Bajari, Patrick L. et al.

Working Paper

Hedonic prices and quality adjusted price indices powered by AI

cemmap working paper, No. CWP04/21

Provided in Cooperation with:
Institute for Fiscal Studies (IFS), London

Suggested Citation: Bajari, Patrick L. et al. (2021) : Hedonic prices and quality adjusted price indices powered by AI, cemmap working paper, No. CWP04/21, Centre for Microdata Methods and Practice (cemmap), London, http://dx.doi.org/10.47004/wp.cem.2021.0421

This Version is available at:
http://hdl.handle.net/10419/241940

cemmap
centre for microdata methods and practice

Hedonic prices and quality adjusted price indices powered by AI

Patrick Bajari
Zhihao Cen
Victor Chernozhukov
Manoj Manukonda
Jin Wang
Ramon Huerta
Junbo Li
Ling Leng
George Monokroussos
Suhas Vijaykunar
Shan Wan

The Institute for Fiscal Studies
Department of Economics, UCL

cemmap working paper CWP04/21

UKRI Economic and Social Research Council

HEDONIC PRICES AND QUALITY ADJUSTED PRICE INDICES POWERED BY AI

P. BAJARI*, Z. CEN*, V. CHERNOZHUKOV*, M. MANUKONDA*, J. WANG* AND R. HUERTA, J. LI, L.LENG, G. MONOKROUSSOS, S. VIJAYKUMAR, S. WAN

Abstract. We develop empirical models of hedonic prices and derive hedonic indices for measuring changes in customer welfare based upon deep learning. We first generate abstract product attributes, or "features," from text descriptions and images using deep neural networks, and then use these attributes to estimate the hedonic price function. Specifically, we convert textual information about the product to numeric product features using the ELMO or BERT language models, trained or fine-tuned using Amazon's product descriptions. We convert the product image to numerical product features by a pre-trained ResNet50 image model. To produce the estimated hedonic price function, we use a multi-task neural network again, trained to predict the price of a product simultaneously in all time periods. We apply the models to Amazon's data for first-party apparel sales to estimate hedonic prices. The resulting models have high predictive accuracy, with R^2 ranging from 80% to 90%. We also construct hedonic price indices: over the period 2013-2017 the hedonic Fisher price index decreased, providing improvement in customer welfare.

Date: December, 2018. This version: February 20, 2021.

P. Bajari, Z. Cen, V. Chernozhukov, M. Manukonda, and J. Wang (in alphabetical order) are the principal authors who contributed to this research paper with all of its scientific stages of development. R. Huerta and G. Monokrousos were also principal contributors to the first version of this paper that relied on Word2Vec and Alexnet embeddings, using random forest to learn the hedonic price. The current version of the paper relies on BERT and ResNet embeddings with multi-task deep neural nets to learn the hedonic price. We would like to thank seminar participants at the ASSA 2020 and FESAC 2018 meetings, where preliminary versions of this paper were presented.

1. Introduction

When prices change, a consumer may become better off or worse off. Economists measure how consumers are affected by price changes in the economic environment, and have developed several tools to do this. One of the most commonly used are price indices, such as the Laspeyeres or Paasch indices: these measure changes in the cost of a standardized basket of products between two periods, holding the basket to be equal to the set of products purchased in the initial period (Laspeyeres) or the final period (Paasch);[1] the two are often combined into what is known as the ideal or the Fisher price index. The Fisher price index is known to provide accurate approximation for true cost of living when the change in prices is small.[2]

A common problem with these indices is product entry and exit, where the previous period's prices are not available for new products and vice-versa. Hence, economists often compute so-called 'matched indices'—indices restricted to the set of products that are bought and sold in both periods. This in itself creates major, well-documented biases in the resulting indices. To overcome this issue, several prominent economists have recommended using chaining: computing the indices over higher frequency and then compounding monthly inflation rates to get yearly rates. This ameliorates the turnover problem but does not completely solve it.

Hedonic prices were introduced by Andrew Court (1939) and Zvi Griliches (1961) as a way of addressing the entry/exit problem in price indices. Hedonic price functions summarize the relationship between the prices and the characteristics of goods sold in differentiated products. Since newer products often have better characteristics, the difference between the prices of the newer and the older products is not

[1]These quantities bound other measures of inflation/deflation based on the expenditure function under certain assumptions; we refer to Diwert's review for details.

[2]It is an exact cost of living for quadratic utility or expenditure functions, and provides second-order accurate approximation for any smooth utility or expenditure function under small changes in prices; see, e.g, Diewert (1977). Price indices with this property are called superlative indices, with another prominent example being the Tornqvist index. In our analysis, the results using Tornqvist index are very similar to the results obtained using the Fisher index.

entirely attributable to inflation. Court and Griliches suggested to measure inflation/deflation by looking at price changes for products holding product characteristics fixed. This paradigm suggests an entirely different way of computing inflation and, in fact, leads to a consumer theory based on utility over product characteristics rather than over products themselves (Lancaster, 1966; McFadden, 1974). Under this theory, products' prices are entirely determined by its characteristics. Economists have developed a number of results on the existence and properties of these functions under various types of assumptions (see, e.g., Berry, Levinsohn, Pakes, Ekeland (1995, 2004); Ekeland, Heckman, and Nesheim (2004), Bajari and Benkard (2005); Chiappori, McCann, and Nesheim (2010), Chernozhukov, Galichon, Henry, Pass (2017)).

Provided that these models are good approximations, hedonic theory suggests that we can learn hedonic price functions by estimating regression functions that explain observed prices in terms of products characteristics. If we successfully represent the product j observed by the customer and producer in terms of its characteristics X_j, then we can approximate the price at a given point in time by a function of X_j. The characteristics of the product observable by the customer are illustrated in Figure 1, where we see the product title, product text description, and the images for the product. Our goal is to represent this information using a numerical vector X_j of moderately high dimension (in our case it has 2000 entries) which can be used to accurately predict prices. Moreover, the representation needs to be algorithmic and scalable.

The success of this approach depends on the existence of parsimonious structures behind images and text. Traditionally, analysts relied on human experts to represent the key features of products to find a low dimensional numerical representation X_j. Successful experts did manage to produce low-dimensional representations for certain groups products that proved to be successful in building hedonic models (e.g., Pakes (1993) reports very high accuracy for predicting computer prices). However, this approach is not scalable to many types of products and is prone to judgment biases. These issues raise important questions: When human experts succeed, what is the underlying fundamental reason? Can this success be replicated by machine learning methods, and can these methods deliver scalable inference?

We believe that humans can easily summarize the red dress image in Figure 1 and understand the key features of the product text description, even though the original representation of this information lives in an extremely high-dimensional space (i.e. millions of dimensions). Indeed, image consists of millions of pixels (three layers of 640 x 480 pixels encoding the blue, red, and green color channels), words live in a tens thousand-dimensional dictionary,[3] and sentences in the product description are created by sequences of words whose length ranges in the hundreds. However, we believe that the images and sentences can be losslessly represented in a much lower-dimensional space, and we call this phenomenon 'structured sparsity.' Somehow, human intelligence is able to exploit this structured sparsity and process information effectively (perhaps exploiting geometry of shapes in images, relative simplicity of color schemes and shade patterns, similarity of many words in the dictionary, and context-specific meaning of words). The field of artificial intelligence (AI) developed neural networks in an effort to mimic human intelligence in many information-processing tasks. These models do create parsimonious structures from high-dimensional inputs, often surpassing human ability in these tasks. Going forward, we will employ state-of-the-art solutions from these fields to the problem of hedonic modeling.

In our approach we generate product attributes or features (from text and images) by state-of-the-art deep learning models and then use them to estimate the hedonic price function. Specifically, we convert text information about the product to numeric features (called embeddings) using the ELMO or BERT deep learning models (Peters et al. 2018, Devlin et al. 2018), fine-tuned on Amazon's product descriptions. We use a pre-trained ResNet50 model (He et al., 2016) to produce the embeddings for the images of the product. This is the first step of the process: the models of the first step are trained on tasks unrelated to predicting prices (e.g., tasks of image classification or predicting a missing word in a sentence), and embeddings are extracted as the hidden (often, penultimate) level of the neural network. In the second step of the process, we produce the estimated hedonic price function, using neural networks with the multi-task structure. The models of the second step are trained to predict prices simultaneously in all time period. All the ingredients to the method rely on

[3]See, e.g., `http://testyourvocab.com/blog/2013-05-10-Summary-of-results`

FIGURE 1. An example of product characteristics for a product sold in the Amazon store

publicly available, open-source software components. We perform data preparation in Apache Spark using the Elastic MapReduce service from Amazon Web Services (AWS), and use an AWS SageMaker multi-GPU machine learning environment to train the neural network models.

We apply these models to Amazon's data for first-party[4] apparel sales to estimate the hedonic prices. The resulting hedonic models have a high predictive accuracy, with R^2 in the hold-out sample ranging from 80 to 90%. Therefore, our approach is able to attribute up to 90% of variation in price to variation in the product embeddings that encode the product attributes. We find this performance remarkable for two reasons:

(1) The production of hedonic prices is completely automatic and scalable, without relying on any human-based feature extraction.

(2) The performance suggests that the hedonic price models from economics provide a good, first-order approximation to prices.

[4]See, e.g, `https://feedvisor.com/university/amazon-1p-vs-3p/` for the definition of the terms, such as first-party and third-party.

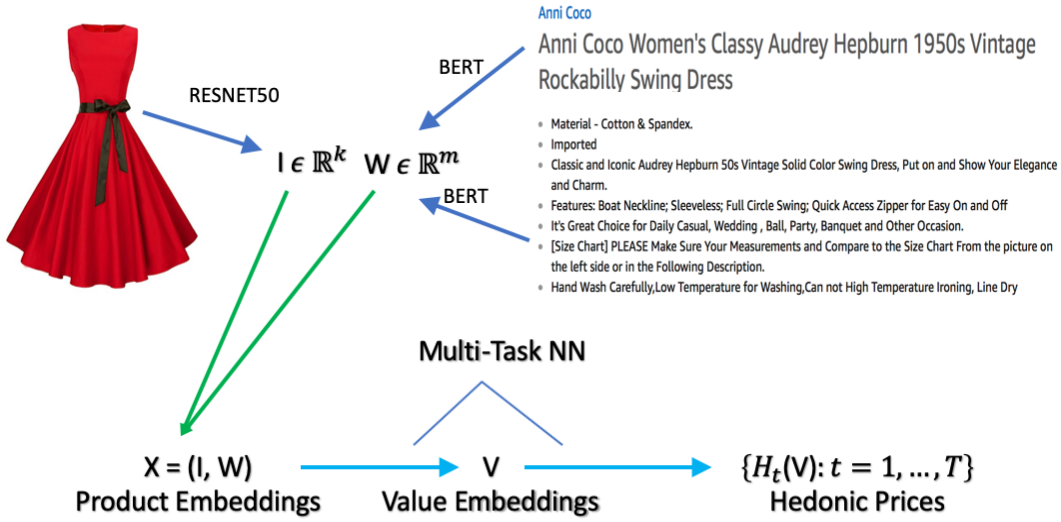

FIGURE 2. Our method for generating hedonic price: The input consists of images and unstructured text data. The first step of the process creates the moderately high-dimensional numerical embeddings I and W for images and text data via state-of-the art deep learning methods, such as ResNet50 and BERT. The second step of the process takes as input $X = (I, W)$ and creates predictions for hedonic prices $H_t(X)$ using deep learning methods with a multi-task structure. The models of the first step are trained on tasks unrelated to predicting prices (e.g., image classification or word prediction), where embeddings are extracted as hidden layers of the neural networks. The models of the second step are trained by price prediction tasks. Our multitask model creates an intermediate lower dimensional embedding $V = V(X)$, called value embedding and then predicts the final prices in all time periods $\{H_t(V), t = 1, ..., T\}$ using linear or parametric linear functional forms, making it easy to perform inference on the last step, using hold-out samples. Some variations of the method include fine-tuning the embeddings produced by the first step to perform well for price prediction tasks (i.e. optimizing the embedding parameters so as to minimize price prediction loss).

We then proceed to construct the hedonic Fisher price indices (HFPI) over 2013-2017, constructing monthly-chained, yearly-chained, and the combined GEKS-type HFPI (GHFPI).[5] We focus discussions on GHFPI and simply refer to it as the hedonic index, unless stated otherwise. We also compare the hedonic index with the matched (repeated sale) Fisher index, posted price Jevons index (the geometric mean of log of price relatives, chained at daily frequency), the BLS Urban CPI index for apparel (CPI), constructed by the Bureau of Labor Statistics, and the Adobe Digital Price Index (DPI) for apparel, constructed by Goolsbee and Klenow (2018) using the Adobe Analytics data.

All indices suggest the apparel price level declines over this period. The annual rate of inflation estimated by GHFPI is -3.16%, by the Jevons index -2.82%, by the matched Fisher index -3.46%. In comparison, the annual rate of inflation estimated by DPI is -1.3%, and by CPI is -.07%. Part of the difference with CPI could be attributed to the limited ability of CPI to address quality change and substitution (see Moulton, 2018 for detailed analysis), specifics of product categorization in the apparel segment, and difference in the composition of baskets that consumers purchase via Amazon.

In comparison to the hedonic indices, the matched Fisher index or Adobe's DPI do not incorporate quality change. Moreover, we cannot use them for chaining over longer time periods due to the high product turnover. (The less frequent chaining is desirable for addressing the "chain drift" – the systematic accumulation of errors due to frequent compounding). In comparison to the hedonic indices, the Jevons index (compounded geometric means of posted price relatives) does not incorporate quantity weighting. Jevons index declines slightly less than GHFPI (reflecting its limitations in capturing substitutions of customers for less expensive items).

We view our paper as developing major modernization of both hedonic price models as well as their application to large-scale data. In this way we contribute to the literature in empirical microeconomics dedicated to hedonic price models and their

[5]The version of the GEKS index we use is a geometric mean of two HFPIs: one chained month-over-month and another chained year-over-year. We use such GEKS-type index to mitigate the "chain drift" problem, which refers to the accumulation of errors under frequent compounding.

Index	CPI	BPP	ADPI	FHPI
Prices	Yes	Yes	Yes	Yes
Revenue Shares for Product Groups	Yes	Unknown	Yes	Yes
Quantities	No	No	Yes	Yes
Quality Adjustment	Yes	No	No	Yes

TABLE 1. Some properties of the CPI, BPP, ADPI, and FHPI

uses in measuring inflation. We are not aware of any prior work in this area which developed a similar modern large-scale hedonic price models based on neural network embeddings for product text and image descriptions. In addition, our data is unique in that it covers the universe of apparel products that has been transacted in the Amazon store by the first party. From these models and data, we generated interesting findings: showing the usefulness of hedonic models in characterizing prices and documenting the decline in quality-adjusted Fisher price index in Apparel. To the best of our knowledge, there are very few related studies: An independent and contemporaneous work by Zeng (2020) develops a related approach to hedonic prices using scanner data, but based on random forest methods and not using the image and text embeddings. An independent and contemporaneous work by Han et al (2020) explores the use of image embeddings to characterize fonts as products, and analyzes the effect of merger on product differentiation decisions (in terms of design) of font producers.

This paper contributes to a fast-growing literature on using alternative, modern data and techniques to measure inflation and other aggregate quantities. The Billion Prices Project at MIT constructs Jevons indices (BPP) using retail price data sets using web-scraped and directly provided retailer's data. The advantages of such data sets include real-time availability at daily frequencies, low collection costs, large product counts, and uncensored price spells. Consequently, BPP constructed using modern real-time data can serve as useful benchmarks for official government statistics. For instance, Cavallo (2013) and Cavallo and Rigobon (2016) study several countries and establish that inflation measures constructed using such online data can be quite different at times from official government statistics on the CPI (the

most extreme example being the case of Argentina). A shortcoming of this approach is that it does not incorporate quantity information. Goolsbee and Klenow (2018) constructed matched Tornqvist indices, called Digital Price Index (DPI), using Adobe Analytics data from e-commerce clients of Adobe (which, most notably, include quantities, in addition to prices). This approach does not have the shortcomings of the BPP index, and is able to account for substitution resulting from consumers minimizing costs. They find that for the US online DPI inflation is substantially lower (for the period 2014-2017) than the official CPI inflation for the categories they study; this is analogous to our findings for Apparel.

In related work using online price data Cavallo (2017) finds that online prices are identical to their offline counterparts about 70% of the time (on average across countries), thereby justifying the treatment of both on-line and off-line retail sector as the single sector. Gorodnichenko and Talavera (2017), Cavallo (2018a) and Gorodnichenko et al. (2018) find that online prices change more frequently than offline prices, thereby responding to competition more promptly, and also they exhibit stronger pass-through in response to nominal-exchange-rate movements than prices found in official CPI data. Results such as these have major implications for important areas of macro and international economics such as the price stickiness literature and the law of one price, as Cavallo (2018b), Cavallo and Rigobon (2016), Gorodnichenko and Talavera (2017), Gorodnichenko et al. (2018) illustrate.

We organize the rest of the paper as follows. In Section 2, we define the hedonic price models and price indices. In Section 3, we discuss estimation of of the hedonic price functions via NNs. In Section 4 we describe feature engineering – the process of representing the product information via Neural Network (NN) embeddings. In Section 5, we examine the empirical performance of the hedonic price functions generated by our method. In Section 6, we construct the hedonic price indices and analyze them and other indices.

Notation. We use capital letters as W as random vectors and w the values they take; we use W to denote matrices. Functions are denoted by arrows $w \mapsto f(w)$ or simply f. Greek symbols, with the exception of ϵ, denote parameter values.

2. Hedonic Prices and Hedonic Price Indices

2.1. **The Hedonic Price Model.** We denote the product by index i and time period (month) by t. An empirical hedonic model is the predictive model for price given the product features:

$$P_{it} = H_{it} + \epsilon_{it} = h_t(X_{it}) + \epsilon_{it}, \quad \mathrm{E}[\epsilon_{it} \mid X_{it}] = 0, \tag{1}$$

where P_{it} is the price of product i at time t, X_{it} are the product features, and the price function $x \mapsto h_t(x)$ can change from period to period, reflecting the fact that product attributes/features may be valued differently in different periods. For our purposes, the advantage of these models is that they allow us to compare new goods to old rather directly; we simply compare the value consumers attach to the characteristics of the old good to those of the new.

Most of the product attributes X_{it} will remain time-invariant, but some may change over time. We shall use the data from time period t to estimate the function h_t using the modern nonlinear regression methods, such as deep neural network methods. We shall contrast this approach with classical linear regression methods as well as other modern regression methods, such as random forest. The key component of our approach is the generation of product features X_{it} using NN embeddings of text and image information about the product. Thus, X_{it} consists of text embedding features W_{it}, constructed by converting the title and product description into numeric vectors, and image embedding features I_{it}, constructed by converting the product image into numeric vectors:

$$X_{it} = (W_{it}', I_{it}')'. \tag{2}$$

These embedding features are generated respectively by applying the BERT and ResNet50 mappings, as explained in detail in the next section.

There is a substantial body of research on economics and empirics of hedonic price models. On the theory side, economists have developed a theory of demand in terms of product characteristics rather than demand (Lancaster, 1966; McFadden, 1974); they also established various existence results for the hedonic price functions under various assumptions (Berry, Levinsohn, Pakes, Ekeland (1995, 2004); Ekeland,

Heckman, and Nesheim (2004), Bajari and Benkard (2005); Chiappori, McCann, and Nesheim (2010), Chernozhukov, Galichon, Henry, Pass (2017)); they developed the use of hedonic prices for bounding changes in consumer's surplus and welfare (Bajari and Benkard (2005)). On the empirical side, economists have estimated a variety of hedonic price models and linked them to the consumer's utility (and marginal willingness to pay for certain characteristics, see Nesheim (2007)), and also have used them for valuation of non-tradable goods (for example, valuation of the effects of improving ecological environment on housing prices, e.g. Stock (1989), or the effect of environmental regulation on costs of automobiles, Berry et al 1996). Our main use of hedonic prices is to estimate the rates of deflation (or inflation) for the aggregate baskets of apparel products that customers buy at Amazon, following the prior work on using hedonics (e.g., Griliches, 1961, Pakes, 2005), but with the major deviation being the use of product features engineered via deep learning, instead of human engineering of features, and price prediction being done with deep learning rather than classical regression methods.

The literature typically specifies three building blocks of theoretical hedonic models: utility functions defined directly on the characteristics of products (rather than on products per se) and customer's characteristics; cost functions which typically include characteristics of the good and of producers; and an equilibrium assumption (or existence is shown as a part of analysis). This determines prices (and quantities) given demand and costs, and establishes existence of the hedonic price function

$$(x, u) \mapsto H^\star(x, u)$$

as a function of product attributes (x, u), which are all attributes observable by the consumer, and u is an attribute that is not observable by the modeler. Price functions give us information about customer preferences. For example, when the customer's utility is given by:

$$V(x, u, p) = V_0(x, u) - p$$

where p is price of the product to be paid by the customer, the first order conditions (for continuously varying attributes) for the utility maximization problem $\max_{(x,u)} V_0(x, u) - H(x, u)$ is given by:

$$\partial_{x_k} V_0(X, U) = \partial_{x_k} H^\star(X, U),$$

where $\partial_{x_k} = \partial/\partial x_k$, where x_k refers to the k-th component of the vector x. Therefore in this model a standard argument for identification of average derivative of structural function gives

$$\mathrm{E}[\partial_{x_k} H^\star(X_j, U_j)|X_j] = \partial_{x_k} h_t(X_j),$$

that is the average marginal willingness to pay for a given characteristic is equal to the average derivative of the hedonic price map.

Moreover, under parametric form of utility, preference parameters of a consumer can be recovered from the first-order conditions, provided that $(x, u) \mapsto H^\star(x, u)$ is additively separable in (x, u), so that $\partial_{x_k} H^\star(x, u) = \partial_{x_k} h_t(x)$, does not depend on u, or provided we can identify $H^\star$ and the unobservable U by other means (for example, by making quantile or multivariate-quantile type assumptions on the way U appears in $H^\star$.) For example, following Bajari and Benkard (2005), if the utility is Cobb-Douglas over observed characteristics, $V_0(x, u, p) = \sum_{k=1}^K \alpha_k \log(x^\star_{jk}) + \beta g(u) - p$, then under additive separability, $H^\star(X, U) = H^\star_0(X) + U$, we have $\alpha_k = \partial_{x_k} h_t(X_j) X_{jk}$ for the consumer who has purchased product j. Hence distributions of tastes parameters α for consumers can be recovered under such modeling approaches.[6] We will not take this approach, but rather adopt a more pragmatic approach of using hedonic prices inside the price indices to measure changes in price levels, following accepted practice in applied price research and in the work of statistical agencies.

2.2. **Price Indices: Hedonic vs Matched.** When the economic environment changes a consumer may be made better off or worse off. Economists often want to measure how consumers are affected by changes in the economic environment, and have developed several types of price indices to evaluate the change.

We focus on hedonic price indices and contrast them with matched (repeated sales) prices indices. The matched price index tracks changes in the price of a basket of products that are sold in both the base period and later time periods. While the matched price method is subject to selection bias (due product entry and exit), it ensures the index tracks goods from a common pool of products. A major

[6]When the characteristics are discrete bounds on individual preference parameters can be constructed and used in policy analysis, see Bajari and Benkard (2005).

shortcoming of this method is that the common pool of products across time can be small and non-representative, which is true in our case. This property will manifest itself empirically.

The hedonic price index replaces transaction prices with predicted values using a rich set of product characteristics (obtained using a combination of AI and ML methods). In principle, the hedonic approach captures changes over time in the value consumers place on product attributes. Hedonic approaches are especially helpful for predicting prices of new goods and dealing with the entry/exit selection bias when product prices are undefined. This is especially relevant in our case, where we observe a very high turnover of the products.

We consider three types of each price index:

- The Laspeyres (L) type, which uses base period quantities for weighting the prices;
- The Paasche (P) type, which uses current period quantities for weighing the prices;
- The Fisher (F) type, which uses the geometric mean of L and P indices.

One defines the L and P -type matched indices as measures of the total rate of price change of a basket of matching products from the current period t with a previous period $t-\ell$:

$$R_{t,\ell}^{P,M} = \frac{\sum_{i\in\mathcal{C}_t\cap\mathcal{C}_{t-\ell}} P_{it}Q_{it}}{\sum_{i\in\mathcal{C}_t\cap\mathcal{C}_{t-\ell}} P_{j(t-\ell)}Q_{it}}; \quad R_{t,\ell}^{L,H} = \frac{\sum_{i\in\mathcal{C}_t\cap\mathcal{C}_{t-\ell}} P_{it}Q_{i(t-\ell)}}{\sum_{i\in\mathcal{C}_t\cap\mathcal{C}_{t-\ell}} P_{i(t-\ell)}Q_{i(t-\ell)}};$$

and the F-type index takes the form

$$R_{t,\ell}^{F,M} = \sqrt{R_{t,\ell}^{P,M}\cdot R_{t,\ell}^{L,M}},$$

where Q_{it} is the quantity of the product i sold in month t, P_{it} is the average sales price for product i at time t, $\mathcal{C}_t$ is the set of all products with transactions at time t, $\mathcal{C}_t\cap\mathcal{C}_{t-\ell}$ is the match set, the set of all products with transactions at both time t and at time $t-\ell$.

Basic economics based on a representative consumer model suggests that matched indices should obey the order restriction: $R^{L,M} \geq R^{P,M}$. Aggregate quantities may not behave like those of a representative consumer, but the relation often holds empirically. Similar arguments can be made for hedonic indices. A way to aggregate the two indices is to use the Fisher's ideal index, which is a superlative index: it measures thee exact cost of living when the utility function is quadratic and provides a second-order approximation to the cost of living index at the given prices when the utility function is smooth (Diewert, 1977).

We define the L, P, and F-type hedonic indices similarly, as measures of the total rate of hedonic price change of a basket of product attributes from the current period t with the previous period $t-1$:

$$R_{t,\ell}^{P,H} = \frac{\sum_{i\in\mathcal{C}_t} H_{it}Q_{it}}{\sum_{i\in\mathcal{C}_t} H_{i(t-\ell)}Q_{it}}; \quad R_{t,\ell}^{L,H} = \frac{\sum_{i\in\mathcal{C}_{t-\ell}} H_{it}Q_{i(t-\ell)}}{\sum_{i\in\mathcal{C}_{t-\ell}} H_{i(t-\ell)}Q_{i(t-\ell)}}; R_t^{F,H} = \sqrt{R_t^{P,H}\cdot R_t^{L,H}}.$$

We note that the P index is defined over the sets of products $\mathcal{C}_t$ and the L index is defined over the set of products $\mathcal{C}_{t-\ell}$, which are supersets of the matching set $\mathcal{C}_t \cap \mathcal{C}_{t-\ell}$.

Using a generic chaining index, we measure the price changes up to time $t = t_0 + \ell m$, where t_0 and ℓ and m are positive integers, by taking the product:

$$R_{t,\ell}^{\bullet,\bullet,C} = \prod_{\bar{m}=1}^{m} R_{\bar{m},\ell}^{\bullet,\bullet} \quad \text{where} \quad R_{t_0,\ell}^{\bullet,\bullet,C} = 1.$$

For the hedonic index we shall use month-over-month chaining with $\ell = 1$ and year-over-year chaining with $\ell = 12$, getting two types of indices:

$$R_{t,1}^{F,H,C} \text{ and } R_{t,12}^{F,H,C},$$

where the first index captures month-over-month changes in prices, especially for non-seasonal apparel, and the second index gets year-over-year changes in products over month ℓ, better capturing price changes for seasonal apparel. The second index is less susceptible to the well-known chain-drift problem that is present that arises from accumulation of errors due to repeated compounding. A GEKS type index[7]

[7]Theoretical GEKS indices aggregated indices obtained at all chaining lags ℓ, but we shall limit ℓ to 1 and 12 to keep computational costs low.

Fisher Hedonic Price Index (GFHPI) aggregates the two types of indices by taking their geometric mean:

$$R_t^{GF,H} = \sqrt{R_{t,1}^{F,H,C} R_{t,12}^{F,H,C}}.$$

3. Price Prediction and Inference with Deep Neural Networks

3.1. **The Multi-Price Prediction Network.** Our model ingests high-dimensional text and image features as inputs, converts them into lower dimensional vector of value embeddings, using state-of-the-art deep learning methods, and outputs simultaneous predictions of price in all time periods.

The general nonlinear regression model we work with takes the form

$$Z_i = \begin{bmatrix} \text{Text}_i \\ \text{Image}_i \end{bmatrix} \overset{e}{\longmapsto} X_i \overset{g_1}{\longmapsto} E_i^{(1)} ... \overset{g_m}{\longmapsto} E_i^{(m)} =: V_i \overset{\theta'}{\longmapsto} \{H_{it}\}_{t=1}^T := \{\beta_t' V_i\}_{t=1}^T. \quad (3)$$

Here Z_i is the original input, which lies in a very high-dimensional space, is nonlinearly mapped into an embedding vector X_i which is of moderately high dimension (up to 5120 dimensions), which is further nonlinearly mapped into a lower dimension vector $E_i^{(1)}$, and so on, until it is nonlinearly mapped into the final hidden layer $V_i = E_i^{(m)}$, which is then linearly mapped to the final output, consisting of hedonic price H_{it} for product i in all time periods $t = 1, ..., T$.

The last hidden layer $V = E^{(m)}$ is called the *value embedding* in our context. The embeddings are moderately high-dimensional (up to 512 dimensions) summary of the product, derived from most common attributes, that directly determine the price of the predicted hedonic price of the product. Note that the embeddings V do not depend on time and represent the intrinsic, potentially valuable attributes of the product. However, the predicted price does depend on time t via the coefficient β_t, reflecting the fact that the different intrinsic attributes are valued differently across time.

The network mapping (3) makes use of repeated composition of nonlinear mappings of the form

$$g_\ell : v \longmapsto \{E_{k,\ell}(v)\}_{k=1}^{K_\ell} := \{\sigma_{k,\ell}(v'\alpha_{k,\ell})\}_{k=1}^{K_\ell}. \quad (4)$$

where the $E_{k,\ell}$'s are called neurons, and $\sigma_{k,\ell}$ is the activation function that can vary with the layer ℓ and can vary with k, from one neuron to another.[8] Standard examples include the sigmoid function: $\sigma(v) = 1/(1 + e^{-v})$, the rectified linear unit function (ReLU) $\sigma(v) = \max(0, v)$, or the linear function $\sigma(v) = v$. In a given layer some neurons can be generated linearly or nonlinearly. The use of nonlinear activation function has been shown to be extremely powerful tool for generating flexible functional forms, yielding both successful approximations in a wide range of empirical problem and backed by approximation theory. Good approximations can be achieved by both considering sufficiently many neurons and sufficiently many layers (e.g., Chen and White, 1999; Yarotsky, 2017; Schmidt-Hieber, 2020; Farrell et. al, 2021; Kidger and Lyons, 2020).

Our empirical model just up to $m = 3$ hidden layers, not counting the input. The dimensions of each layer are described in the appendix. The first layer generated using auxiliary text and image classification and prediction tasks, as described below.

The model can be trained by minimizing the loss function

$$\min_{\eta \in \mathcal{N}, \{\beta_t\}_{t=1}^T} \sum_t \sum_i (P_{it}^c - \beta_t' V_i(\eta))^2 Q_{it}, \tag{5}$$

where η denotes all of the parameters of the mapping

$$X_i \mapsto V_i(\eta)$$

and $\mathcal{N}$ represents the parameter space. Here we are weighting by the quantity Q_{it}. Regularization can be used to reduce time fluctuations of predicted price across time. This is done by adding the penalty function to the objective function:

$$\lambda \sum_i \sum_{t=1}^{T-1} |\beta_{t+1}' V_i(\eta) - \beta_t' V_i(\eta)|, \tag{6}$$

with penalty level λ chosen to yield good performance in validation samples.

Next we give overview of how the initial embedding is generated. A multilingual BERT model is used to convert text information and ResNet50 model is used to

[8]The standard architecture has activation function that does not vary with k, but some architectures such as ResNet50 discussed later can be viewed as having an activation function depending on k, with some neurons linearly activated and some nonlinearly.

convert image into a subvector of $E_i^{(1)}$. These models are trained on auxiliary predictions tasks with auxiliary output A_{T_i} for text and A_{I_i} for image, which can be illustrated diagrammatically as:

$$X_i = \begin{bmatrix} \text{Text}_i \\ \text{Image}_i \end{bmatrix} \overset{e}{\longmapsto} \begin{array}{c} A_{T_i} \\ \uparrow \\ W_i \\ X_i \\ \downarrow \\ A_{I_i} \end{array} =: E_i^{(1)} ... \longmapsto E_i^{(m)} := V_i \longmapsto \{\hat{P}_{it}^*\}_{t=1}^T, \tag{7}$$

The embeddings W_i and X_i forming $E_i^{(1)}$ are obtained by mapping them in an auxiliary outputs A_{Tj} and A_{Ij} that are scored on natural language processing tasks and image classification tasks respectively. This step uses data that is not related to prices, as we further describe below. The parameters of the mapping generating $E_i^{(1)}$ are considered as generally frozen in our analysis, although we've experimented with fine-tuning this parameters with the goal of improving price prediction performance.

The estimates are computed using sophisticated stochastic gradient descent algorithms, where sophistication is needed because this optimization is generally not a convex problem, making the computation difficult. In particular, for price prediction task we used the Adam algorithm (Kingma and Ba, 2014). The overall process has many tuning parameters, and in practice we chose them by cross-validation. The most important choices concerned the number of neurons and the number of neuron layers.

We visualize the process conceptually via Figure 3, where we have a regression problem, and the network depicts the process of taking raw regressors and transforming them into predicted values. In the first column on the left we see the inputs (features), and the second column the first layer of neurons. The neurons are connected to the inputs and the connections represent the coefficients. Finally, the last layers of neurons is combined to produce a vector of outputs. The coefficients β_m are shown by the connections between the last hidden layer of neurons and the output,

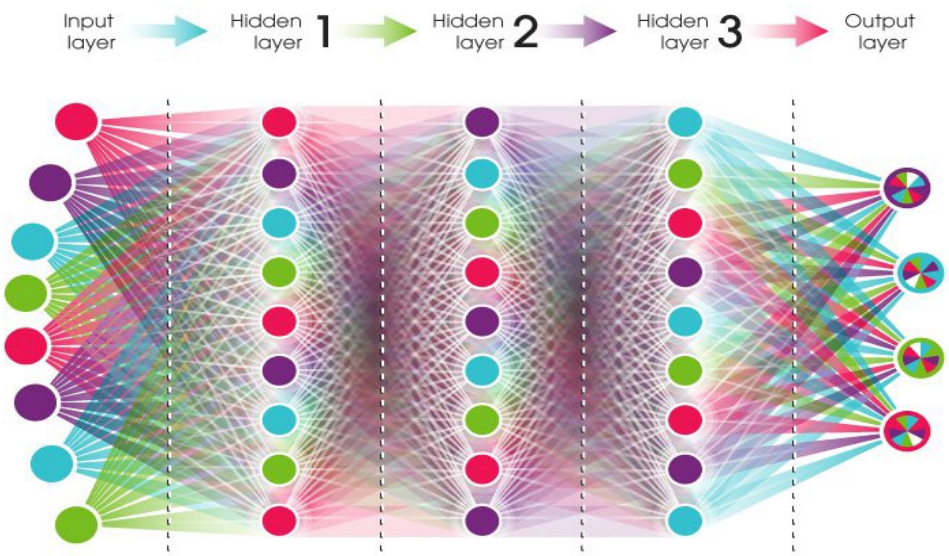

FIGURE 3. Standard Architecture of a Deep Neural Network as depicted in Goodfellow et. al (2016). Relative to such standard architecture, the networks used for text and image processing have very high dimensional inputs and very high-dimensional outputs, and low-dimensional intermediate hidden layers. Precisely these layers, typically the penultimate layers, are used for dense embedding. In the hedonic price prediction network the penultimate layer is interpreted as value embedding and the output layer contains predicted hedonic prices in all time periods.

which is multivariate. The networks with vector outputs are called the multi-task networks.

Prediction methods based on neural networks with several layers of neurons are called the "deep learning" methods. The popularity of the methods stems from their impressive success in solving various problems. Neural networks recently emerged as a powerful and all-purpose method for regression and classification analysis. Using many neurons and multiple layers gives rise to deeper networks that are very flexible and can approximate the best prediction or classification rules quite well.

3.2. **Assessing Statistical Significance and Confidence Intervals.** The last layer of the NN model provides us with what we interpret as "value embeddings",

$$V_{it} = (V_{1t}, ..., V_{pt})',$$

where we have considered p up to 512. We condition on the training and validation data, so that V_i's are considered as frozen for all products i. Then we use the hold-out data to estimate the following linear regression model:

$$P_{it} = V_{it}'\beta_t + \nu_{it}, \quad \beta_t = (\beta_{1t}, ..., \beta_{pt})'.$$

This is a low-dimensional linear regression model, for which we can apply standard inference tools based on linear regression. Applying linear regression to the test data, we obtain the estimate $\hat{\beta}_t$ and the estimated hedonic price

$$\hat{H}_{it} = V_{it}'\hat{\beta}_t. \tag{8}$$

For example, the statistical significance of features can be assessed by testing whether the regression coefficients β_t are equal to zero, using p-values and adjusting for multiplicity using Bonferroni approach (or other standard approaches such as step-down methods or methods that control false discovery rates). We can also construct confidence intervals for individual coefficients, as well as the level $1-\alpha$ confidence intervals for the predicted hedonic price:

$$[L_{it}, U_{it}] = [\widehat{H}_{it} \pm \Phi^{-1}(1-\alpha/2)\mathrm{SE}_{it}], \quad \mathrm{SE}_{it} = \sqrt{V_{it}'\widehat{\mathrm{Cov}}(\hat{\beta}_t)V_{it}}. \tag{9}$$

This advantage of this approach is its simplicity, while the disadvantage is that it does not account for uncertainty in estimating the value embeddings themselves (indeed, we consider them as frozen conditional on the training+validation sample).

Following Chernozhukov et. al. (2018), one way to account for this variability is to consider random multiple splits, $s = 1, ..., S$, of the data (with stratification by month), into the test subset and training+validation data subset. Different splits would result in different value embeddings V_{it}^s, the coefficient estimates $\hat{\beta}_t^s$, the predicted hedonic price , and the covariance estimates $\widehat{\mathrm{Cov}}(\hat{\beta}_t)^s$, as well as the confidence intervals $[U_{it}^s, L_{it}^s]$. Then we can aggregate the estimates and the confidence intervals as follows:

$$\widetilde{H}_{it} = \mathrm{median}\left((\hat{H}_{it}^s)_{s=1}^S\right), \quad \widetilde{CI}_{it} = \left[\mathrm{median}\left((\hat{L}_{it}^s)_{s=1}^S\right), \mathrm{median}\left((\hat{U}_{it}^s)_{s=1}^S\right)\right]. \tag{10}$$

The adjusted nominal level for the confidence interval for $V_{it}'\beta_s$ is $1-\alpha/2$. Similarly, for judging statistical significance of particular coefficients (or other functionals) we can consider multiple p values $(P^s)_{s=1}^S$ and aggregate them by taking the median p-value and comparing this p-value to the adjusted nominal level $\alpha/2$. This approach exploit the fact that the median of arbitrary correlated variables with the marginal standard uniform distribution is stochastically dominated by the variable $2 \cdot \text{Uniform}(0,1)$. The p-values are asymptotically uniformly distributed. We have not pursued this approach in this version of the paper due to the high computational costs associated with training S instead of just one model.

The above approach extends to other ways of quantifying modeling uncertainty. Indeed, $s = 1, ..., S$ could as well denote various perturbations of the network structure, e.g. using drop-out perturbations, which randomly omit neurons in each layer, generating different value embedding different value embeddings V_{it}^s. The mathematical argument in Chernozhukov et al (2018) does not rely on the source of multiplicity generated in this way and the confidence intervals and statistical significance calculations immediately extend to cover this quantification.

4. Image and Text Embeddings Via Deep Learning

A customer usually sees the product in the form shown in Figure 1. Our task is to convert the product characteristics, such as the product title, description, image, and into numerical vectors which can be used for constructing hedonic price, as shown in Figure 4. In what follows we give a high-level description of how the text and image features are generated.

4.1. **Text Embeddings from the Title and Product Description.** We begin with text processing.

First generation: Word2Vec Embeddings. We first recall some basic ideas underlying the Word2Vec algorithm (Mikolov et al., 2013). The j-th word in the product description can be represented by a binary encoding of a very high dimension d, but this representation is not very useful because it is too sparse and it is not able

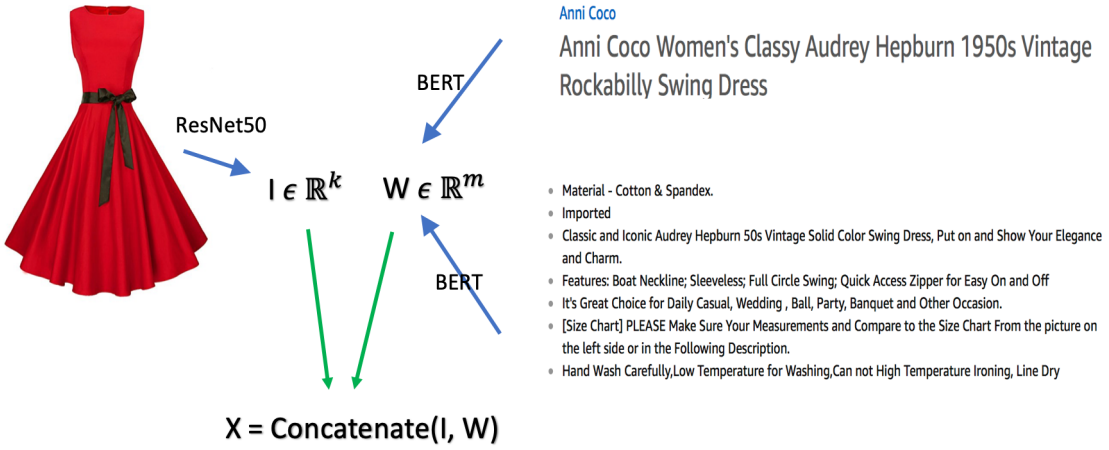

FIGURE 4. A schematic illustration of the map from the product text information and image to the embedding features $X = (I', W')'$. Our best model uses BERT DNN to convert title, brand, product description to an embedding vector W, and uses ResNet50 to convert winning image to an embedding vector I.

to explore word similarity to compactly approximate the dictionary. Instead we aim to represent words by vectors of much lower dimension r. Denote representation of j-th word by u_j, then the dictionary is $r \times d$ matrix

$$\omega = \{u_j\}_{j=1}^{d},$$

where r is the reduced dimensionality of the dictionary. Then each word t_j in a human-readable dictionary can be represented by word u_j. In our context, we can think of each word appearing in the product description as a random variable T and represent its corresponding embedding representation by U.

The goal of Word2Vec is to find an effective representation with the dimension r of the embedding to be much smaller than d. This is achieved by treating ω as parameters and estimating them so that the model performs well in some basic natural language processing tasks. These tasks are not related to hedonic prices.

One of the ways to train the word embeddings is to predict the middle word from the words that surround it in word sentences. Given a subsentence s of $K+1$ words, we have a central word $T_{c,s}$ whose identity we have to predict and we have the words $\{T_{o,s}\}$ that surround it. Collapse the embeddings for context words by a sum,

$$\bar{U}_o = \frac{1}{K}\sum_o U_{o,s},$$

where $U_{o,s}$ is the element of ω corresponding to the word $T_{o,s}$. This step imposes a drastically simplifying assumption that the context words are exchangeable.

The probability of middle word $T_{c,s}$ being equal to t is modeled via multinomial logit function:

$$p_s(t;\pi,\omega) := P\Big(T_{c,s} = t \mid \{T_{o,s}\};\omega\Big) = \frac{\exp(\pi_t'\ \bar{U}_s(\omega))}{\sum_{\bar{t}}\exp(\pi_{\bar{t}}'\ \bar{U}_s(\omega))},$$

where $\pi = (\pi_1, ..., \pi_d)$ is $m \times d$ matrix conformable parameter vectors defining the choice probabilities. The model constraints $\pi = \omega$, and estimates ω by using the maximum quasi-likelihood method:

$$\max_{\omega=\pi}\sum_{s\in\mathcal{S}}\log p_s(T_{i,s};\pi,\omega),$$

where the summation is over many examples $\mathcal{S}$ of subsequences s. Once we are done training, we can generate the embedding for title or description of product i, containing the embedded words $\{U_{j,i}\}_{j=1}^J$ by simply averaging them:

$$W_i = \frac{1}{J}\sum_{j=1}^{J} U_{j,i}. \tag{11}$$

In summary, the Word2Vec algorithm transforms text into a vector of numbers that can be used to compactly represent words. The algorithm trains a neural network in a supervised manner such that the contextual information is used to predict another part of the text. For example, let's say that the title description of the item is: "Hiigoo Fashion Women's Multi-pocket Cotton Canvas Handbags Shoulder Bags Totes Purses". The model will be trained using many n-word subsentence examples, such that the center word is predicted from the rest. If we just use $K = 3$ subsentence examples, then we train the model using the following examples: (Hiigoo,Women's)

$\rightarrow$ Fashion, (Fashion,Multi-pocket)$\rightarrow$ Women's, (Women's,Cotton) $\rightarrow$ Multi-pocket, and so on.

How do we judge whether the text embedding is successful or not? In the hedonic price context, we can check whether Word2Vec features improve the quality of prediction of the price by the hedonic model. [9] We can also check if similar words T_k and T_l have similar embeddings. We can measure the similarity through the correlation or cosine-similarity, more precisely:

$$\text{sim}(T_k, T_l) = U_k' U_l / (\|U_k\| \|U_l\|).$$

The more similar the words are, according to our human notion of similarity, the higher the value the formal similarity measure should take, with the maximal value of 1. For example, the following are the 5 words that are most similar to "tie" under the similarity measure: "necktie" and "bowtie". The dense embedding also induces an interesting vector space on the set of words, which seem to encode analogies well. For example, the word "briefcase" is most cosine-similar to the artificial latent word

$$\text{Word2Vec(men's)} + \text{Word2Vec(handbag)} - \text{Word2Vec(women's)}.$$

Word2vec embeddings were among the first generation of early successful algorithms. These algorithms have been improved by the next generation of NLP algorithms, such as ELMO and BERT, which are discussed next.

Second Generation: ELMO. The Embeddings from Language Models (ELMO) algorithm (Peters et al, 2018) uses the ideas of the Shannon game, where we guess the next word in the sentence m with n words, i.e.

$$p^f_{k,m}(t) = P[T_{k+1,m} = t | T_{1,m}, ..., T_{k,m}; \theta]$$

and also uses the reverse guessing as well:

$$p^b_{k,m}(t) = P[T_{k-1,m} = t | T_{k,m}, ...T_{n,m}; \theta],$$

[9] Indeed, in our early experiments we verified that such embeddings were helpful in prediction and improved over using basic catalogue features, and our experiments below make use of more sophisticated word embeddings that improved the prediction even further.

where θ is a parameter vector. Recursive neural networks with a single or multiple hidden layers are used to model these probabilities. Parameters are estimated using quasi maximum log-likelihood methods, where the forward and backward log quasi-likelihoods are added together.

To give a simple example, suppose we wanted to grasp the context better in the previous example, so we could be, instead of collapsing embeddings for the context word by a sum assign individual parameters to each context. This would result in a model closely resembling the previous model, but where the order of context words would play a role. For example, we could model

$$P(T_{k,m} = t \mid \{T_{j,m}\}_{j=1}^{k-1}) = \frac{e^{\sum_{j=1}^{k-1} \pi'_{t,k} U_{k,m}(\omega)}}{\sum_{\bar{t}} e^{\sum_{j=1}^{k-1} \pi'_{\bar{t},k} U_{k,m}(\omega)}},$$

and similarly in reverse order. ELMO uses a more sophisticated (and more parsimonious) non-linear recursive nonlinear regression (recurrent neural network) model to build these probabilities, shown in Figure 5.

The basic structure of ELMO is as follows: Given a sentence m of n words, (1) words are mapped to context-free embeddings in $\mathbb{R}^d$. (2) A network is trained to predict each word $T_{k,m}$ of a string given (a) words $(T_{1,m}, \ldots, T_{k-1,m})$ or (b) words $(T_{k+1,m}, \ldots, T_{n,m})$. The objective is to minimize the average over sum of the log-loss over the $2n-2$ prediction tasks, where the average is taken over all sentences. (3) The embedding of word $T_{k,m}$ is given by a weighted average of outputs of certain hidden neurons corresponding to this word's entire context. Importantly, the same final logistic ("softmax") layer is used for prediction objectives (2a) and (2b). Thus the inputs to this layer, which represent the forward and backward context, are constrained to lie in "the same space."

Training. In Figure 5, the output probability distribution p_k^f is taken as a prediction of $T_{k+1,m}$; similarly p_k^b is taken as a prediction of $T_{k-1,m}$. The parameters of the network θ are obtained by maximizing quasi-loglikelihood:

$$\max_{\theta} \sum_{m \in \mathcal{M}} \left(\sum_{k=1}^{n-1} \log p_{k,m}^f(T_{k+1,m}; \theta) + \sum_{k=2}^{n} \log p_{k,m}^b(T_{k-1,m}; \theta) \right),$$

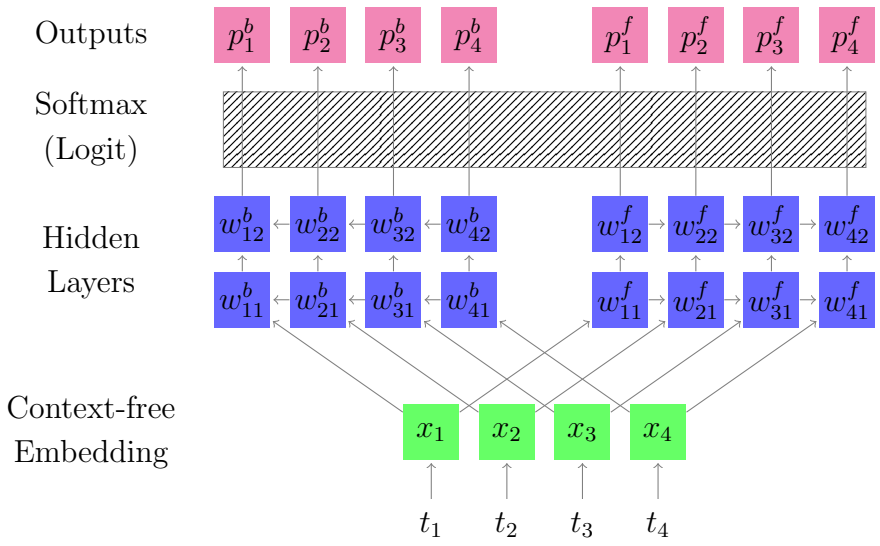

FIGURE 5. ELMO Architecture. This is ELMO network for a string of 4 words, with $L = 2$ hidden layers. Here, the softmax layer (multinomial logit) is a single function mapping each input in $\mathbb{R}^d$ to a probability distribution over the dictionary Σ.

where $\mathcal{M}$ is a collection of sentences (titles and product descriptions) in our data set.

Producing embeddings. To produce embeddings from the trained network, each word t_k in a sentence $m = (t_1, ..., t_k)$ is mapped to a weighted average of the outputs of the hidden neurons indexed by k:

$$t_k \mapsto w_k := \sum_{i=1}^{L} \gamma_i w_{ki}^f + \bar{\gamma}_i w_{ki}^b.$$

The embedding for the sentence (or product description) is produced by summing the embeddings for each individual word. The weights γ and $\bar{\gamma}$ can be tuned by the neural network performing the final task. In principle, however, the whole network could be plugged in to the network performing the final task and allowed to update.

Second generation: BERT. Bidirectional Encoder Representations from Transformers (BERT) is another contextualized word embedding learned from deep language model (Devlin et al, 2018). It is a successor of ELMO and achieved state-of-art results on multiple NLP tasks, improving somewhat on ELMO. Instead of using Recurrent Neural Network as in ELMO, BERT uses the Transformer structure with attention mechanism (Vaswani et al., 2017) that pays attention to whole sentence or context.

Unlike the language model in ELMO which predicts the next word from previous words, the BERT model is trained on two self-supervised tasks simultaneously:

- Mask Language Model: randomly mask certain percentage of the words in the sequence and predict the masked words
- Next Sentence Prediction: given a pair of sentences, predict whether one sentence proceeds another.

The structure of BERT model is as follows: (1) Each word in the input sentence is broken to subwords and tokenized using a context-free embedding called WordPiece. A special token [`cls`] is added to the beginning of the sequence. And x% of the tokens are replaced by [`mask`]; (2) For each token, its input representation consists of i) its token embedding from (1), ii) position embedding indicating the position of the token in the sentence, and iii) segment embedding indicating whether it belongs to sentence A or B. (3) The input representation of tokens in the sequence is fed into the main model architecture: L layers of Transformer-Encoder blocks. Each block consists of a multi-head attention layer, followed by a feed forward layer. (4) The output representation of the mask token [`mask`] is used to predict the masked word via a softmax layer, and the output representation of the special [`cls`] token is used for Next Sentence Prediction. The loss function is a combination of the two losses.

We next focus in detail on the main structure step (3), especially the "multi-head attention" layer.

Computing The Attention. We begin with n word context-free embeddings $(x_1, x_2, \ldots, x_n)$, with each $x_k \in \mathbb{R}^d$. Let X denote the matrix whose kth row is x_k. The Multi-Head

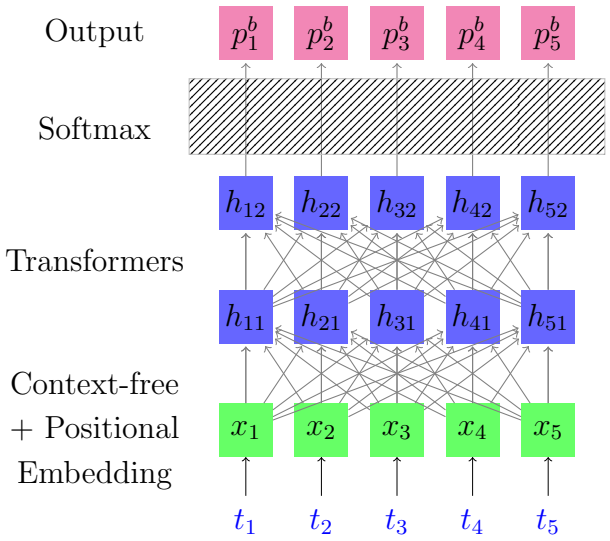

FIGURE 6. BERT Architecture

Attention mapping is applied on X directly:

$$\mathsf{X} \longmapsto \text{MultiHead}(\mathsf{X}, \mathsf{X}, \mathsf{X}),$$

where

$$\text{MultiHead}(\mathsf{Q}, \mathsf{K}, \mathsf{V}) = \text{Concatenate}(\text{Head}_1, \ldots, \text{Head}_h)\omega^O,$$
$$\text{Head}_i = \text{Attention}(\mathsf{Q}\omega_i^Q, \mathsf{K}\omega_i^K, \mathsf{V}\omega_i^V),$$
$$\text{Attention}(\tilde{\mathsf{Q}}, \tilde{\mathsf{K}}, \tilde{\mathsf{V}}) = \text{softmax}\left(\tilde{\mathsf{Q}}\tilde{\mathsf{K}}^T/\sqrt{d_k}\right)\tilde{\mathsf{V}},$$

where ω^O and $(\omega_i^Q, \omega_i^K, \omega_i^V)$ are matrix parameters, which are trained to maximize the model performance. In other words, each word embedding is replaced by weighted average of all other words', and the weights are learned from the scaled dot-product of different projections of the word embeddings themself. The projection matrices are parameters learned during training.

Generating product embeddings. Depending on specific tasks and resources, Devlin et al. (2018) suggested to use the BERT embeddings in various ways: 1) use the last layer, second-to-last layer or concatenate last 4 layers of the encoder outputs from the pre-trained BERT model, 2) fine tune the whole BERT model on the downstream task, or 3) train the BERT language model from scratch on the new data. For this

study, we choose the feature-based approach, and extract the second-to-last layer as embeddings from the pre-trained BERT model. Each product's text embedding is the average of the embeddings of each word/token from the input text field.

Comparing ELMO and BERT. While ELMO and BERT are both recent breakthroughs in NLP, the former marked the first contextual word embedding trained from deep language model, and the latter was the first contextual word embedding using Transformer architecture. Note that since the BERT paper was published second and could respond directly to the ELMO paper (but not vice-versa), the comparisons are likely to be biased towards the latter.

There appear to be several differences between the two proposals: (1) They use different initial context-free embeddings. (2) ELMO applies an initial convolutional layer to a *character* embedding, while BERT augments the WordPiece embedding at *sub-word* level with positional data. ELMO is based on Recurrent Neural Network while BERT is based on Transformer architecture. (3) The ELMO implementation only allows the averaging weights to be fine-tuned, whereas BERT proposes fine-tuning the whole network. The biggest difference lies in the choice of fundamental architectures. RNNs are known for not being able to capture long-term dependencies. The transformer architecture is more efficient at capturing long-range dependencies in the text. Furthermore, ELMO creates context by using the left-to-right and right-to-left language model representations, while in BERT models the entire context simultaneously.

4.2. **Construction of Image Embedding using ResNet50.** One of the most successful deep learning methods for image classification was developed He et al. (2016). At the time of the release, the paper achieved the best results in image classification, in particular, for the ImageNet and COCO datasets.

The central idea of the paper itself is to exploit "partial linearity": traditional nonlinearly generated neurons are combined (or added together) with the previous layer of neurons. More specifically, a building block is to take a standard feed-forward convolutional neural network and add skip connections that bypass two (or one or

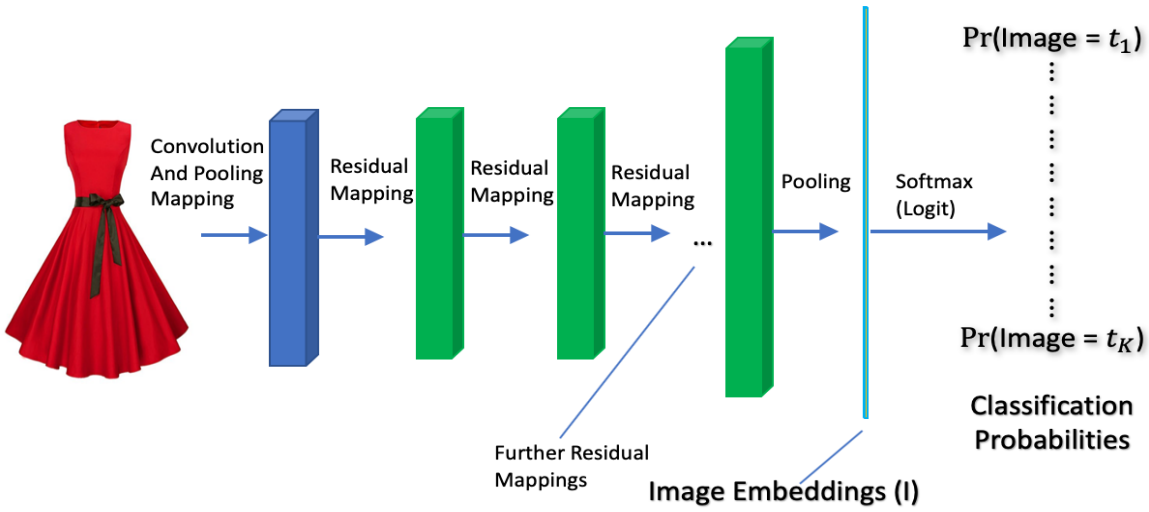

FIGURE 7. The ResNet50 operates on numerical 3-dimensional arrays representing images. It first does some early processing by applying convolutional and pooling filters, then it applies manyL residual block mappings, producing arrays shown in green. The penultimate layer produces a high-dimensional vector I, the image embedding, which is then used to predict the image type.

several) convolution layers at a time. Each skipping step generates a residual block in which the convolution layers predict a residual. Formally each k-th residual block is a neural network mapping

$$v \longmapsto (v, \sigma_k^0(\omega_k^0 v)) \longmapsto (v, \sigma_k^1 \circ \omega_k^1 \sigma_k^0(\omega_k v)) \longmapsto v + \sigma_k^1 \circ \omega_k^1 \sigma_k^0(\omega_k^0 v),$$

where ω's are matrix-valued parameters or "weights". This can be seen as a special case of general neural network architecture, designed so that it is easy to learn the identity maps (or submaps entering the composition of the entire network). Putting together many blocks like these sequentially results in the overall architecture depicted in Figure 7.

The deep feed-forward convolutional networks developed in prior work suffered through major optimization problems – once the depth is high enough, additional

layers often resulted in much higher validation and training error. It was argued that this phenomenon was a result of vanishing gradients, making it difficult to optimize. The residual network architecture addresses this by using the residual block architecture. This possibility allowed high quality training even for very deep networks.

Just like with text embeddings, we are not interested in the final predictions of these networks but rather in the last hidden layer, which is taken to be the image embedding. We rely on publicly trained ResNet50 model to generate inference.[10]

5. Empirical Performance of Hedonic Price Models

Data. We use Amazon's proprietary data on daily average transaction prices and quantities for the first-party sales from the entire population of apparel, that has tens of millions of products sold in the Amazon store.[11] Our study covers the period from 01/2013 to 11/2018. The transaction prices of a product j in month t are defined as the ratio of total sales (S_{it}) over the quantity sold (Q_{it}),

$$P_{it} = S_{it}/Q_{it},$$

where the price is treated as missing for the case of no sales.

One key characteristic of our data is the very high turnover of products from month to month, as shown in Figure 12 below. Moreover, there is a considerable growth in selection of products. The data set we used for this analysis was roughly 20 terabytes because of the number of products and the length of the time period covered. The Spark environment on an EMR cluster was the most appropriate environment to hold and process such large data. The product embeddings and estimation of the hedonic

[10]We also tested with image model fine tuned on Amazon Catalog with similar structure as ResNet50, and observed some small performance improvement.

[11]The quantity information is proprietary, but we note, however, that there are methods of approximating quantity weights based upon product ranking data; see, e.g., Chessa and Griffioen (2019) and Office of National Statistics (2020). Therefore, the hedonic prices and hedonic price indices derived using quantity weight can be approximated to a various extent by the publicly available price information, product information and images, and product rank information.

price functions were carried out on a GPU cluster of the Amazon Web Services using the Sagemaker notebooks.

Performance for Predicting Prices out-of-Sample. In Table 2 we first examine the predictive performance, recording the R^2 for predicting prices in the hold-out sample of products. We consider three methods for estimating the hedonic function:

- linear regression using features generated by a collection of binary encodings of the product catalogue;
- linear regression using DL features W and I generated by BERT and ResNet50;
- tree-based (random forest and generalized boosting) regression using DL features W and I;
- multi-task neural network regression using DL features W and I.

As we can see from Table 2, as we switch from the basic catalogue features to using the DL embeddings W and I, we obtained the first major improvement, just using linear regression methods. As we switch from the linear regression to a scalable implementation of tree-based methods, we obtained the second major improvement. Finally, as we switch from the random forest to the multi-task neural network we obtain the final major improvement. Thus, the neural network model easily achieves better predictive performance over other methods.

Figure 8 also presents the month-to-month performance of the various models. The out-of-sample R^2 for the best multi-task NN model ranges between 80% to 90%. Multi-task NN uniformly dominate singe-task NNs, which in turn uniformly dominate boosted tree models, which in turn uniformly dominate linear models. The BERT-based multi-task NN nearly uniformly outperforms ELMO-based multi-task NN, although the performance of the two models is generally similar. The performance of the best models is better at the beginning of the studied period and worsens at the end of the period, perhaps in part due to increasing variety and number of products being transacted (this is shown in Figure 13). It is worth mentioning that the out-of-sample R^2 agrees with validation R^2 we obtained in training (not reported), which is reflective of our training approach that assures that no overfitting happens: we make sure that the training R^2 and the validation R^2 are very close.

Method		R^2
Linear Model with basic catalogue features	$\approx$	$30-45\%$
Linear Model with DL features W and I	$\approx$	$55-65\%$
Random Forest/Boosted Tree Models with DL features W and I	$\approx$	$70-80\%$
Single-Task Neural Network with DL features W and I	$\approx$	$75-85\%$
Multi-Task Neural Network with DL features W and I	$\approx$	$80-90\%$

TABLE 2. Summary of Out-of-Sample Performance of the Empirical Hedonic Price Function.

Examples of hits and misses. It is instructive to examine the performance of the best NN model through the lenses of particular examples. Here we take one example where the prediction worked well and another where the prediction worked poorly, see Figures 10 and 11. For the first example, a sweater, the NN model predicts the price of this item at about 100. The time-averaged average price as shown on camelcamel.com is 97, with the price ranging from 39 (corresponding to liquidation events) to 120. For the second example, a designer dress, the NN predicts the price of this item at about 300, but the most recent offer prices for this item were around 2400. While this seems like a major miss, the price history for this item as seen on camelcamel.com, suggests that there were periods where the price for this item ranged between 206 and 2800, with an average sale price of 464. An important feature that is possibly missed by the NN embeddings are "cruising around custom rose gold buttons," although it is hard to be verify the importance of this feature objectively.

Statistical Significance and Inference. Next we examine the statistical significance of the hedonic price model using the approach of Section 3.2. We illustrate this approach in Figure 9, which reports the estimated coefficients on value embeddings for the month of November 2018 and reports the component-wise confidence intervals. We also illustrate the construction of the confidence intervals for predicted hedonic price in Table 3, which reports the 90% confidence intervals for estimated hedonic prices for two example products.

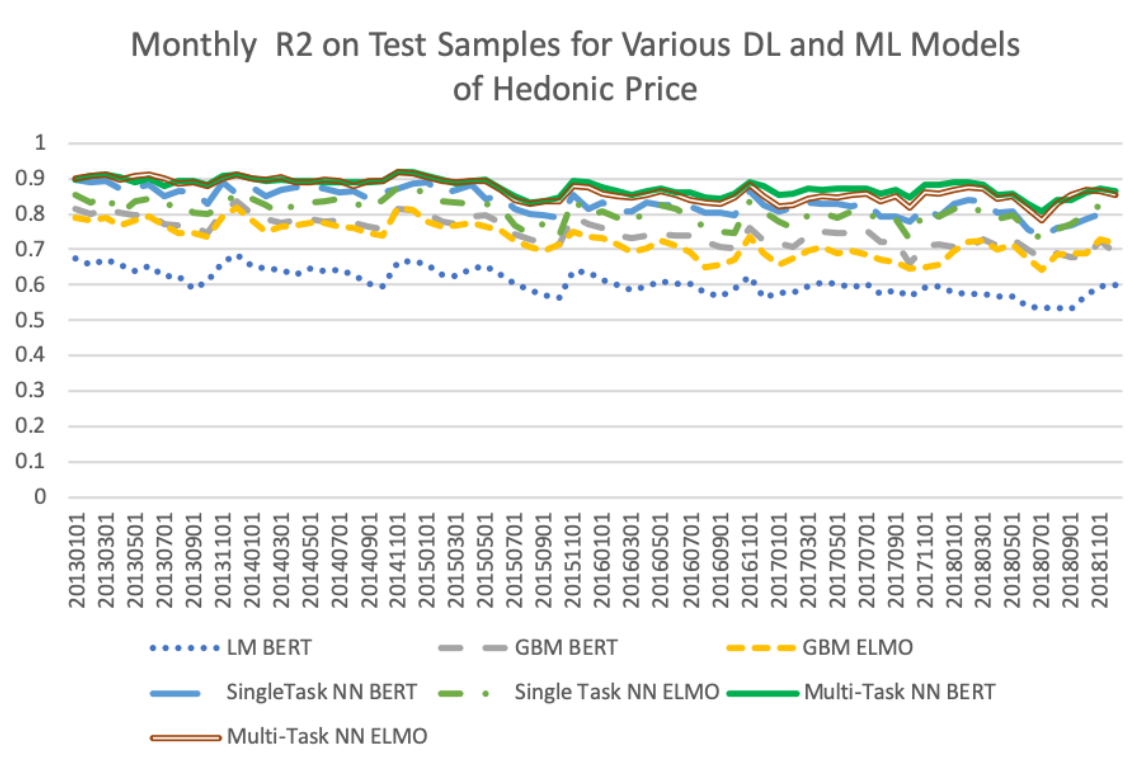

FIGURE 8. The Out-of-Sample Performance of the Empirical Hedonic Price Function obtained Using Neural Network every month since March of 2013. Multi-Task Neural Networks dominate Singe-Task Neural Networks, which in turn dominate Boosted Tree Models, which in turn dominate Linear Models.

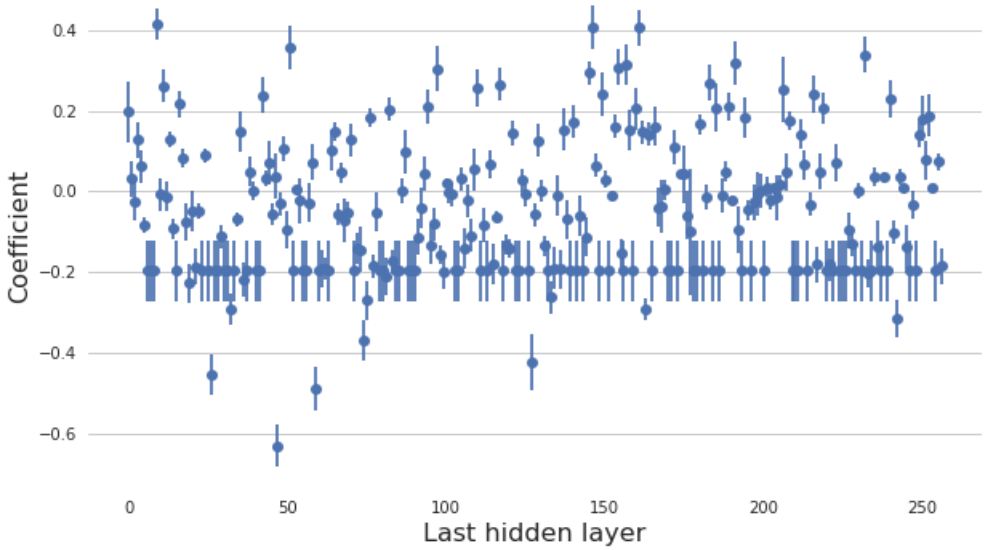

FIGURE 9. Statistical Significance of Value Embeddings via Linear Regression Model Applied to the Test Sample. The figure shows the point estimates and the pointwise 95% confidence intervals on the coefficients of the value embeddings as estimated by the linear regression model applied to the hold-out sample. Note that two thirds of the coefficients are significant at the 10^{-5} level.

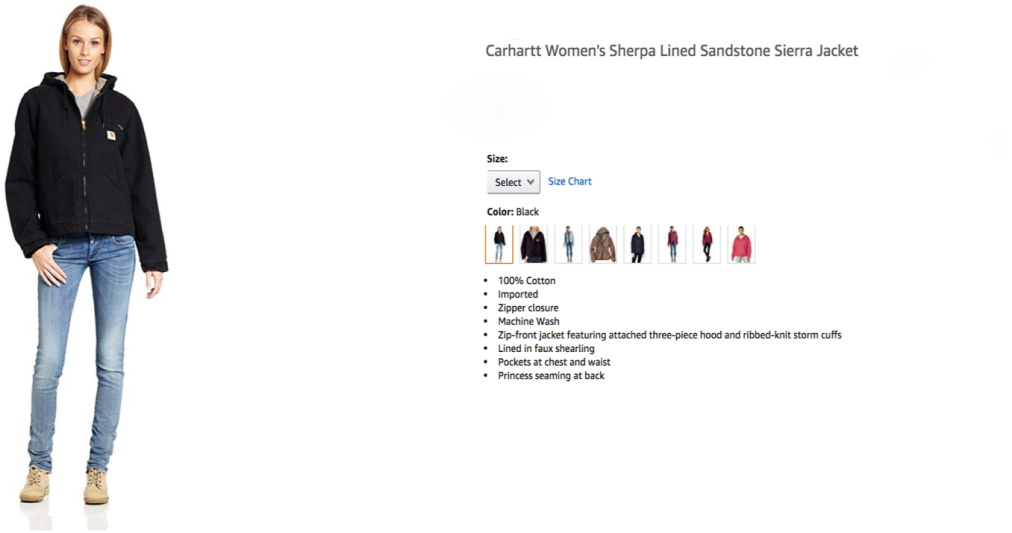

FIGURE 10. An example of accurate prediction of (B001GUN1N6): The Neural Network model predicts the price of this item at about 100. The average price on camelcamel.com is 97, with the price ranging from 39 to 120.

Product	Year/Month	P_{it}	$\widehat{H}_{it}$	SE_{it}	$[L_{it}(H_{it}), U_{it}(H_{it})]$	$[L_{it}(P_{it}), U_{it}(P_{it})]$
B01DJ34PD2	201812	118.8	114.6	0.05	[114.5, 114.7]	[102, 126]
B01MSBDGTL	201812	119.9	110.25	0.06	[110.1, 110.4]	[98 , 122]

TABLE 3. Examples of Construction of Confidence Intervals for Predicted Hedonic Price $H_{it} = V_{it}'\beta_t$ and the Sale Price P_{it}. Here $\widehat{H}_{it} = V_{it}'\hat{\beta}_t$ is the estimated hedonic price. The term $\hat{\sigma}^2 = V_{it}'\widehat{\text{Cov}}(\hat{\beta}_t)V_{it}$ is the square of the standard error, and $[L_{it}, U_{it}] = [\widehat{H}_{it} \pm z_{.95}\hat{\sigma}]$ is the 90% confidence interval for H_{it}. The predictive confidence interval for P_{it} is $[L_{it}(P_{it}), U_{it}(P_{it})] = [\widehat{H}_{it} \pm z_{1-\alpha/2}\hat{\nu}]$ with $\hat{\nu}^2 = \hat{\sigma}^2 + \widehat{\text{Var}}(P_{it} - H_{it})$. The term $z_{.95}$ is .95-th quantile of the standard normal distribution.

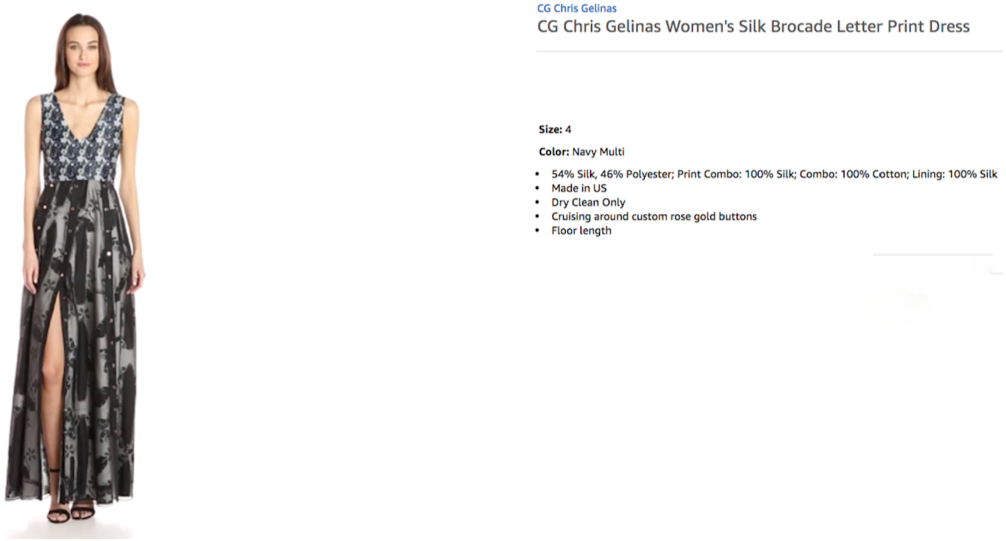

FIGURE 11. An example of inaccurate prediction: The Neural Network model predicts the price of this item at about 300, but the recent offer prices for this product were around 2400. While this seems a miss, the price history for this item (B06XR39DJ1), that can be seen on camelcamel.com, suggests that there were periods where the price for this item ranged between 206 and 2800 and 206, with an averaged price of 464.

Apparel Indexes	*Inflation*
Fisher Hedonic, Yearly Chaining (FY)	-0.77%
Fisher Hedonic, Monthly Chaining (FM)	-5.32%
Fisher Hedonic, GEKS ($\sqrt{FY \cdot FM}$)	-3.16%
Fisher Matched, Monthly Chaining	-3.46%
Jevons Posted, Daily Chained	-2.82%
Adobe DPI, Monthly Chained	-1.30%
U.S. Urban Apparel (BLS)	-0.04%

TABLE 4. Estimates of Average Annual Rate of Inflation in Apparel over 4 years, 2013-2017: Fisher Hedonic Index, Fisher Matched Index, Jevons Posted Price Index, Adobe DPI, and the BLS Index for Urban Areas. Adobe DPI is based on 2014-2017.

6. Hedonic Price Indices for Apparel

Here we will use the definitions of hedonic indices introduced in Section 2. For computation of the hedonic indices we use the best predictive model – the multi-task NN model with W and I features, where W features are obtained as BERT embeddings and I features are obtained as ResNet50 embeddings.

In Table 4, we present our main result – the estimates of average annual rate of inflation in apparel over 2013-2017, via GEKS Fisher HPI, the Fisher HPI, Jevons Posted PI, and the CPI for Apparel in Urban Areas by BLS:

The main conclusions we draw from these results are the following.

E.1 All indices, apart from CPI, suggest an average price decline in 2013-2017 of at least .77%. In contrast, the BLS CPI for apparel suggests that there was very little decline in the price level.

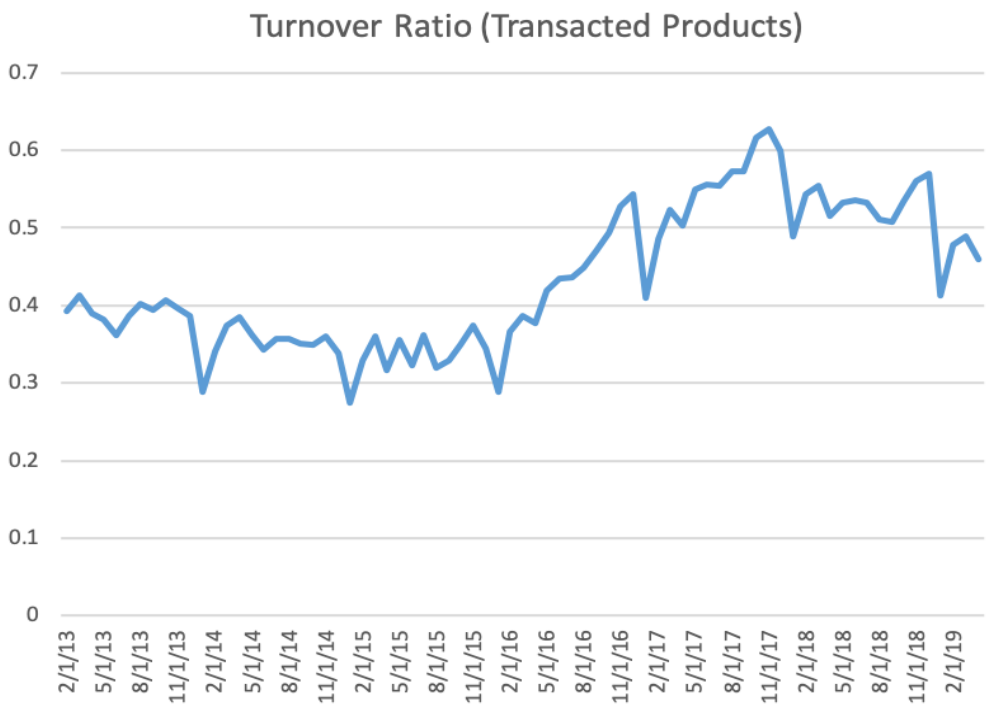

FIGURE 12. Turnover Rate for Products. The figure shows the ratio of the number of products with transactions in a given month and no transaction in the previous month.

E.2 GFHPI declines at a slower rate than the matched Fisher index potentially highlighting the importance of hedonic adjustment.

Indeed, the matched index potentially contains a matching bias created by the high turn-over of products illustrated in Figure 12.

The differences with CPI could be attributed to a number of sources: There are methodological differences: BLS uses a hybrid index where price levels are first measured within narrow subgroups of products, without quantity weighting, and then aggregated by Tornqvist index using expenditure shares for subgroups; moreover,

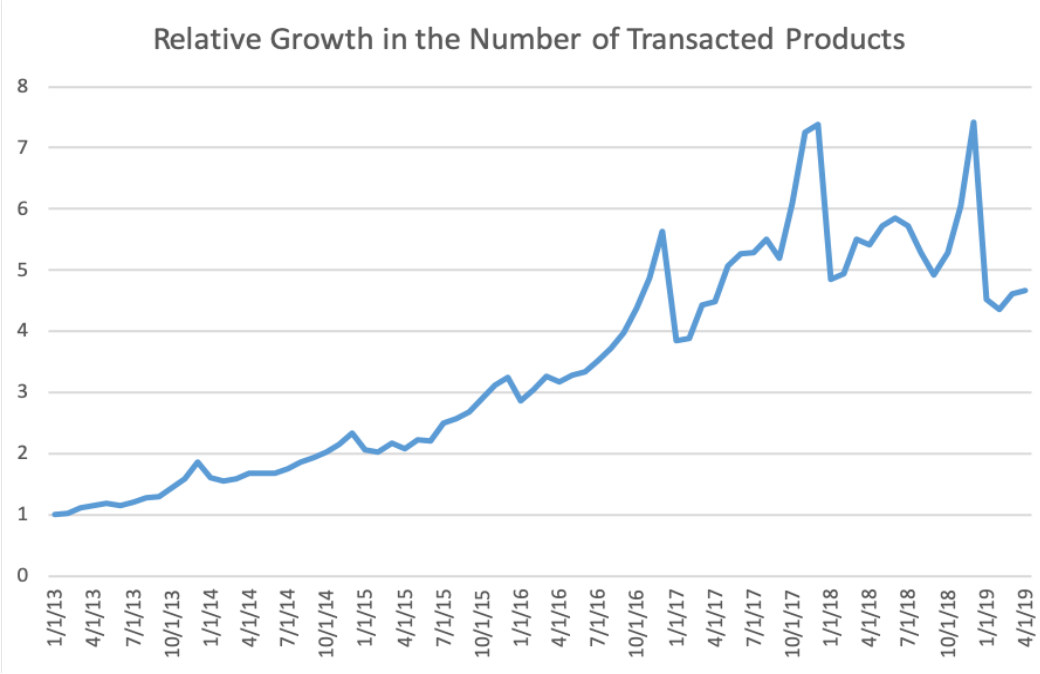

FIGURE 13. Number of Products in the Current Period over the Base Period.

BLS also uses different quality adjustments. Moulton (2018) discusses methodological points, and estimates that the upward bias in the overall (non-apparel specific) CPI index may be in the range $[.4\%, 1.3\%]$, which could potentially reconcile some of the difference. Other sources of the difference could be the Amazon-specific productivity improvements leading to lower prices (e.g., improvements in supply-chain productivity) and different shares of products in the customers' baskets

We also performed similar calculations using linear model instead of NN (not reported), and conclusions seem qualitatively robust with respect to whether linear model or nonlinear NN model used. However, the nonlinear NN models, which do exhibit superior predictive ability, results in a more pronounced quantitative drop in the price index level. For example, the month-over-month chained FHPI index based on the linear model declined 5% less (in absolute terms) over 2013-2017 than the same index based the NN model. The difference is less drastic than one might

expect from having markedly different predictive performance, but the hedonic price index is a ratio of weighted averages of predicted hedonic prices. Since noise gets reduced by averaging, the key determinants of the biases in the index are the weighted averages of biases of the estimated hedonic prices. The NN models are more noisily estimated than the linear models, but they are much less biased giving better predictive performance as a result. Still, taking weighted averages of prices translates into only "moderate" differences between the indices based on the NN and on the linear models.

In Figure 14, we demonstrate the construction and dynamics of the GFHPI and compare it with the Jevons index. The GFHPI is constructed by taking the geometric mean of the Fisher HPI chained at month-over-month frequency and the Fisher HPI chained at year-over-year frequency.[12] The Jevons index is a geometric mean of the posted price relatives, and does not incorporate any quantity weighting. It measures the average "within" price level change calculated on a much larger universe of products (with and without transactions). The Fisher index reflects both the "within" price change and the "between" price change, since it also reflects the substitutions arising from cost-minimizing behavior by customers. The key empirical observations we draw are as follows.

E.3 The GFHPI and yearly-chained FHPI exhibit slower rates of decline than the monthly-chained FHPI, highlighting perhaps the importance of the "chain drift".

E.4 The Jevons index exhibits steady declines over the studied period, where the rate of decline is slightly smaller than the rate of decline in GFHPI.

This may suggest that the "between" effects are not large, and that GFHPI can be approximated using the Jevons index. This is attractive because the Jevons index can be constructed by web-scraping the publicly available price data.

[12]As mentioned before, the motivation for GFHPI is to mitigate the chain drift problem, caused by propagation of errors due to frequent compounding in monthly chaining.

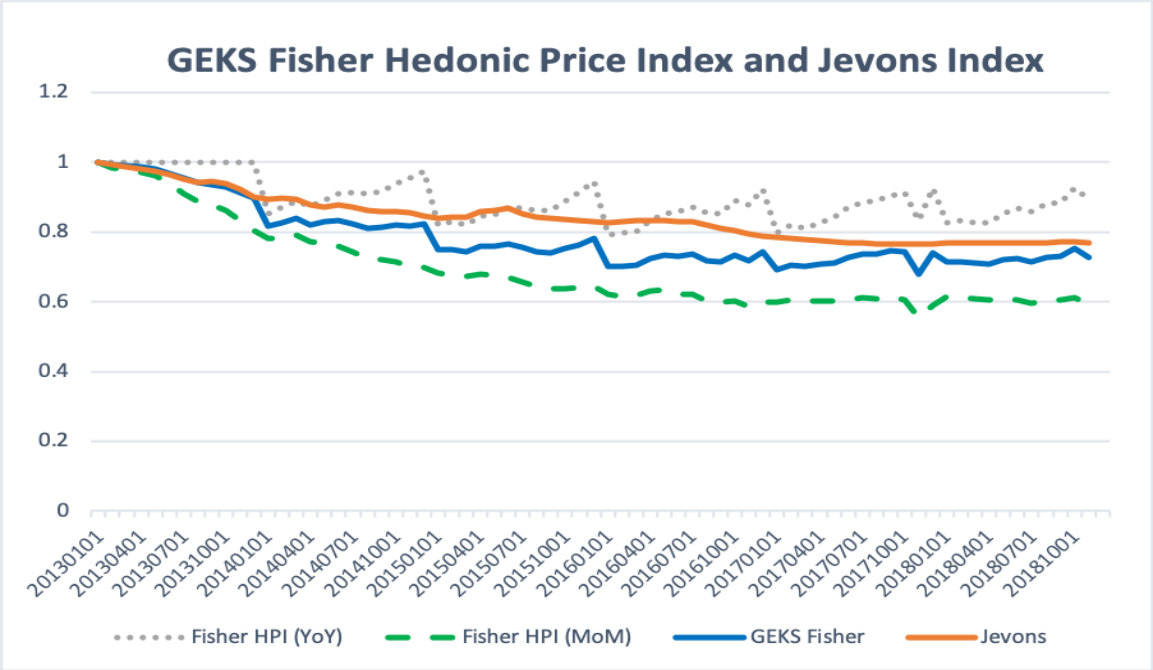

FIGURE 14. Construction of Dynamics of the GEKS-type Fisher Hedonic Price Index and Comparison with the Jevons index. The index is constructed by taking the geometric mean of the Fisher HPI chained at month-over-month frequency and the Fisher HPI index chained at year-over-year frequency. The Jevons index is a geometric mean of posted price relatives, and does not incorporate any quantity weighting. It measures the average "within" price level change. The Fisher index reflects both the "within" price change and "between" price change, since it also incorporates the substitutions effects arising from cost-minimizing behavior by customers.

7. Conclusion

We develop empirical models of hedonic prices and derived hedonic prices for measuring the changes in customers' welfare, based upon deep learning. We generate product attributes or features (from text description and images) by deep learning models, and then use them to estimate the hedonic price function, utilizing modern

machine learning methods. Specifically, we convert the text information about the product to numeric product features using the ELMO or BERT neural network model, trained on Amazon's product descriptions. We convert the product image to numerical product features by a pre-trained ResNet50 neural network model. To produce the estimated hedonic price function, we use neural networks again. All the ingredients to the method rely on publicly available, open-source software components. Computing was done in Spark over EMR for data wrangling and Sage Maker (multi-core GPU) environment at the Amazon Web Services to train the neural network models.

We apply the models to Amazon's proprietary data on first-party sales in apparel to estimate the hedonic prices. The resulting hedonic models have a high predictive accuracy for several product categories, with the R^2 ranging from 80% to 90%. We construct the hedonic price indices for measuring the welfare of Amazon's customers and observe that over 2013-2017 the price index decreased, providing improvement in customers welfare.

Appendix A. Details of Different Architectures for Prediction Using Embeddings

Here we summarize different configurations for neural nets that we've tried.

Model 1: ELMO + Single Task Models. The first neural network model we tried is the Single Task model, i.e. predicting product prices one period at a time. For text, we used a pre-trained ELMO embedding. For images, we used pre-trained ResNet 50 embedding. The structure is shown in Figure 15. This neural network takes the pre-computed ELMO text embedding (of dimension 256) and the pre-computed ResNet50 image embedding (of dimension 2048) as input, transforms through 1 to 3 fully connected hidden layers with dropout, and a final linear layer maps the last hidden layer of neurons to the one-dimensional output, which is the predicted hedonic price. We trained one model for each time period, so in total there are T neural networks for T time period.

Model 2: BERT + Multitask model. The second neural network model that we tested is the Multitask NN with pre-trained BERT embeddings. The BERT embeddings are precomputed from a multi-lingual BERT model trained by Google. The input is ResNet50 image embeddings (of dimension 2048) and concatenated BERT sentence embeddings for title, brand, description and bullet points (of dimension $768 \cdot 4 = 3072$). The multitask NN has 1 to 3 dense layers and the output is a T dimensional vector, which represents the hedonic price for each of the T time periods.

Fine-Tuned BERT + Multitask model. In the last experiment, we used the end-to-end training framework to fine-tune a BERT model for hedonic price prediction. The model takes raw product text as input, tokenized using the WordPiece tokenizer and truncated / padded to a maximum sequence length d_S, and run through a BERT base model which consists of 12 transformer blocks. Then the sequence output (of dimension $d_S \times d_T$) from the transformer blocks is aggregated through a Global Average Pooling layer to product embedding (of dimension d_T). Then the product embedding is linearly mapped to the output which is T-dimensional hedonic price vector for T time periods. In our experiment, we use $d_S = 512$, and $d_T = 768$.

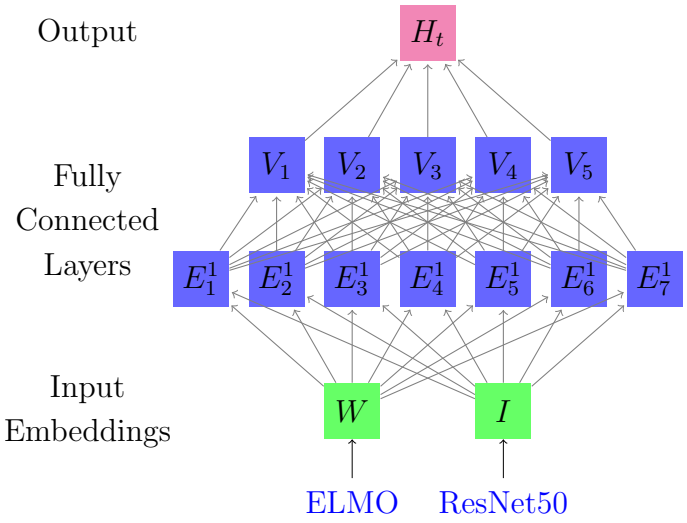

FIGURE 15. SingleTask + ELMO model. Product text is mapped to W and image is mapped to I of dimensions 256 and 2048. For illustration purpose, we only show two hidden layers with dimension 7 and 5 respectively. In practice, we use three layers with dimension 2048, 1024, and 256. The output is price for one time period t.

The loss function is the same as in Model 2, which combines the weighted squared error term and a regularization term that controls volatility. The weights from all or some layers of the transformer blocks are fine-tuned for the pricing task. Figure 17 is a simple illustration of the end-to-end BERT + Multitask model structure.

We are experimenting with different numbers of fine-tuned layers. Some initial results show that fine-tuning model improves model performance by a large margin, but it also takes much longer to train. This part of the work is not included in this paper, and we are continuing to explore this research direction.

REFERENCES

Bajari, Patrick, and C. Lanier Benkard. "Demand estimation with heterogeneous consumers and unobserved product characteristics: A hedonic approach." Journal of political economy 113.6 (2005): 1239-1276.

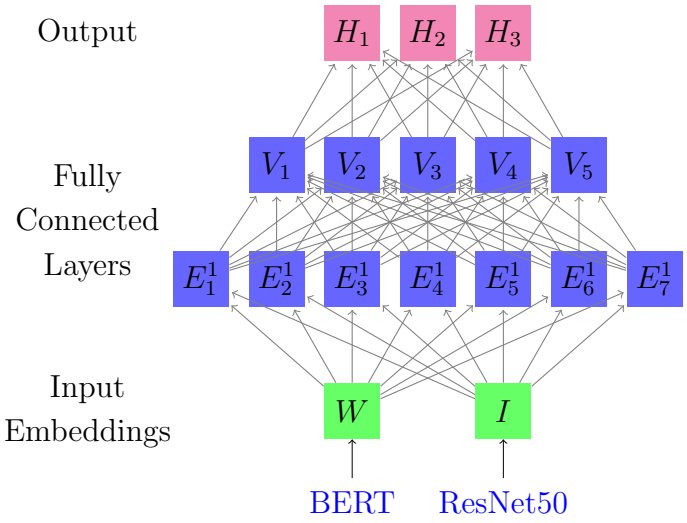

FIGURE 16. MultiTask + BERT. Product text is mapped to W and image is mapped to I of dimensions 3072 and 2048 respectively. For illustration purpose, we only show two hidden layers with dimension 7 and 5 respectively, and output is of dimension 3. In practice, we use three layers with dimension 2048, 1024, and 256, and the output is a price vector over $T = 72$ time periods.

Berry, Steven, Samuel Kortum, and Ariel Pakes. "Environmental change and hedonic cost functions for automobiles." Proceedings of the National Academy of Sciences 93.23 (1996): 12731-12738.

Berry, Steven, James Levinsohn, and Ariel Pakes. "Differentiated products demand systems from a combination of micro and macro data: The new car market." Journal of political Economy 112.1 (2004): 68-105.

Berry, Steven, James Levinsohn, and Ariel Pakes. "Automobile prices in market equilibrium." Econometrica: Journal of the Econometric Society (1995): 841-890.

Berry, Steven, and Ariel Pakes. "Estimating the pure hedonic discrete choice model." (1999).

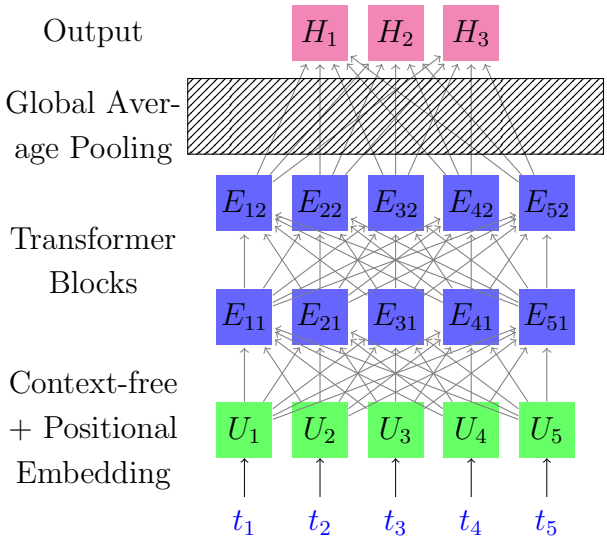

FIGURE 17. MultiTask + Fine-tuned BERT. Product text (sentence) is tokenized and padded to X, where components represent the context-free input embedding plus a positional encoding for a token (word). Then the input is fed into a BERT model which consists of 12 layers of transformer blocks, and outputs T dimensional price vector. For illustration purpose, we only show 5 tokens and two transformer blocks.

Benkard, C. Lanier, and Patrick Bajari. "Hedonic price indices with unobserved product characteristics, and application to personal computers." Journal of Business & Economic Statistics 23, no. 1 (2005): 61-75

Bils, Mark, and Peter J. Klenow. "Some evidence on the importance of sticky prices." Journal of political economy 112, no. 5 (2004): 947-985.

Boskin, Michael J., Ellen L. Dulberger, Robert J. Gordon, Zvi Griliches, and Dale W. Jorgenson. "Consumer prices, the consumer price index, and the cost of living." Journal of economic perspectives 12, no. 1 (1998): 3-26.

Boskin, Michael J., Ellen M. Dulberger, Robert J. Gordon, Zvi Griliches and Dale W. Jorgenson (1996) "Toward A More Accurate Measure of the Cost of Living, Final Report to the Senate Finance Committee from the Advisory Commission to Study

the Consumer Price Index". Washington DC: U.S. Government Printing Office; December 4.

Boskin, Michael J., and Dale W. Jorgenson. "Implications of overstating inflation for indexing government programs and understanding economic progress." The American Economic Review 87, no. 2 (1997): 89-93.

Breiman, Leo. "Bagging predictors." Machine learning 24, no. 2 (1996): 123-140.

Breiman, Leo. "Random forests." Machine learning 45, no. 1 (2001): 5-32.

Cavallo, AlBERTo, and Roberto Rigobon. "The Billion Prices Project." Journal of Economic Perspectives. Forthcoming (2016).

Cavallo, AlBERTo, and Roberto Rigobon. The distribution of the Size of Price Changes. No. w16760. National Bureau of Economic Research, 2011.

Chernozhukov, Victor, Alfred Galichon, Marc Henry, Brendan Pass, "Single nonparametric identification of multi-attribute hedonic equilibrium models," Journal of Political Economy, to appear (2021); arXiv:1709.09570.

Chen, Xiaohong, and Halbert White. "Improved rates and asymptotic normality for nonparametric neural network estimators." IEEE Transactions on Information Theory 45, no. 2 (1999): 682-691.

Chessa, A. G. and Griffioen, R. (2019). Comparing Price Indices of Clothing and Footwear for Scanner Data and Web Scraped Data. Economie et Statistique / Economics and Statistics, 509, 49–68. https://doi.org/10.24187/ecostat.2019.509.1984

Chiappori, Pierre-André, Robert J. McCann, and Lars P. Nesheim. "Hedonic price equilibria, stable matching, and optimal transport: equivalence, topology, and uniqueness." Economic Theory 42.2 (2010): 317-354.

Court, A.T. "Hedonic Price indices with automotive examples, in "The Dynamics of Automobile Demand", General Motors, New York, pp. 98-119 (1939)

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).

Diewert, W. E. Exact and superlative index numbers. Journal of econometrics, 4(2), 115-145. (1976).

Diewert, W. E. "Index number issues in the consumer price index." Journal of Economic Perspectives 12.1 (1998): 47-58.

Ekeland, Ivar, James J. Heckman, and Lars Nesheim. "Identification and estimation of hedonic models." Journal of political economy 112.S1 (2004): S60-S109.

Feenstra, Robert C. 1994. "New product varieties and the measurement of international prices." The American Economic Review, 157–177.

Farrell, Max H., Tengyuan Liang, and Sanjog Misra. "Deep neural networks for estimation and inference." Econometrica 89.1 (2021): 181-213.

Golosov, Mikhail, and Robert E. Lucas Jr. "Menu costs and Phillips curves." Journal of Political Economy 115, no. 2 (2007): 171-199.

Goodfellow, Ian, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

Goodman, Allen C. "Andrew Court and the invention of hedonic price analysis." Journal of urban economics 44.2 (1998): 291-298.

Goodman, Allen C. "Hedonic prices, price indices and housing markets." Journal of urban economics 5.4 (1978): 471-484.

Goolsbee, Austan D., and Peter J. Klenow. "Internet rising, prices falling: Measuring inflation in a world of e-commerce." Aea papers and proceedings. Vol. 108. 2018.

Goolsbee, Austan, and Peter J Klenow. 2006. "Valuing Consumer Products by the Time Spent Using Them: An Application to the Internet." American Economic Review, 96(2): 108–113.

Groshen, Erica L, Brian C Moyer, Ana M Aizcorbe, Ralph Bradley, and David M Friedman. 2017. "How Government Statistics Adjust for Potential Biases from Quality Change and New Goods in an Age of Digital Technologies: A View from the Trenches." Journal of Economic Perspectives, 31(2): 187–210.

Griliches, Zvi. "Hedonic price indices for automobiles: An econometric of quality change." The price statistics of the federal goverment. NBER, 1961. 173-196.

Han, S., Schulman, E., Grauman, K. and Ramakrishnan, S., Shapes as Product Differentiation: Neural Network Embedding in the Analysis of Markets for Fonts; ArXiv Preprint, 2020.

Hausman, Jerry. "Sources of bias and solutions to bias in the consumer price index." Journal of Economic Perspectives 17, no. 1 (2003): 23-44.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

Heckman, James J., Rosa L. Matzkin, and Lars Nesheim. "Nonparametric identification and estimation of nonadditive hedonic models." Econometrica 78.5 (2010): 1569-1591.

Ivancic, Lorraine, W Erwin Diewert, and Kevin J Fox. 2011. "Scanner data, time aggregation and the construction of price indices." Journal of Econometrics, 161(1): 24–35.

Kaplan, Greg, and Sam Schulhofer-Wohl. 2017. "Inflation at the household level." Journal of Monetary Economics, 91: 19–38.

Kidger, Patrick, and Terry Lyons. "Universal approximation with deep narrow networks." In Conference on Learning Theory, pp. 2306-2327. 2020.

Klenow, P. J., & Kryvtsov, O. (2008). State-dependent or time-dependent price: Does it matter for recent US inflation?. The Quarterly Journal of Economics, 123(3), 863-904.

Klenow, Peter J., and Benjamin A. Malin (2010). "Microeconomic evidence on price-setting." In Handbook of monetary economics, vol. 3, pp. 231-284. Elsevier, 2010.

Kingma, Diederik P., and Jimmy Ba. "Adam: A Method for Stochastic Optimization", ArXiv 1412.6980 : https://arxiv.org/abs/1412.6980

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." nature 521, no. 7553 (2015): 436.

McFadden, Daniel. "The measurement of urban travel demand." Journal of public economics 3.4 (1974): 303-328.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In Advances in neural information processing systems, pp. 3111-3119. 2013.

Moulton, Brent R. "The Measurement of Output, Prices, and Productivity." (2018). Brooking Institution.

Nesheim, Lars. "Hedonic price functions." CEMMAP, London (2006).

Office of National Statistics, UK. "Using statistical distributions to estimate weights for web-scraped price quotes in consumer price statistics", ONS, UK (2020).

Ohta, Makoto, and Zvi Griliches. "Automobile prices revisited: Extensions of the hedonic hypothesis." Household production and consumption. NBER, 1976. 325-398. Lancaster, Kelvin J. "A new approach to consumer theory." Journal of political economy 74.2 (1966): 132-157.

Pakes, Ariel. "A Reconsideration of Hedonic Price Indexes with an Application to PC's." American Economic Review 93, no. 5 (2003): 1578-1596.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep contextualized word representations." arXiv preprint arXiv:1802.05365 (2018).

Schmidt-Hieber, Johannes. "Nonparametric regression using deep neural networks with ReLU activation function." Annals of Statistics 48.4 (2020): 1875-1897.

Stock, James H. "Nonparametric policy analysis." Journal of the American Statistical Association 84.406 (1989): 567-575.

Stock, James H. "Nonparametric policy analysis: an application to estimating hazardous waste cleanup benefits." Nonparametric and Semiparametric Methods in Econometrics and Statistics (1991): 77-98.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." Cvpr, 2015.

Targ, Sasha, Diogo Almeida, and Kevin Lyman. "Resnet in Resnet: generalizing residual architectures." arXiv preprint arXiv:1603.08029 (2016).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

Yarotsky, Dmitry. "Error bounds for approximations with deep ReLU networks." Neural Networks 94 (2017): 103-114.

Zeng, Shipei. "Hedonic Imputation with Tree-based Decision Approaches"; Preprint, December 2020; School of Economics, University of New South Wales