**Project Report - Predicting Car Price Using Machine Learning**

*Team 6 - Buqi Liao, Dennis Wang, Eric Tang*

## Abstract

This project investigates the use of Convolutional Neural Networks (CNNs) to predict used car prices based solely on vehicle images. Utilizing the DVM-CAR dataset with over 1.4 million images of 899 car models, we addressed class imbalance and long training times by preprocessing the data (resizing images, normalizing pixels, and filtering classes) to create a balanced dataset of 225,206 images with around 400 car models.

Our optimized CNN model consists of two convolutional layers followed by dense layers for regression. It achieved a Mean Absolute Error (MAE) of 8,882.9 and an $R^2$ score of 0.936, explaining 93.6% of the variance in the data. This performance significantly surpasses that of a Support Vector Machine (SVM) regressor, which had an MAE of 21,567 using features from a pre-trained VGG16 network.

By simplifying the model architecture and refining the dataset, we improved both training efficiency and prediction accuracy. The findings demonstrate that a well-designed CNN can effectively predict used car prices from images.

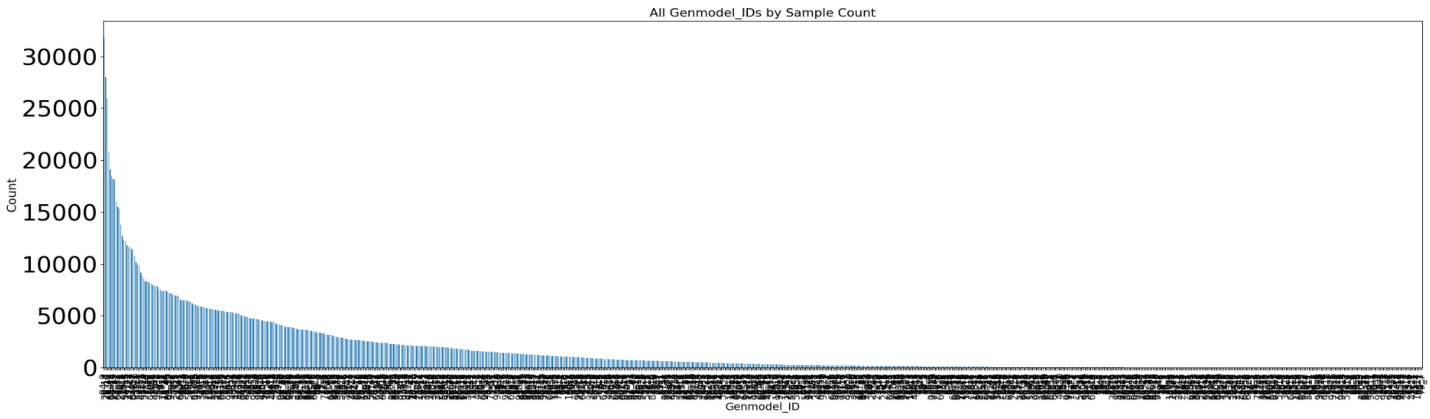
## Introduction

The rapid growth of the used car market has created a strong demand for accurate price prediction models to assist both buyers and sellers in making informed decisions. Traditional methods for valuing used cars often rely on manual appraisals or basic regression techniques, which may fail to capture complex patterns. Recent advancements in AI have introduced more robust approaches, such as Stanford's "AI Blue Book: Vehicle Price Prediction using Visual Features," which uses a custom CNN architecture, PriceNet, to predict vehicle prices based on images. Building on this foundation, our project aims to explore the application of Convolutional Neural Networks (CNN) or Support Vector Machines (SVM) to predict used car prices using visual and other relevant features. Our goal is to train a model capable of accurately predicting the prices of popular car models commonly seen on the streets, minimizing prediction error.

## Data

We utilized the DVM-CAR dataset, comprising 1,451,784 images of 899 car models from the UK market. The images were resized to a resolution of 300x300 pixels, and their backgrounds were removed for consistency. Each image is annotated with a *Genmodel_ID*, serving as a unique identifier for each car model; an *Image_name*, which is the filename of the car image; and an *Entry_price*, representing the listed price of the car.

**Data Distribution:** The *Entry_price* in the dataset has a mean of 28,097.8, a median of 18,045.0, a standard deviation of 35,052.4, and a mean absolute deviation of 18726.45. The dataset exhibited a significant class imbalance, with some car models having over 30,000 images while others had as few as one. Here's a graph of image counts against its corresponding *Genmodel_ID:*



To mitigate dataset imbalance, we implemented several measures during data preprocessing to reduce the dataset size and improve balance.

**Class Filtering:** We began by dropping underrepresented classes, specifically those with fewer than 300 images, to eliminate noise and ensure adequate representation across all categories. For overrepresented classes with more than 500 images, we randomly sampled 500 images to achieve uniformity across all classes.

**Image Normalization:** We resized all images to a resolution of 128*128 to maintain uniformity and normalized pixel values to a standard range of [0, 1] which facilitates more efficient and effective model training.

As a result of these preprocessing steps, the final dataset was refined to include 225,206 images evenly distributed across around 400 car models, ensuring both a manageable size and balanced representation for robust model training**.**

## Model

Our best performant mode (we'll refer to it as the base model) consists of two convolutional layers: the first with 16 filters of size 3×3 and ReLU activation, followed by a 2×2 max-pooling layer, and the second with 32 filters of size 3×3 with ReLU activation, also followed by 2×2 max-pooling. These layers extract features from the input image. The output is flattened into a 1D vector and passed through a dense layer with 64 neurons and ReLU activation to combine features. Finally, a dense layer with 1 neuron outputs the regression prediction.

Yang, Chen, and Chou (2018) proposed a CNN model with several convolutional and dense layers to achieve image-to-float output. This approach inspired our base model. Through trial and error, we made adjustments based on the results of our experiments. The table below shows all the models we experimented with, along with their corresponding performance metrics. Those models are described relative to the base model for clarity.

| | Model Description (all descriptions relative to the base model) | MAE | R^2 Score |
|---|---|---|---|
| 1 | Base model | 1705.9 | 0.892 |
| 2 | One extra convolution layer | 1731.9 | 0.878 |
| 3 | One less convolution layer | 3148.6 | 0.670 |
| 4 | With regularization | 1850.7 | 0.868 |
| 5 | Dropped neurons | 3986.6 | 0.478 |
| 6 | Double amount of kernels | 1832.8 | 0.873 |
| 7 | Use sigmoid activation function | 6117.7 | –0.032 |

To ensure comparability and reduce training time, all models were trained and evaluated using the same dataset, which consisted of 1,600 images randomly selected from the 1.5 million images in the main dataset.

**Impact of Convolutional Layers on Model Performance:** We evaluated the effect of the number of convolutional layers by comparing three models: the base model with two layers (16 and 32 filters), Model 2 with an additional 64-filter layer, and Model 3 with only one layer. Model 2 showed marginally worse performance (less than 2% increase in MAE and drop in $R^2$ score) than the base model, suggesting two layers are sufficient for this dataset. Model 3 performed significantly worse, with an 81.9% higher MAE and a 31% lower $R^2$ score, highlighting the inadequacy of a single convolutional layer.

**Overfitting Prevention Techniques:** We tested L2 regularization (Model 4) and dropout (Model 5) to prevent overfitting. L2 regularization led to an 8.4% higher MAE and a 2.6% lower $R^2$ score, while dropout caused severe degradation (130% higher MAE and 46% lower $R^2$). These results suggest the base model is well-fitted, and additional techniques lead to underfitting.

**Kernel Count and Activation Function:** Doubling the number of kernels (Model 6) resulted in slightly worse performance (7.4% higher MAE, 2.1% lower $R^2$), indicating the base kernel configuration is adequate. Replacing ReLU with a sigmoid activation function (Model 7) severely hindered performance (258.6% higher MAE, $R^2$ of –0.032) due to vanishing gradients.

From these observations, we conclude that our base model has an optimal configuration of convolutional layers and achieves a good balance between overfitting and underfitting. Any adjustments made to the model, such as increasing layers, modifying kernel counts, or introducing regularization techniques, consistently resulted in worse performance. This reinforces the suitability of the base model's architecture for this dataset.

In addition to our base model, we experimented with a Support Vector Machine (SVM) using features extracted from images via a pre-trained VGG16 network. Features were normalized, and the SVM was trained with an SGD-based regressor, achieving a mean absolute error (MAE) of 21,567. While the base model excels in capturing complex spatial features due to its convolutional architecture, the SVM's linear approach struggled with the non-linearity of the data, resulting in lower performance. Thus, the base model is better suited for predicting car prices in this context.

### Evaluation

We used MAE, RMSE, and R^2 values to evaluate the performance of our base model, the metrics for our best-performant model are given in the following table

| MAE | RMSE | R^2 | Dataset Size | Training Time | Testset Size | Trainset Size |
|---|---|---|---|---|---|---|
| 8,882.9 | 16,004 | 0.936 | 225,206 | 1h43m | 49,038 | 176,168 |

Our model explains 93.6% of the variance in the data, indicating a good fit. However, the Mean Absolute Error (MAE) of 8,882.9 is relatively high. Despite this, the high R^2 value assures us that the model is well-fitted and does not exhibit signs of overfitting. Furthermore, compared to the dataset's Mean Absolute Deviation (MAD) of 18,726.45, our model reduces the error significantly compared to randomly selecting values from the dataset.


The graph on the left illustrates the loss function over the training epochs. The loss function used is Mean Squared Error (MSE), normalized to enhance readability. We observe that the training loss decreases steadily as the model learns to minimize errors on the training data. A similar pattern is observed for the validation loss. The very small difference between the training loss curve and the validation loss curve indicates that the model is not overfitting, as it performs well on unseen data.

To conclude, the model effectively reduces error and generalizes well to unseen data. The alignment of training and validation losses confirms that it does not overfit. However, there is still room for improvement in reducing the MAE. Below are some sample prediction outputs from our model.

Predicted price: 26025.46, Actual price: 29365.00

Predicted price: 16119.73, Actual price: 17500.00

Predicted price: 26271.97, Actual price: 26665.00

## Discussion

The first challenge we encountered was related to model design. We initially aimed to predict car prices by separating the process into two stages: extracting features from images using a classifier and then feeding these features into a regression model for price prediction. This modular design can independently optimize each stage and evaluate their respective contributions. To achieve this, we utilized multi-class classification models, including CNNs and pre-trained architectures like VGG-16 and ResNet50, to extract features such as car model, production year, engine type, etc. While the model achieved an impressive accuracy of 0.98 in extracting the car's model, the performance for other features was unsatisfactory, with accuracies below 0.3.

The unbalanced feature extraction critically impacted the regression model's performance, resulting in a Mean Absolute Error (MAE) of over 20,000, even exceeding the Median Absolute Deviation (MAD), indicating the model's inability to capture meaningful price patterns. Recognizing this limitation, we pivoted to a direct prediction approach, combining the feature extraction and regression into a single end-to-end model. By using CNNs to predict car prices without intermediate feature extraction directly, we transformed the task into a black-box learning problem. This adjustment significantly improved performance, achieving an MAE of around 1,800 on small datasets and 9,000 on larger datasets, showcasing the effectiveness of simplifying the model structure for this task.

Another major problem we tackled is the training time. Initially, our model design included a combined classification and regression approach, which required substantial computational resources. When we attempted to train the model using the entire dataset, the process took over 24 hours and only reached about halfway through completion. This outcome was unsustainable for further development.

Our first solution was to enable GPU acceleration by downloading the appropriate TensorFlow version that supported GPU access. While this step significantly reduced the training time, the improvement was still insufficient, the training process continued to exceed 24 hours, which was impractical for iterative model refinement.

Recognizing that the dataset itself contributed to the training bottleneck, we decided to reduce its size strategically. As noted in the Data section, our dataset was unbalanced, with some car models having only a few images. These models were unlikely to yield reliable predictions due to insufficient data. Thus, we removed car models with minimal images from the dataset. For models with a larger number of images, we randomly selected a subset for training. This approach allowed us to control the dataset size without compromising its representativeness.

| Number of pictures/model, model eliminate condition | Data Set Size | MAE | R^2 |
|---|---|---|---|
| 500, <300 | 225206 | 8882.9 | 0.936 |
| 200, <20 | 113668 | 15524 | 0.65 |
| 10, <10 | 5950 | 17173 | 0.52 |
| 50, <10 | 29085 | 15958 | 0.64 |
| 50, <200 | 25300 | 13513 | 0.61 |

However, this solution introduced a secondary challenge: balancing the trade-off between the model's ability to cover a wide variety of car types and its prediction accuracy. Including more car types required reducing the number of images per type, which could negatively impact prediction quality. To address this, we conducted a series of trial-and-error experiments to identify the optimal dataset configuration. Referring to the table, our tests revealed that using 500 images per car type provided a reasonable balance between accuracy and training efficiency. It optimizes the MAE and R^2 while keeping a relatively small data set. By the model eliminating condition of "<300", we will have around 400 types of vehicles in total.

Through these adjustments, we successfully mitigated the issue of long training time while maintaining a satisfactory prediction performance.

**Reference**

Huang, J., Chen, B., Luo, L., Yue, S., & Ounis, I. (2022). DVM-CAR: A large-scale automotive dataset for visual marketing research and applications. https://arxiv.org/pdf/2109.00881

Yang, R. R., Chen, S., & Chou, E. (2018). AI Blue Book: Vehicle Price Prediction using Visual Features. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1803.11227

| Number of pictures/model, model eliminate condition | Model type | Data Set Size | MAE | R^2 |
|---|---|---|---|---|
| 500, <300 | 400 | 225206 | 8882.9 | 0.936 |
| 400, <200 | 482 | 202420 | 10589 | 0.73 |
| 200, <20 | 684 | 113668 | 15524 | 0.65 |
| 10, <10 | 742 | 5950 | 17173 | 0.52 |
| 50, <10 | 742 | 29085 | 15958 | 0.64 |
| 50, <200 | 482 | 25300 | 13513 | 0.61 |