UNIVERSITY OF GRONINGEN

NEURAL NETWORKS

Predicting Car Prices from Images

github.com/blat-blatnik/Opti-Car

ATANAS JANESHLIEV, s3814254
BORIS RADOVANOVIC, s3301419
PAUL PINTEA, s3593673
SALIH SALEH, s4170237

7th July 2020

Abstract

In this project, a neural network based image regression system was developed in order to predict the manufacturer's suggested retail price of automobiles solely off of their image. We present an original artificial neural network architecture loosely based off of AlexNet. The ANN was trained on 13,000 images from the Car Connection Picture Dataset. The tuned final model was found to predict prices of cars with a higher degree of accuracy than a classical multiple-linear regression model, obtaining a mean absolute percentage error of 13.5%. In this report we present the design and performance of the model, the various pitfalls encountered during its design, as well as possible future extensions and improvements.

Keywords: price prediction, image regression, artificial neural network, convolutional neural network, dataset.

Contents

Introduction

The rise of the internet and its rapid evolution has brought upon the biggest change to humankind yet. It has managed to fully reshape daily human life, and alongside that revolutionize the means via which we conduct even our simplest activities. The changes to the ways we go throughout life are numerous, however for now let us focus on how the internet has influenced trade.

In the middle to late 90s, the first online marketplaces had begun to pop-up. Most of these marketplaces were simple websites where any user could create a listing for an item they wanted to sell, or were requesting to buy. Over time these marketplaces had improved significantly and as of 21st century e-commerce has become one of the most popular online businesses (Reynolds, 2000).

Withing this realm, automobile marketplaces have been rather slow to adopt the new e-commerce paradigm (Marshall, Sor, & McKay, 2000). Customers wishing to purchase cars online have very limited options, and the options that they do have are seen as untrustworthy and risky as the customer has no way of verifying car prices before purchase (Smith, 2009). As such there is a massive demand for the implementation of features that would help users determine whether they are getting a good deal or not (Häubl & Trifts, 2000; Rusmevichientong, Salisbury, Truss, Van Roy, & Glynn, 2006).

The drastic advancement in the field of artificial neural networks over the past couple of years has rendered their utilization fairly suitable for the development of such features. This is especially true when it comes to tasks such as image recognition, categorization, as well as parameter prediction (Hijazi, Kumar, & Rowen, 2015). Artificial neural networks were also historically used for a relatively long time for price prediction tasks, even before the recent explosion of interest in neural networks (White, 1988).

Thus we tailored our project to the development of a feature that would help out potential automobile customers in their purchase decision by utilizing artificial neural networks. More specifically our project aimed to develop an artificial neural network which could predict the manufacturer's suggested retail price (MSRP) of an automobile based on an image of the automobile in question. This could greatly improve a customer's trust in the automobile they are purchasing, thus potentially increasing sales while simultaneously minimizing the purchase risk.

Development of such a system requires a relatively large dataset consisting of car images from various manufacturers, numerous models, and many price points. These car images would also need to be accompanied with the corresponding MSRP of the car. The dataset is described in detail in the following section. The dataset used will be from Gervais (2020).

Using such a large car image dataset, we hope to develop an artificial neural network (ANN) architecture to predict the MSRP of cars. In order to evaluate the performance of this ANN, we will also develop a multi-linear regression model to predict the MSRP of cars based on the manufacturer, model, horsepower, etc. rather than basing it solely on an image. Regression models are a more traditional means of price prediction (Miller & Masters, 1973) and as such we believe that comparing this regression model to the ANN model will provide an accurate assessment of the potential performance, accuracy and flexibility that can be gained from using an ANN to perform car MSRP prediction.

We predict that the ANN model will outperform, or at least match the performance of the classical multi-linear regression model when used to predict the MSRP of cars. The measurement used to assess model performance will be the mean absolute percentage error of the price predictions made by the models.

Data

The dataset used for this project is based on the dataset by Gervais (2020). This dataset consists of roughly 60,000 images of cars that were scraped from The Car Connection. The full dataset also includes images of car interiors, duplicate images, images of car design concepts, images including more than one car, as well as other images that were unfit for the purposes of this project which were removed from the final dataset. Example images from the full dataset can be seen in Figure 1.

Figure 1: Example images from the full dataset.

All images from the dataset fit into a 320×320 pixel bounding box. The file names of each image included the make, model, suggested retail price, front wheel size, horsepower, displacement, engine type, width, height, length, gas mileage, drive-train, passenger capacity, number of doors, and body style of the car in the image. A python script was used to compile this data into a single csv file. The images from the full dataset were then cleaned, filtered and compiled into the final dataset used for training. An overview of the data cleaning pipeline can be seen in Figure 2

The duplicate images from the full dataset were removed using a python script that searches for exact pixel-perfect matching duplicate images from the dataset. Since this script only detects exact pixel-perfect duplicates, some images that could be considered duplicates remained in the dataset if they were off by one or more pixels. An example of this can

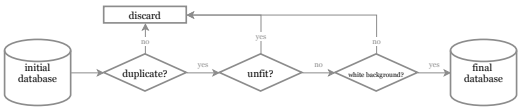

Figure 2: An overview of the data cleaning pipeline that was used to construct the final dataset.

be seen in Figure 3. The other forms of unfit images were removed from the dataset manually. All images were thoroughly examined during the removal, however some unfit images may have still remained in the dataset. Table 4 shows examples of which images were removed by hand. Roughly 32,000 images were removed through these two steps, leaving a usable dataset that consists of 28,000 images.

Figure 3: Examples of 3 duplicate images of the Chevrolet Trailblazer 2006 in the full dataset. The images on the left and in the middle were correctly detected as duplicates, while the image on the right was not. This is because the image on the right has a small water-mark in the bottom right corner.

The 28,000 images that remained after the previous data cleaning steps were separated into car images that have a purely white background, and car images that have a more complex background using a python script. The script separated the images based on the color values of the pixels on the perimeter of the image. If any of these pixels had a red, green, or blue component smaller than 240, the image was categorized as having a more complex background, otherwise it was categorized as having a purely white background. After running the script, a visual check was performed and confirmed that the images were categorized properly. Roughly 13,000 images were categorized as having a white background, and 15,000 images were categorized as having a more complex background. The 13,000 images with a purely white background were compiled into the final dataset that was used to train the network. Example images from the final dataset can be seen in Figure 5.

Of the 15,000 images that were categorized as having a more complex background, 3,000 of them were uploaded to a free online image background removal tool, photoscissors.com (2020). Since only a single image could be uploaded at a time, the process of uploading these images was automated using a python script that took control of the mouse and keyboard input of the computer in order to "manually" upload the images. The quality of the images that were cleaned using this online tool was reasonable, however it was noticeably worse than the quality of the images that originally had a white background. As a result, these cleaned images were

description	example	justification
interior		outside of the scope of this project
open doors		obstructs the exterior of the car
open trunk		obstructs the exterior of the car
open hood		obstructs the exterior of the car
multiple cars		can be impossible to focus on a single car, the network will get confused
reflections		same problem as multiple cars
car missing		outside the scope of this project
part of car missing		hard to determine price if most of the car is missing
complex background		hard to distinguish car from background
concept car		outside of the scope of this project
car is obstructed		crucial details may be obstructed from view
car too small		cannot discern important details from only a few pixels
hazy image		hard to discern important details
person in image		hard to separate the foreground from the background
wrecked car		we are predicting the suggested *retail* price however a wrecked car would have a significantly decreased value on the market
car not in foreground		difficult to separate the car and the background or foreground

Figure 4: Examples and justifications for all images from the original full dataset that were deemed unfit and excluded from the final dataset.

Figure 5: Example images from the final dataset used to train the model. All 3 images have a purely white background, and the bounding box of the car fits entirely within the image.

Figure 6: An image of a Honda Accord 2014 with a background (left). The background was then removed (middle). Compared to an image that did not have a background initially (right), the quality of the cleaned image is significantly lower. For example, there are noticeable artifacts on the front tire in the middle picture.

not included in the final dataset. Figure 6 shows an example of an image whose background was removed in this manner.

An R script was used to analyse the distributions of cars in the final dataset. Plots of 4 different distributions are shown in Figure 7. The car prices approximately follow a log-normal distribution with $\mu = 40685$, $\sigma = 29850$. Most cars from the dataset were manufactured between 2010 and 2020 (90%), with most cars being manufactured in 2011. Most cars were either SUVs or standard 4-door cars, and most cars were manufactured by Chevrolet, Ford, or Toyota (25%).

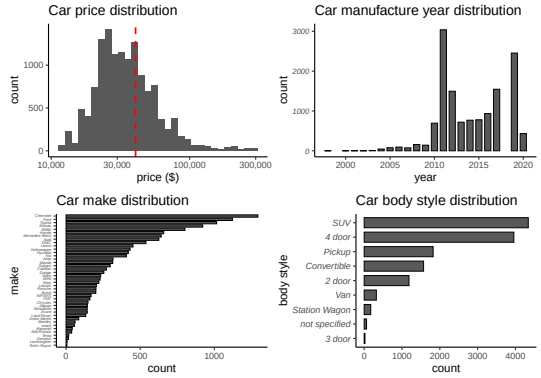

Figure 7: The distributions of the cars in the final dataset. The distributions of car prices, manufacture year, make, and body styles are shown in the top-left, top-right, bottom-left and bottom-right respectively.

Methods

The project is made up of a single pipeline which is described in detail below. The pipeline consists of three parts: a data pre-processing step, a model training step, and an evaluation step.

Pre-processing

At the pre-processing step each image from the dataset is loaded. The MSRP attached to the title of the image file is extracted. The image is then converted to grayscale, padded with white pixels to a square size, and then finally resized to a resolution of 150×150 pixels using a high quality Lanczos interpolation over an 8×8 pixel neighborhood as described by Burger and Burge (2010).

Furthermore the MSRP of the cars is normalized to a range of [0, 1] by dividing with the maximum price of all cars in the dataset. The car image pixel values were also normalized to a range of [0, 1] by dividing by 255. These steps were were previously shown to increase the stability of the neural network model (Sola & Sevilla, 1997).

The model was trained through cross-validation, so the images were randomly separated into a training set and a validation set, with a ratio of 90:10 respectively. This ratio was used in order to increase the variety and the size of the training set. However by making the validation set smaller than what is normally recommended, we risk over-fitting and reducing the model's generalizability.

Model training

Once the dataset pre-processing has been completed, the model training was run for 500 epochs. The training was done with a batch size of 64 and a learning rate of 0.001 for the final model. The training was performed on a mobile Nvidia GTX 1050 Ti, with 4GiB of video RAM, and 16GiB of RAM. The model is described in detail below.

The ANN model that we used was *sequential*, with $150 \times 150 = 22500$ input neurons for the image, 1 output neuron for the price prediction, and 6 hidden layers. The model architecture is loosely based off of the successful AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), which was modified to fit the purpose of this particular task. A visual representation of the model architecture can be seen in Figure 8.

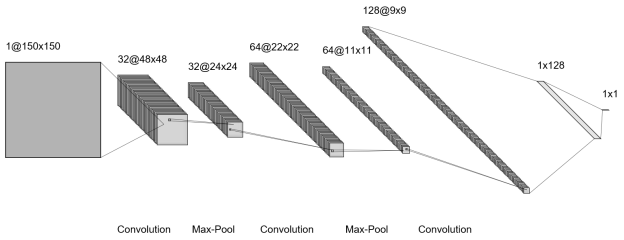

Figure 8: A visual illustration of the final ANN model architecture.

The first rectangle on the left displays the input and each individual pixel. All other layers correspond to layers from the final model. The input is fed through the model pipeline consisting of a 3×3 convolution with 32 filters, followed by a 2×2 max pooling layer. 25% of the outputs from this max pooling layer are dropped out, and the rest are fed into another 3×3 convolution with 64 filters. This is followed by another 2×2 max pooling layer, and then a dropout of 25%. A final 3×3 convolution with 128 filters is then performed. 40% of the outputs of the final convolution are dropped out, and then fed to a fully connected layer of 128 neurons. 30% of this layer's output is then dropped out and the rest is connected to the final output layer which consists of a single neuron representing the predicted price.

Each convolutional layer in the network consists of n filter tensors whose dimensions are $w \times h \times d$ tensors. In a convolutional layer, d is always equal to the number of channels in the input, and w and h are parameters. Each of the filters then "slides" along the 3 dimensional input and produces a 3 dimensional output according to Equation 1 below.

$$C[x,y,z] = \sum_{i=0}^{w-1} \sum_{j=0}^{h-1} \sum_{k=0}^{d-1} W[i,j,k] \cdot P[i+x, j+y, k] \quad (1)$$

Where $C[x,y,z]$ is the output of the convolutional layer at coordinates x,y,z, $W[i,j,k]$ is a trainable network weight of the kernel, and $P[i+x, j+y, k]$ is the input value at the specified coordinates. With $x \in \mathbb{N}[0, 1+P_w-w]$, $y \in \mathbb{N}[0, 1+P_h-h]$, and $z \in \mathbb{N}[0, d)$, and P_w, P_h are the width and height of the input.

The output neuron of the network uses a linear activation function, as it's purpose is to output the final predicted MSRP of the image. All other neurons in the model use a rectified linear unit activation function (Nair & Hinton, 2010) which is described by equation 2 below.

$$\mathrm{ReLU}(x) = \max(x, 0) \quad (2)$$

The optimization algorithm that was used to train the model was *Adam*. Adam is currently a popular choice, and it has shown to be effective in many applications (Kingma & Ba, 2014). Adam improves upon the classical stochastic gradient descent algorithm by utilizing a adaptive learning rate as opposed to a fixed learning rate. The algorithm attempts to optimize the magnitude of the steps taken during gradient descent by estimating the moment of the gradient. This makes Adam excel at optimizing on problems with sparse or noisy gradients.

The loss function used to evaluate and train the model was the *mean absolute percentage error*. It is described by equation 3 below.

$$M = \frac{1}{n} \sum_{i}^{n} \left| \frac{a_i - p_i}{a_i} \right| \quad (3)$$

Where n is the number of samples in one batch, a_i is the actual MSRP for sample i, and p_i is the predicted MSRP for sample i.

Evaluation

The final performance of the trained neural neural network is evaluated by running on all of the images that were set aside in the validation set earlier. The mean absolute percentage error of the predicted prices is then calculated for the last time, and gives an indication on how the model performs on data that it was not trained on.

Note that the lock-box approach was not used for this project due to the relatively small size of our data set. This suggests that the model may still overfit the images in our dataset, as all hyper-parameter tuning was done on this dataset.

Results

The multiple linear regression was found to significantly predict car prices based on the make, manufacture year, horsepower, engine type, dimensions, and body style, $F(51, 12389) = 1471$, $p < 0.05$, $R^2_{adj} = 0.8577$. All predictors were found to be significant at $p < 0.05$. The mean absolute percentage error for the predictions made from the linear regression model was found to be 17.6%. These results set a high point of comparison for the neural network image regression model.

After training for 500 epochs, the initial neural network model obtained a training loss of 14.8 and a validation loss of

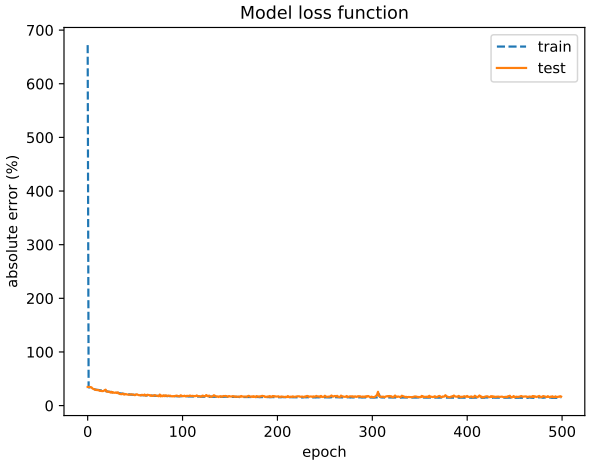

Figure 9: The loss function of the neural network model during training on the final dataset with a learning rate of 0.002. The loss function is plotted against the training epoch number. The model achieved a validation loss of 16.7% in the last epoch.

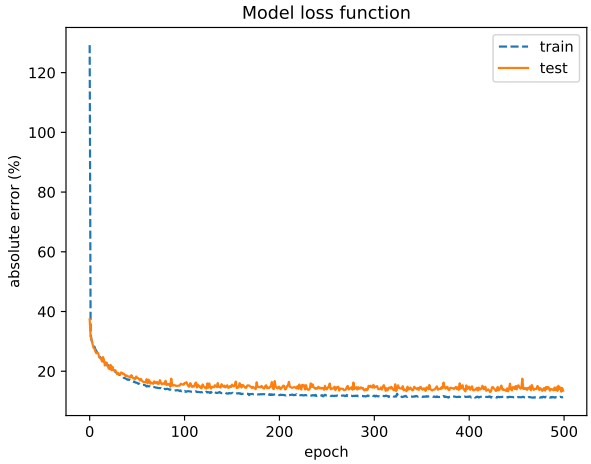

Figure 10: The loss function of the neural network model during training on the final dataset with a learning rate of 0.001. The loss function is plotted against the training epoch number. The model achieved a validation loss of 13.5% in the last epoch.

16.7. This was a learning rate of 0.002. Since the loss function used for the neural network model was the mean absolute percentage error, the trained neural network model obtained a mean absolute percentage error of 16.7%. Figure 9 shows how the loss of the initial model varied during training.

After tuning the hyper-parameters a more optimal learning rate for the model was found to be 0.001. After training the model for 500 epochs with a learning rate of 0.001, the model obtained a training loss of 11.3 and a validation loss of 13.5. Therefore, this final trained neural network model obtained a mean absolute percentage error of 13.5%. Figure 10 shows how the loss of final the model varied during training.

From these results we can see that the mean absolute percentage error was 4.1% lower when using the fully trained neural network model compared to using a multiple-linear regression model to predict the MSRP of cars. So we can conclude that indeed the neural network model achieves a higher performance of the two when predicting car prices.

While this might not seem like a large difference in performance we have to keep in mind that the linear regression model had direct access to car metrics such as the manufacturer, horsepower, body type, etc. while the neural network model predictions were based solely off of an image of the car. In a typical use case, access to the complete car metrics might be restricted or unavailable, while it is easy to take a picture of a car.

Discussion

Our ANN model was able to successfully predict the MSRP of cars based on images. The mean absolute percentage error of the model on the validation data was 13.5%, while the multiple-linear regression model obtained a mean absolute percentage error of 17.6%. Therefore, the ANN model achieved a 4.1% prediction accuracy improvement over the regression model. This is a notable improvement considering that the ANN model predicts prices solely based on images, and is therefore more flexible than the regression model which uses many more features.

Even though multiple models were tested and trained in search for better results, the performance of the final model seems comparable to other well known Convolutional Neural Networks presented in the survey by Khan, Sohail, Zahoora, and Qureshi (2019). Convolutional neural networks such as AlexNet typically have error rates in the 10% – 20% range. Therefore, we can conclude that the model presented here has relatively good performance.

These results were obtained when the model was trained on 13,000 scaled-down 150×150 resolution images, for 500 epochs, with a learning rate of 0.001, and with a batch-size of 64. Other values of these hyper-parameters were tested, however they either decreased the performance of the model, or they took excessive amounts of processing time and memory to run for more than a couple of epochs.

Additionally, various neural network architectures were tried before the one presented above was created. All of these other architectures had worse performance. In particular, quick test

runs on models with larger or smaller models either failed to converge on a good accuracy in a reasonable amount of time, or were not sufficiently complex to accurately extract the necessary features from the dataset.

Overall, the process of creating the ANN model was rather frustrating, as online information about the subject is vague with statements like "the perfect neural network architecture is not too big, not too small, but just right" (Shukla, 2019). Much of this online information led us down what in hindsight we know to be wrong paths. The final architecture was created based on what feels like mostly random guesswork, rather than reasoned decisions.

Some of the prototype models that were tested were based off of large pre-trained models such as ResNET (He, Zhang, Ren, & Sun, 2016), as many online articles mentioned that this would be a good approach to take for tasks like this (Shukla, 2019). However, we were unable to make these models work. They were large and cumbersome and did not provide any increase in performance over the custom trained network that we tried.

Through the process of hyper-parameter tuning, we experienced plenty of disappointing moments. By trying to optimize what we initially started with, we ended up with far worse models. Something as minor as increasing the learning rate from 0.001 to 0.002 had a drastic impact on performance despite all of our intuition. As a result, this was a large point of frustration during development. Whenever we were sure that changing a hyper-parameter would improve the model, the exact opposite would happen. This resulted in countless hours spent re-training and optimising the model.

There is no perfect solution to any problem, and our model is no exception. There are plenty of improvements to be made. One such improvement would be expanding the dataset used to train the model. As presented here the final dataset used contained roughly 13,000 images. However, it is well known that ANNs, and especially convolutional neural networks, benefit greatly from as large a dataset as possible.

One simple method that could artificially increase the size of the dataset would be to use data augmentation methods such as scaling, rotating, translating, and flipping the car images. This technique was shown to be successful in other applications (Taylor & Nitschke, 2017), and it was not fully implemented in this project due to time constraints.

Additionally, it would likely be beneficial to include images of cars manufactured before 2010, as our current dataset is strongly skewed towards newer car models. Including more older cars could increase the robustness and generalizability of our model.

The final dataset used to train the model consisted only of car images on a purely white background. This limits the usefulness of the model, as it can only be reliably used to predict the prices from car images without a background, in spite of the fact that in most real use-cases car images would indeed have a background. This could be remedied either by adapting the model, and retraining it on a dataset that includes images with a background, or by implementing a data pre-processing step that can effectively remove the backgrounds from car images before they are passed through the model.

Finally, the hyper-parameters used while training the final model were far from perfectly fine-tuned. Additional tuning could provide dramatic increases in final performance, as the model seems to be very sensitive to the hyper-parameter values. In particular the resolution of the input images seems to drastically impact model performance. In the final model, 150×150 images were used, however this could likely be further increased to 200×200, or even 320×320 which is the original resolution of the images in the dataset. This would, however, require drastically more powerful hardware than what was available at this time.

In particular it would appear from our limited experiments that the performance of the model improves with a lower learning rate, as we have demonstrated in Figure 9 and Figure 10. The model also seems to have a higher performance when the resolution of the input images is larger, although this significantly prolongs the training time.

In spite of all of the frustrations and confusion, we have learned a lot over the course of this project. We learned to better understand Neural Networks and how to apply them to a particular problem. We were introduced to convolutional neural networks which seem incredibly useful for image related tasks.

But most importantly we have learned and experienced the value of having patience when designing a neural network. Only patience can help with persevering through countless failed experimental architectures, misinformation on the internet, and confusing hyper-parameter tuning.

Overall it was bitter-sweet experience through which we learned a lot.

References

Burger, W., & Burge, M. J. (2010). *Principles of digital image processing: core algorithms*. Springer Science & Business Media.

Gervais, N. (2020). *The Car Connection Picture Dataset*. Retrieved 2020-07-07, from https://github.com/nicolas-gervais/predicting-car-price-from-scraped-data/tree/master/picture-scraper

Häubl, G., & Trifts, V. (2000). Consumer decision making in online shopping environments: The effects of interactive decision aids. *Marketing science*, *19*(1), 4–21.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 770–778).

Hijazi, S., Kumar, R., & Rowen, C. (2015). Using convolutional neural networks for image recognition. *Cadence Design Systems Inc.: San Jose, CA, USA*, 1–12.

Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2019). A survey of the recent architectures of deep convolutional neural networks. *arXiv preprint arXiv:1901.06032*.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097–1105).

Marshall, P., Sor, R., & McKay, J. (2000). An industry case study of the impacts of electronic commerce on car dealerships in western australia. *J. Electron. Commerce Res.*, *1*(1), 1–12.

Miller, B. R., & Masters, G. C. (1973). A short-run price prediction model for eggs. *American Journal of Agricultural Economics*, *55*(3), 484–489.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Icml.*

photoscissors.com. (2020). *PhotoScissors automatically remove background from your photos online.* Retrieved 2020-07-07, from https://photoscissors.com/

Reynolds, J. (2000). ecommerce: a critical review. *International Journal of Retail & Distribution Management*, *28*(10), 417–444.

Rusmevichientong, P., Salisbury, J. A., Truss, L. T., Van Roy, B., & Glynn, P. W. (2006). Opportunities and challenges in using online preference data for vehicle pricing: A case study at general motors. *Journal of Revenue and Pricing Management*, *5*(1), 45–61.

Shukla, L. (2019). *Designing your neural networks.* Retrieved 2020-07-07, from https://towardsdatascience.com/designing-your-neural-networks-a5e4617027ed

Smith, A. D. (2009). Online accessibility concerns in shaping consumer relationships in the automotive industry. *Online Information Review*.

Sola, J., & Sevilla, J. (1997, 07). Importance of input data normalization for the application of neural networks to complex industrial problems. *Nuclear Science, IEEE Transactions on*, *44*, 1464 - 1468. doi: 10.1109/23.589532

Taylor, L., & Nitschke, G. (2017). Improving deep learning using generic data augmentation. *arXiv preprint arXiv:1708.06020*.

White, H. (1988). Economic prediction using neural networks: The case of ibm daily stock returns. In *Icnn* (Vol. 2, pp. 451–458).