CS230

State-of-the-Art: End-to-End Deep Learning for Art Appraisal

Computer Vision

Alexander Verge
Department of Computer Science
Stanford University
averge@stanford.edu

Ishaan Singal
Department of Computer Science
Stanford University
ishaan@stanford.edu

Abstract

In this work, we build and test various deep learning models to predict the price of an artwork purely from its image, and attempt to find any meaningful features that make an artwork expensive. We collect 10 years of auction data for fine-art from the world's largest auction house. We establish baseline performance using linear regression and multilayer perceptron models. Subsequently, we train a shallow CNN and several complex CNN architectures for this task and observe the performance on Mean Average Percentage Error. We find that while it is possible to develop models capable of achieving low bias, such models learn features that do not reliably generalize to validation data and result in extremely high variance.

1 Introduction

Valuing art strictly on inherent properties is a challenging task that today requires human experts. Not only is there risk of a valuation being biased due to the prejudices of an individual, but the problem is made even more difficult by attempts of art forgery. The ability to objectively evaluate art in the absence of provided indicators such as the style, artist, medium, and date of creation would provide radical transparency into the intrinsic value of art. Additionally, the discovery of features that correlate with art valuation could provide valuable insight to the art community.

2 Related work

Art appraisal is a widespread problem and has been the subject of open data challenges [1]. However, these projects tend to base their models on the artist and the art style, as opposed to operating on raw image properties. Therefore, artwork classification and style identification techniques [2, 3] that are widely popular, despite providing useful indicators for artwork price prediction, do not represent attempts to directly learn the valuation from pixel input.

Studies have explored the efficacy of using machine learning approaches to predict prices for products based solely on images [4] as well as housing prices based on satellite imagery [5, 6]. There have also been attempts to commercialize applications of image based art valuation [7]. However, there is a dearth of academic literature specifically on the topic of art valuation from images. This is likely in part due to the absence of a standard dataset as well as the tremendous difficulty of the task even for experts.

3 Dataset and Features

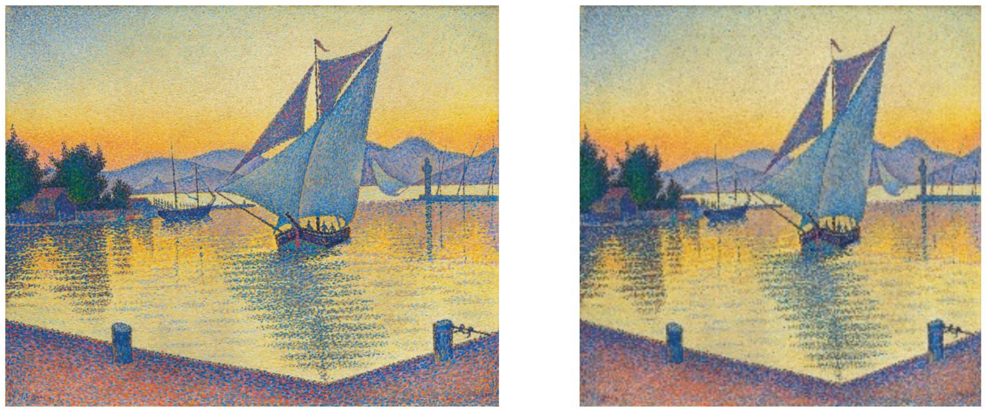
(a) Original image (b) Resized 256px $\times$ 256px image

Figure 1: *Le Port au soleil couchant, Opus 236 (Saint-Tropez)* by Paul Signac (1863-1935) auctioned as part of the collection "Impressionist and Modern Art Evening Sale" with realized value 19,501,250 GBP on February 27, 2019. [8]

Given that there is no publicly available dataset of art images and their price valuations, we collected our own dataset by scraping Christies (popular auction house) [8] for auctions categorized as fine-art, obtaining both raw images and realized auction prices. In total, 109,247 data points were acquired for auctions dated between 2010 and 2019. All valuations were converted to USD.

We faced two issues with this original dataset: (1) it contained many images of items such as furniture, antiques, and ceramics that are dissimilar to the type of data that we were targeting (paintings and drawings) and (2) the data was positively skewed due to a long right tail. We rectified both of these concerns by discarding data and keeping only "high-value" art, which we defined as having a valuation of at least $100,000$ USD. This both eliminated the majority of art categories outside paintings and drawings, which tended to be more expensive, and mitigated the data skew by shifting attention to the long tail. This transformation left us with 20,881 data points, which we divided into train/dev/test sets using a 80/10/10 split, allocating 16705/2088/2088 examples.

These images of varied dimensions were pre-processed and down-sampled. We considered the resized dimension to be a hyper-parameter and tuned it by (1) manually inspecting example images of different sizes for what appeared to be sufficient granularity to interpret the image as a human baseline and (2) training our baseline models on different settings of the hyper-parameter. We evaluated square images with widths of 64px, 128px, 256px, and 512px. We opted for 256px$\times$256px for all of our experiments as we found that lower dimension values had astonishingly poor baseline performance, and higher values resulted in significantly greater computational requirements. Additionally, dimensions of 256px $\times$ 256px allowed us to train our models on complex CNN networks like AlexNet and VGG16. An example image acquired through [8] and its resized counter-part are presented in Figure 1.

4 Methods

4.1 Evaluation metric

For quantitative regression problems, there are many suitable evaluation metrics like Root-Mean-Squared-Error (RMSE), Mean Absolute Error (MAE), and Mean Absolute Percentage Error (MAPE). We selected MAPE as the optimizing metric because of the desirable property of being easily interpreted as the percent error of the appraisal.

4.2 Network Architectures

We explored a series of architectures with progressively increasing complexity in an attempt to improve validation loss. Additionally, we tuned hyper-parameters with grid search over all proposed values to find an optimal configuration for each individual model.

4.2.1 Linear regression

We trained linear regression (LR) as a rudimentary baseline using the Adam optimizer with the default momentum ($\beta_1 = 0.9$) and RMSProp ($\beta_2 = 0.999$) hyper-parameters. The learning rate, α, was the only hyper-parameter tuned for this model.

4.2.2 Multilayer perceptron

Relatively shallow multilayer perceptron (MLP) models were also trained as a baseline model for this task. These models also used the Adam Optimizer with default values of β_1 and β_2. The hyper-parameters tuned were the learning rate α, the number of hidden layers $l \in [5, 10, 25]$ and number of units per hidden layer $n_l \in [5, 10, 25]$. The number of units per hidden layer was kept constant for all layers.

4.2.3 Vanilla CNN

Next we applied a very basic, shallow convolutional neural network (CNN) which we call "Vanilla CNN". The objective here was to evaluate and understand the performance of a very simple CNN and investigate if there is any immediate and obvious advantage of this type of architecture over the baseline LR and MLP models. This model consisted of a single convolutional layer with 64 kernels of size $k \in [5, 11]$, followed by a 2 by 2 max-pooling layer with stride 2, and finally a fully-connected layer with ReLU activation.

4.2.4 AlexNet

AlexNet [9], a model originally developed for the ImageNet Large Scale Visual Recognition Challenge, has become a very popular architecture and is now widely applied to different problems. Given that it is a relatively small model with 5 convolutional layers followed by 3 fully-connected layers, we decided to train the whole model. In total there are roughly 62 million trainable parameters. We replaced the final classification layer with a linear layer and ReLU activation to get a regression value, and tuned the learning rate α and the keep probability for the dropout layer as the hyper-parameters.

4.2.5 VGG-16

Similar to AlexNet, VGG-16 was originally developed for object recognition and is now widely popular. VGG-16 applies kernels that are similar to AlexNet, but uses many more filters, and has more than twice as many trainable parameters (138 million), making it a logic next step from AlexNet to improve model performance. As VGG-16 is a significantly larger network than AlexNet and contains many layers, we used pre-trained weights to leverage already learned low-level features from a larger dataset (ImageNet) and to reduce training time. In addition to the learning rate, the number of unfrozen layers l_u from the pre-trained network as well as the number of added layers and the number of units in each were tuned for this model.

5 Experiments & Results

5.1 Linear Regression

We found the optimal learning rate for linear regression to be $\alpha = 0.01$. As can be seen in Figure 2, the linear regression model improves on train and validation MAPE very rapidly for only a few iterations before flatlining. While there is some sign of overfitting to the training data, realized as a gap between train and validation MAPE, it is obvious that this model has a far more egregious bias problem than variance.

5.2 Multilayer Perceptron

We found that the MLP model performed best with hyper-parameters learning rate $\alpha = 0.001$, $l = 25$ hidden layers, and $n_l = 5$ units per layer. Due to the greater expressive capacity of this model, it was able to outperform the baseline. Additionally, it achieved its minimum validation loss far earlier. As a result, we applied early stopping on the 40th epoch. With this configuration, the MLP outperformed linear regression by over 4.7 absolute percentage points on the validation set.

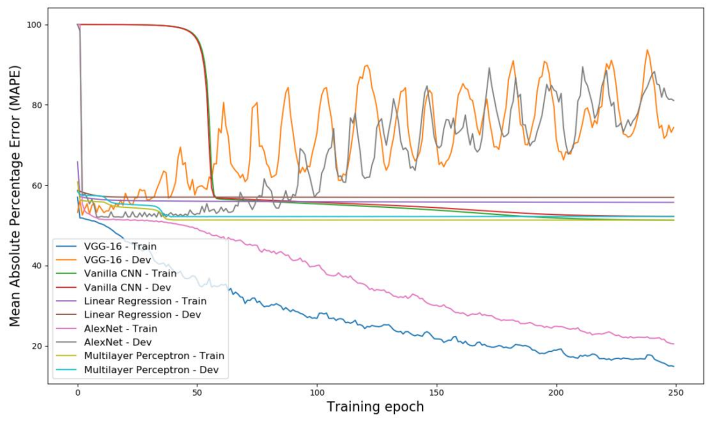

Figure 2: Train and validation performance for all evaluated models with optimal hyper-parameter configurations (see model specific details in Experiments & Results for hyper-parameters).

5.3 Vanilla CNN

The Vanilla CNN architecture that we evaluated had a striking learning pattern: very slow progress until a period of rapid improvement. This was likely due to a plateau in the loss function over the parameters. Ultimately, under the optimal hyper-parameters of kernel size $k = 11$ and learning rate $\alpha = 0.001$, the model achieved a final performance that was nearly identical to the MLP. While it was disappointing that this model did not have incremental improvement upon MLP, it was not surprising given how simple the architecture is. In fact, the ability of this rudimentary architecture to achieve parity with an MLP that leveraged an order of magnitude more trainable parameters was a promising sign that more sophisticated convolution based models might be capable of far better performance.

5.4 AlexNet

As can be seen in figure 2 AlexNet demonstrated a significant capacity to overfit the training data, achieving remarkably low train loss over 250 training epochs. The validation loss, however, achieved only a minuscule improvement over the MLP and Vanilla CNN models. After observing the patent high variance of this model, we explored regularization techniques to address it. We applied L1 and L2 regularization, neither of which mitigated the variance, and finally tuned the dropout configuration. In the end we found the optimal hyper-parameters to be learning rate $\alpha = 0.0001$ and dropout keep probability of 50% along with the use of early stopping.

Model	**Training set MAPE**	**Dev set MAPE**	**Test set MAPE**
Linear Regression	55.72	56.96	56.12
Multilayer Perceptron	51.30	52.13	51.38
Vanilla CNN	51.29	52.24	54.76
AlexNet	51.43	52.01	51.46
VGG-16	51.82	52.52	51.78

Table 1: Final performance for each model with its optimal hyper-parameter configurations (see model specific details in Experiments & Results for hyper-parameters). Training and dev set MAPE are reported at the point of early stopping when applied.

5.5 VGG-16

VGG-16, given its greater expressive capacity relative to AlexNet, was even more capable of overfitting to the training dataset and achieved the lowest training set error rates over 250 epochs. Disappointingly, the variance problem also persistent for VGG-16. We again applied the typical techniques to mitigate variance (L1 and L2 regularization) to no avail. Given that we applied transfer learning, we tuned both the number of frozen layers and the configuration of added layers in an attempt to address overfitting. We found the optimal hyper-parameter setting to be when no layers were added, the two final layers were unfrozen, and the learning rate was set to $\alpha = 0.001$ with the application of early stopping.

6 Conclusion/Future Work

While we were successful in developing models that are capable of achieving low bias, we were unable to mitigate a severe variance problem that would render our models unusable in any real-world application. We believe there are two clear reasons for the inability of our models to generalize to unseen data. First, it is often the case that the value of an artwork is not comprehensively encapsulated by RGB values. An artworks price is heavily influenced by the artist, style, medium, era of creation, market liquidity and efficiency, as well as many other factors and biases. The second is that even given perfect information about all the factors outlined, this is still a remarkably challenging task even for human experts. This is a hard problem. While there is no well established human baseline performance to serve as a proxy for Bayes error, this is a task that necessitates trained experts, and even the reliability of expert estimates is unknown.

We believe there are a few paths forward that could help provide interesting insight on model performance as well as result in meaningful performance improvements.

6.1 Data collection

While we collected over 100,000 images of artwork spanning a decade, we focused only on high-value art and had less than 20,000 data points in the training set. Collecting more data is a well established method for combating high variance, and would be a suitable path for improving this model.

6.2 Additional data inputs

Given that the entirety of an artworks value is not captured by RGB values, we believe that it may be possible to achieve vastly superior model performance by feeding in additional data such as artist and art-style.

6.3 Human baseline

Establishing a human baseline for performance of the task would provide a useful benchmark for evaluating the bias of the model.

6.4 Saliency maps, learned feature visualizations

Visualizations of the learned features could provide very interesting insights to the art community on visual patterns that correlate with art value.

7 Contributions

The authors had equal contribution, working together for data-collection, developing/training the models and setting up GPU instances on Azure and AWS. Ideas were discussed together and the work was shared in creating the report, poster, and presentation video.

The authors are grateful to Jay Whang for his insight, guidance and mentorship throughout this project.

References

[1] Painter by Numbers. (n.d.). Retrieved April 15, 2019, from https://www.kaggle.com/c/painter-by-numbers/

[2] Bosch, A., Zisserman, A., & Munoz, X. (2007). Image Classification using Random Forests and Ferns. 2007 IEEE 11th International Conference on Computer Vision. doi:10.1109/iccv.2007.4409066

[3] Khan, Fahad & Beigpour, Shida & Weijer, Joost & Felsberg, Michael. (2014). Painting-91: A large scale database for computational painting categorization. Machine Vision and Applications. 25. 1385-1397. 10.1007/s00138-014-0621-6.

[4] Chen, S., Chou, E., & Yang, R. (2018). The Price is Right: Predicting Prices with Product Images. 1-6. Retrieved from http://cs229.stanford.edu/proj2017/final-reports/5237321.pdf

[5] A. J. Bency, S. Rallapalli, R. K. Ganti, M. Srivatsa and B. S. Manjunath, "Beyond Spatial Auto-Regressive Models: Predicting Housing Prices with Satellite Imagery," 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), Santa Rosa, CA, 2017, pp. 320-329.

[6] S. J. Semnaji and H. Rezaei. House Price Prediction Using Satellite Imagery. http://cs230.stanford.edu/projects_spring_2018/reports/8303683.pdf

[7] Shazam for Art - Art App. (n.d.). Retrieved April 15, 2019, from http://www.magnus.net/

[8] Christie's Auctions & Private Sales | Fine Art, Antiques, Jewelry & More. (n.d.). Retrieved April 15, 2019, from https://www.christies.com/

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. pages 1097–1105, 2012.

[10] M. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In ECCV, 2014.