-resolution mosaic picture, with every pixel representing a variable in sales, e.g., land area, number of bedrooms, etc. While human beings can hardly sense structures in these low-resolution images, if any, the hope is that CNNs can, as they've long demonstrated, pick up the underlying substructures and "decipher" the secrets lie in these sales images.

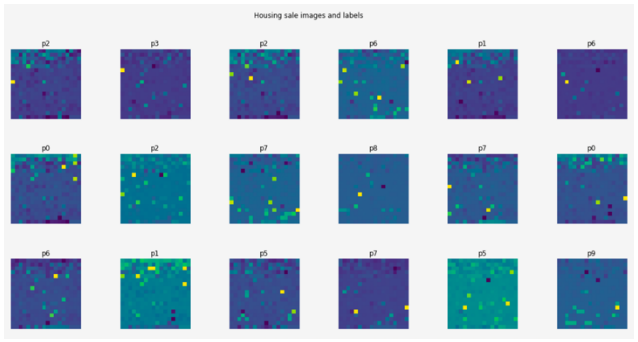

Figure 1: Training "images" in the 10-class classification task, each is treated as an 18 x 18 x 1 grayscale image. p0 ~ p9 are the ten price categories, from the highest to the lowest.

3.3. Tasks

The first part of this project aims to confirm the viability of using CNNs to predict property prices. For this purpose, four types of tasks were carried out: 3-class classification, 5-class classification, 10-class classification, and regression task.

The second part tests whether applying super-resolution techniques to up-sample sales images helps with model performance. The best-performing model from part 1 was taken as the baseline. The model used for super-resolution was "*Enhanced Super Resolution Generative Adversarial Network,"* published by Wang et al. in Tensorflow Hub [1]. The images were up-sampled from 18 x 18 x 1 to 72 x 72 x 3 and then fed into the baseline model with necessary adjustments such as input dimension changes. The rest of model architecture remained the same. Note that the recommended procedure of such an approach is usually to retrain or fine-tune the ESRGAN model on a small subset of domain data before applying it to generate super-resolution images. However, due to the lack of such a subset, I had to use the ESRGAN model out of the box, which may be a limiting factor for this experiment.

	3-class	5-class	10-class	Regression	(Total Images)
Images per Class	~960	~582	~291	*	2911

Table 1: number of images after splitting into 3, 5, and 10 classes. This includes both training and testing sets. The train/test split was conducted with a ratio of 85:15 *Regression task doesn't split and uses all 2911 images

3.4. CNN architectures

A series of preliminary tests were conducted to determine the proper architecture design. In their low-resolution fish species classification research, Rachmatullah et al. suggested that two layers CNNs performed the best on their dataset. However, from my preliminary tests, I found that three layers CNNs tend to perform the best. I preserved the best-performing architecture from the fish classification research as a baseline model and moved on with 3-layer designs.

All models use 3x3 kernels and stride = 1 in all convolutional layers, which is a reasonable design given the minimal size of the images. For classification tasks, there are two fully connected layers: one hidden layer of 128 nodes and one output layer of 3, 5, or 10 nodes, depending on the number of price categories of each classification task. For the regression task, there are four fully connected layers with the number of nodes decreasing by a factor of 2 from 128 to 1. ReLU was used as the activation function in all layers other than the output layer of classification tasks, where Softmax was used to get the prediction probability of each class.

Another factor I set out to test is the pooling mechanism: whether using max-pooling or average-pooling impacts model performance. The rationale is that because of the limited amount of information (pixels) in each image, max-pooling may be too coarse in such a low-resolution setting, tossing useful information and thus reducing model performance, while average-pooling might preserve more useful information and hence has a better overall performance.

Layer	Type	Kernel Size	Stride, Pad	Description
Classification Tasks				
2	Conv + Relu Max Pool Conv + Relu Max Pool	3x3x32 3x3x32	1, 0	The suggested architechture by Rachmatullah et al. in fish species classification research
3	Conv + Relu Max Pool Conv + Relu Max Pool Conv + Relu Max Pool	3x3x16 3x3x32 3x3x64	1, 0	3 layers with max pooling
3	Conv + Relu Avg Pool Conv + Relu Avg Pool Conv + Relu Avg Pool	3x3x16 3x3x32 3x3x64	1, 0	3 layers with average pooling
Dense Layers (applied to classification models)				
	FC + Relu	128		
	FC + Softmax	3 or 5 or10*		*number of classes in each task
Regression Tasks				
3	Conv + Relu Max Pool Conv + Relu Max Pool Conv + Relu Max Pool	3x3x16 3x3x32 3x3x64	1, 0	regression max pooling
3	Conv + Relu Avg Pool Conv + Relu Avg Pool Conv + Relu Avg Pool	3x3x16 3x3x32 3x3x64	1, 0	regression average pooling
Dense Layers (applied to regression models)				
	FC + Relu FC + Relu FC + Relu FC + Relu	128 64 32 1		

Table 2: CNN architectures in the project

3.5. Training

All training was done locally on a 2021 Apple MacBook Pro with an M1 Pro chip and native M1 GPU support for Tensorflow via the Metal plug-in by Apple Inc.

3.5.1. Hyperparameter

All models were trained from scratch. After several rounds of testing, I settled on the Adam optimizer with learning rate = 0.005 since this setting gave the most stable training process. L2 kernel regularization was also added to the last dense layer.

3.5.2. Loss functions & metrics

For classification tasks, the loss function used was the built-in categorical-cross-entropy function in Keras. For regression tasks, I used root-mean-square-error, defined as

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

Where $\hat{y}$ denotes the ground truth and y the predicted price. The metric for classification tasks is accuracy and RMSE for regression tasks.

3.5.3. Data augmentation

It is common to incorporate data augmentation when training with small image datasets to mitigate model overfitting. However, I found the training accuracy suffered severely once augmentation went beyond a certain level. This may be reasonable because each pixel in my "images" is essentially a sales variable. In contrast to flipping an image pixel, flipping "number of bathrooms" may not make much sense in this context. I kept the augmentation and restricted the ratio to 0.1 to randomly flip, rotate, and zoom the images. Hence, the models benefitted from data augmentation but not to a point where adverse training effects started to kick in.

3.5.4. Dropout

Another common technique for training on small datasets is to apply dropouts. Rachmatullah et al. suggested in the fish species classification research that applying dropouts during model training positively influences the testing performance, although only to a minor degree. I applied a 0.1 dropouts ratio after every pooling layer to achieve a minorly smoother training with slightly less overfitting. I did not observe any impact on training and testing performance.

3.5.6. Cross-validation

Due to the small amount of data, I decided to apply 10-fold cross-validation to achieve the best model possible. As mentioned in the data section, 15% of total samples were reserved as the testing dataset, and 85% of data were used for training. During each fold, 90% of training data were used to train the model using Scikit-learn's cross validator (*sklearn.model_selection.KFold.split*), while the remaining 10% was used for validation. The number of epochs was set to 120 with an early-stopping mechanism. The top 5 performing models on the validation data were preserved and later tested on the test dataset to decide the best model.

4. Results

4.1. baseline vs. 3-layer model

First, I compared the 5-class classification accuracy on test datasets of the best 2-layer and 3-layer models. The former is the baseline model that adopts the same architecture from the best model in the fish species classification paper. I used the 5-class classification task to evaluate the efficacy of these two models because, during preliminary tests, the 5-classes task seemed to deliver the most consistent and robust results compared to the 3 and 10 classes tasks.

The 3-layer model performed significantly better than the baseline model, with an almost 10% improvement. This result aligns with my preliminary tests, which suggest that 3-layer models perform best for this specific dataset and task settings.

Setting	Testing Accuracy
2-layer	73.40%
3-layer	83.49%

Table 3: test set accuracy of 2-layer baseline model and 3-layer model

4.2. Overall results

The main results are shown in Table 4. For both types of pooling, I saw a performance drop from fewer classes to more classes, where the best performing models achieved 92.66% in the 3-class classification task and 83.49% in the 5-class task. For regression, the best model managed to pull out a 13880.73 RMSE. Considering most housing transactions in this dataset fall between \$100k ~ \$400k, this is single-digit-error performance. Figure 5 visualizes the regression model's prediction. The diagonal red line represents perfect prediction. The model shows a mostly decent prediction across all values with only a few outliers spreading across \$250k ~ \$420k. Across all types of tasks, max-pooling models surpass their avg-pooling counterparts by a margin. The average-pooling model was able to keep up in the 5-class task but fell short by ~4% in the other two classification tasks; it also produced an extra \$2597 error in the regression task. This was surprising as I was expecting average-pooling models to perform slightly better than max-pooling counterparts.

Setting	Task	Testing Accuracy*
3-layer (max pooling)	3-class	92.66%
	5-class	83.49%
	10-class	67.43%
	Regression	13880.73
3-layer (avg pooling)	3-class	88.99%
	5-class	82.80%
	10-class	63.07%
	Regression	16477.68

Table 4: Performance comparison across tasks and pooling mechanisms. *The metric for regression tasks is RMSE

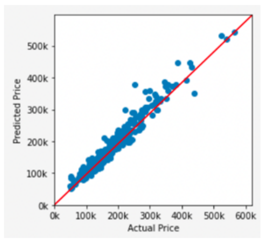

Figure 2: Regression prediction results on test set

4.3 Super-resolution

The result of up-sampling images with the ESRGAN model and then training on the same architecture as the baseline model is shown in Table 5. A performance drop was observed in all types of tasks compared to the baseline 3-layer model. The 5-class classification suffered the least with a 2.53% drop in accuracy, while the 10-class task suffered a 13.3% drop. The RMSE also has a minor increment of 1593.45, bringing the total error to 15474.18, meaning the model's prediction is on average \$15474.18 off the actual sales price for a property.

Setting	Task	Testing Accuracy*
3-layer (max pooling)	3-class	92.66%
	5-class	83.49%
	10-class	67.43%
	Regression	13880.73
SR => 3-layer (max pooling)	3-class	85.32%
	5-class	80.96%
	10-class	54.13%
	Regression	15474.18

Table 5: Performance comparison of baseline model and up-sampling with ESRGAN. *The metric for regression tasks is RMSE

5. Discussion

5.1. CNNs' effectiveness

Overall, all models demonstrated certain degrees of ability to correctly recognize patterns in sales images and make correct predictions across a variety of tasks.

The baseline model that follows the suggested architecture in Rachmatullah et al.'s fish classification research delivered 73.4% accuracy in the 5-class classification task, showing that even a simple two-layer CNN may successfully capture abstract substructures in sales data and learn from such a tiny dataset with merely 2475 training samples. Built upon the baseline model, the three-layer CNN model boosted more than 10% accuracy, achieving an 83.49% accuracy in the 5-class classification task. It is thus safe to conclude that the notion of converting the real estate price prediction problem to an image pattern recognition problem is not only viable but possesses good potential.

5.1.1. Number of classes

It was expected that the models' performance would decrease as the number of classes increased, and this is indeed the case. In the coarse three-class classification task, the best model achieved 92.7% accuracy. The accuracy dropped to 83.5% as the number of classes increased to 5 and 67.4% to 10. While it may seem the model stumbled upon more-classes tasks, there are two factors we need to take into account: (1) price categories are not a clear-cut standard, and (2) the small sample size. These two factors may play a decisive role and should be considered confounds. Consider the process of splitting samples into categories. All samples were sorted and then assigned to a bucket according to their rankings among all. The ambiguity happens on the boundaries of buckets. For samples that lie on the edge of two buckets, it would make sense to classify them into either class as these samples supposedly encompass attributes that resemble both sides, i.e., the abovementioned point (1). The more classes we split, the more ambiguous regions are introduced. In this case, splitting 2475 training samples into ten classes implies that each class contains merely 247 images. First, it might be naïve to expect a model to learn well from this little data. Second, with such a small number of samples, a few wrongly classified images may cause a perceptible influence on the overall accuracy. Third, as explained above, those wrongly classified samples that lie on the boundaries of classes may indicate the model isn't too far off the truth given the nature of housing sales data, but the accuracy metric wouldn't be able to tell the story. Thus, I suggest a practice of keeping at least a 1000:1 samples-to-bucket ratio whenever possible. That is, balancing the number of classes and the size of the dataset to avoid potential confounds. Commercial entities can easily obtain datasets that are larger by magnitudes than the one used in this paper; this grants much potential, and I am confident the model's performance can scale on large-scale datasets.

5.1.2. Pooling mechanism

Contrary to my initial expectation, the max-pooling models constantly outperformed their average-pooling counterparts. As explained in the method section, the expectation was that max-pooling might be too coarse in choosing features on such low-resolution "images" where each "pixel" is essentially a meaningful sales variable. I was expecting average pooling to be able to keep the information contained in the variables better. However, we can tell this is not the case from the result. Figure 3 shows side-by-side comparisons for the two candidates across all four types of tasks. The average-pooling model keeps up well in the 5-class task but falls short in all three others. I reason that although max-pooling leaves out some information, its picking the most influential feature still creates an overall better effect than simply averaging all feature values in each pool.

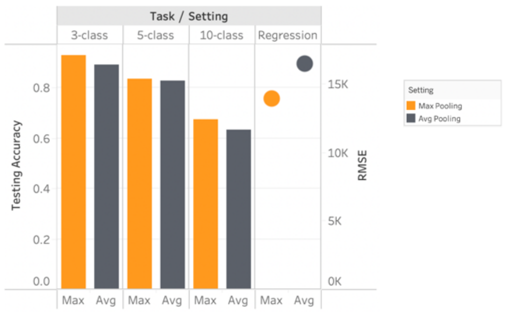

Figure 3: Max-pooling models vs. average-pooling models across tasks. Note the dual y-axes: "Testing Accuracy" on the left is the metric for classification tasks and "RMSE" on the right is for the regression task

5.1.3. Regression

Overall, the regression models reflected their classification counterparts faithfully. Whenever the change in architecture or hyperparameters brought up the classification models' performance, the same behavior can be seen in the regression model and vice versa. This is not surprising since the regression variants simply substitute the last dense layer of the classification models, so naturally, they have the same ability to extract low- and high-level features.

The metric used for regression tasks is the root mean square error. This provides an intuitive way of interpreting the results. For instance, an RMSE of 13880 can be interpreted as the model would produce, on average, a $13,880 error in price estimation for a property. While $13k may seem a lot, considering typical transactions in the real estate housing market where properties are bought and sold at hundreds of thousands of dollars or more, the models actually have a pretty precise sense for price assessment.

Moreover, given the limitation of classification tasks described in section 5.1.1, the regression models may reflect better CNN's capability in such a tiny dataset.

5.2. Effectiveness of Super Resolution

It is a thrilling idea to leverage state-of-the-art super-resolution techniques to enhance image quality and boost the CNN models' performance, as is the case in some prior work [4]. However, for this strategy to play out well, a subset of images to fine-tune the pre-trained model is often necessary. Otherwise, unless the dataset that the backbone model was trained on shares some resemblance with the target dataset, one shouldn't expect the super-resolution model to perform magic out of the box. The ESRGAN model that I used for image quality enhancement was trained on the DIV2K Dataset on bicubically down-sampled 128 x 128 images, which does not resemble the sales data. This explains why the super-resolution process did not just fail to bring up the model performance but caused a negative impact: the same model trained on enhanced images performed worse than that trained on the original 18 x 18 images by 2.5%~13.3%; a similar effect was found in the regression models.

Even if we had enough data, we don't know what the ground truth, i.e., the high-resolution images for fine-tuning of the sales data, should look like. One possible workaround would be to rely on other domains' data that share similarities in terms of abstract structures, such as biomedical and clinical images [7] [8]. All in all, although super-resolution failed to deliver a promising result in this particular context, as should be the case given the stated limiting factors, it does not imply such an approach should be abandoned. On the contrary, more work is required to confirm how the proposed workaround would work on sales data.

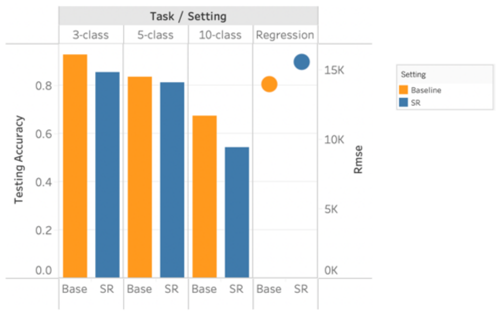

Figure 4: Baseline model vs. same model trained on enhanced images using super-resolution technique

6. Conclusion

In this paper, I explored the notion of transforming the real estate property assessment task into a pattern recognition problem using an existing feature engineering process and leveraging convolutional neural networks to achieve relatively decent results on a tiny dataset with less than 3,000 samples. The result demonstrates the potential of such an approach. Considering the goal of this project was to verify a novel approach to an old problem, this project can be seen as a successful first step toward much more extensive possibilities.

As a part of the project, I explored the possibility of up-sampling low-resolution images with a super-resolution GAN model to improve CNN models' performance. As explained in the discussion, such an approach may require some workarounds to deliver a promising result. Future work includes exploring such possibilities and incorporating larger datasets and possibly improving the feature engineering process to bring the most out of the data. Since image substructures have much to do with pattern recognition, it is worth exploring whether using different arrangements of pixels will impact the result. At the time of writing, I was not aware of any pre-trained model based on sales data or context. As such, image up-sampling or transfer learning approaches require creative workarounds. It would be tremendously helpful if such models become available in the future.

The intersection of real estate and technology is full of virgin land and thrilling opportunities. This project only scratched the surface of a small branch of it, namely, property assessment and price prediction. My work provides a foundation and reference point upon which future work and be built.

Acknowledgments

Big thanks to Professor Belhumeur, Ruoshi, and Haoxuan. You have brought me a wonderful and enjoyable semester. Despite being a relatively deep learning novice and the inevitable stumbling across difficulties and failures along the way, I have learned a lot in terms of both theoretical knowledge and practical skills. This is a great course that will stay in my mind for a long time.

References

[1] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, C. Loy. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. (2018). Computer Vision – ECCV 2018 Workshops

[2] F. Massoli, G. Amato, F. Falchi, C. Gennaro, C. Vairo. CNN-based System for Low Resolution Face Recognition. (2019). SEBD

[3] D. Rustia, Y. Wu, P. Shih, S. Chen. Tree-based Deep Convolutional Neural Network for Hierarchical Identification of Low-resolution Insect Images. (2021). ASABE Annual International Virtual Meeting

[4] D. Cai, K. Chen, Y. Qian, J. Kämäräinen. Convolutional low-resolution fine-grained classification. (2019). Pattern Recognition Letters

[5] K. Robart, K. Angshuman, D. Bhagyashree, S. Borah, M. Monoj. House Price Prediction Using Machine Learning. (2021). The Journal of Philosophy Psychology and Scientific Methods

[6] A. Krzywoszańska, M. Bereta, The use of urban indicators in forecasting a real estate value with the use of deep neural network. (2018). Reports on Geodesy and Geoinformatics

[7] Housing Prices Competition for Kaggle Learn Users. Kaggle (2014). https://www.kaggle.com/c/home-data-for-ml-course

[8] M. Chevalier, N. Thome, M. Cord, J. Fournier, G. Henaff, E. Dusch. (2015). LR-CNN for fine-grained classification with varying resolution. IEEE International Conference of Image Processing

[9] Y. Liu, Y. Qian, K. Chen, J.-K. Kämäräinen, H. Huttunen, L. Fan, J. Saarinen. (2016). Incremental convolutional neural network training. International Conference of Pattern Recognition Workshop on Deep Learning for Pattern Recognition

[10] G. Naya, S. Kumar. Blending of 6 Models, feature engineering and ensembled models for the top 10 in Kaggle "Housing Prices Competition". (2019). https://towardsdatascience.com/feature-engineering-

and-ensembled-models-for-the-top-10-in-kaggle-housing-prices-competition-efb35828eef0

[11] M. Rachmatullah, I. Supriana. Low Resolution Image Fish Classification Using Convolutional Neural Network. (2018). 5th International Conference on Advanced Informatics: Concept Theory and Applications (ICAICTA)

[12] R. Gupta, A. Sharma, A. Kumar. Super-Resolution using GANs for Medical Imaging. (2020). Procedia Computer Science

[13] H. Zheng, P. Fei. Deep Learning for Super-Resolution Biomedical Imaging. (2018). IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)