HEDONIC PRICES AND QUALITY ADJUSTED PRICE INDICES POWERED BY AI

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

ABSTRACT. We develop empirical models that efficiently process large amounts of unstructured product data (text, images, prices, quantities) to produce accurate hedonic price estimates and derived indices. To achieve this, we generate abstract product attributes (or "features") from text descriptions and images using deep neural networks. These attributes are then used to estimate the hedonic price function. To demonstrate the effectiveness of this approach, we apply the models to Amazon's data for first-party apparel sales, and estimate hedonic prices. The resulting models have a very high out-of-sample predictive accuracy, with R^2 ranging from 80% to 90%. Finally, we construct the AI-based hedonic Fisher price index, chained at the year-over-year frequency, and contrast it with the CPI and other electronic indices.

Key Words: Hedonic Prices, Price Index, Transformers, Deep Learning, Artificial Intelligence

Date: March, 2019. This version: November 1, 2024.

*P. Bajari, Z. Cen, V. Chernozhukov, M. Manukonda, S. Vijaykumar, and J. Wang (in alphabetical order) are the principal authors who contributed to the current version of this paper. R. Huerta and G. Monokrousos were also principal contributors to an earlier version of the paper. For helpful comments, we would like to thank the participants at the 2018 Federal Economic Statistics Advisory Committee meeting, the 2019 Allied Social Science meetings, the 2019 Federal Reserve Board conference on "Nontraditional Data, Machine Learning, and Natural Language Processing in Macroeconomics", the 2019 Brookings Conference "Can big data improve economic measurement?" and the 2021 CEMMAP conference "Measuring prices and welfare," and seminars at Berkeley, European Bank of Reconstruction and Development, MIT, UCL, and York. We are grateful to Andrew Chesher, Greg Duncan, Kevin Fox, John Haltinwager, James Heckman, Daniel Miller, Mathew Shapiro, Bernhard Schölkopf, James Stock, and Weining Wang for helpful comments during the various stages of this project. The paper was first publically circulated in 2021 as a CEMMAP Working Paper CWP04/21.

1. INTRODUCTION

Economists and policy-makers rely on price indices, such as the Consumer Price Index (CPI), to measure inflation, consumer welfare, and the cost of living. As a result, the methods used to construct price indices warrant considerable attention. Two of the most widely used methods are the Laspeyres and Paasche indices, which measure changes in the cost of a standardized basket of products between two periods, where the basket is chosen to represent aggregate demand in either the initial period (Laspeyres) or the final period (Paasche). The two are often combined into what is known as the Fisher Price Index (FPI).[1] The FPI has been shown to accurately approximate the relative cost of living between two periods when the difference in prices is small.[2]

A common problem with the aforementioned indices is product entry and exit, where the previous period's prices are not available for new products and vice-versa. Consequently, economists often produce so-called 'matched indices': indices restricted to the set of products that are bought and sold in both periods. This introduces selection bias because products exiting the marketplace may not resemble those that remain, especially when products turn over quickly. To mitigate this bias, economists often use high-frequency chaining combined with compounding: for example, computing price indices at a monthly frequency and then compounding monthly inflation rates to get yearly rates. This approach does mitigate the turnover problem if the rate of turnover is not high from month to month. Still, it can suffer from chain-drift bias—bias in estimated price levels due to geometric compounding of measurement errors over many steps.

[1] These quantities bound other measures of inflation/deflation based on the expenditure function under certain assumptions; see Diewert (1998) and also Diewert and Fox (2022) for a state-of-art review of price indices.

[2] More precisely, the FPI captures the exact cost of living for a representative consumer with quadratic utility or expenditure functions, and provides a second-order approximation for any smooth utility or expenditure function under small price changes (see Diewert 1976). Price indices with this property are called superlative indices, with another prominent example being the Tornqvist index (see, e.g, Office of National Statistics, 2020). For both matched and hedonic indices, the Tornqvist indices we obtain are numerically very similar to the corresponding (matched or hedonic) Fisher indices. All of the discussion and results we present using the Fisher index carry over to the Tornqvist index.

Hedonic price models were introduced by Court (1939) and Griliches (1961); these models postulate that the prices of differentiated products are determined by the market value of each product's underlying characteristics. Court and Griliches suggested measuring inflation or deflation by modeling how hedonic prices change while holding product characteristics fixed. Importantly, one can compute hedonic prices for a basket of goods at any time point—despite entry and exit—because such prices are determined only by product characteristics. These prices can then be used in the Fisher price index and other price index formulas, giving rise to hedonic price indices, which are regularly employed both in academic research and by statistical agencies (e.g., Wasshausen and Moulton, 2006; Office of National Statistics, 2020).

The resulting hedonic price indices are sometimes called *quality-adjusted* price indices because we implicitly fix the set of characteristics ("qualities") of a basket in a reference period and compute the ratio of costs of the basket in the comparison period and in the reference period. The ability to compute hedonic prices at any time point for any product allows us to address the entry/exit problem and also reduce chain-drift biases by making "long" or "low-frequency" comparisons (year-to-year, for example).[3] The success of this approach depends on both the ability of the hedonic price models to approximate real-world prices and on our ability to accurately estimate the hedonic price function.

Provided that hedonic models provide a good approximation of real world supply and demand, the theory suggests we can explain equilibrium prices by regressing observed prices on product characteristics X_j. In traditional empirical hedonic models, the construction of X_j is performed using human expertise, and statistical agencies perform the data collection through extensive field surveys and interviews.

In this paper, we develop an alternative approach to building hedonic models. Our method uses electronic data and AI tools to collect real time data and automatically generate product characteristics X_j, which are used to accurately predict prices. The resulting approach is highly scalable and has a very low cost compared to the traditional approach. There is also significant interest in using electronic records to improve the

[3]Year-to-year chaining is recommended in the CPI manual (ILO et al., 2004) to deal with chain drift. This approach was used, for example, in Handbury et al. (2013) to construct non-hedonic indices using electronic data from Japan. See Diewert and Fox (2022) for pertinent discussion and other methods used to address chain drift. One prominent class of alternative methods are multilateral indices—including the GEKS index—which average many matched indices with varying base periods.

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

construction of price indices, both from academic researchers and statistical agencies (see e.g. Groves and Harris-Kojetin, 2017; Ehrlich et al., 2019; Jarmin, 2019; Lebow, 2023). Our approach thus offers a valuable complement to traditional methods for measuring price-level changes.

1.1. **Deep Learning and Structured Sparsity.** Let us illustrate the problem we are solving. Figure 1 shows the product characteristics visible to customers, including the product title, description, and images. 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 predict prices accurately. 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 by means of a low dimensional numerical representation, X_j. Successful experts did produce low-dimensional representations for certain groups of products, which proved to be successful in building hedonic models (e.g., Pakes, 2003 reports very high accuracy for predicting computer prices). However, this approach does not scale well to many types of products and is prone to judgment biases. These issues

raise important questions: when human experts do succeed, what is the underlying reason?; can we replicate this success with artificial intelligence, and can these methods deliver scalable inference?

Arguably, humans can easily summarize the red dress depicted in Figure 1 and its accompanying text, even though the original representation of this information lives in an extremely high-dimensional space. Indeed, the image consists of nearly one million pixels (three layers of 640 x 480 pixels encoding the blue, red, and green color channels), while words belong to a dictionary whose dimension is in the tens of thousands[4] and sentences live in much higher-dimensional space. However, we believe that information in images and sentences can be effectively represented in a much lower-dimensional space, a phenomenon we call "structured sparsity." Human intelligence can exploit this structured sparsity to process information effectively—perhaps using the geometry of shapes in images, the relative simplicity of color schemes and shade patterns, the similarity of many words in the dictionary, and the context-specific meaning of words. The field of artificial intelligence (AI) developed neural networks to mimic human intelligence in many information-processing tasks. These models do create parsimonious structures from high-dimensional inputs and often surpass human ability in such tasks. Going forward, we will employ state-of-the-art solutions from AI to the problem of hedonic modeling.

In this work we extract relevant product attributes or features from text and images using deep learning models, then use these features to estimate the hedonic price function. We convert text information about the product to numeric features (embeddings) using the BERT model of Devlin et al. (2018), a transformer-based large language model; our version of the model has been fine-tuned on Amazon's product descriptions and prices. We similarly use a pre-trained ResNet50 model, a convolutional neural network used to understand images, to produce embeddings for product images (He et al., 2016). For context, these models were initially trained to comprehend text and images in tasks unrelated to predicting prices (e.g., image classification or predicting a missing word in a sentence). We then take the internal numeric representation generated by each

[4]See, e.g., Milton and Treffers-Daller (2013).

model as an information-rich, parsimonious *embedding* of the input.[5] With these embeddings, we then estimate the hedonic price function using a neural network. In particular, we design *multi-task* networks, which predict a complete time-series of hedonic prices $(\hat{H}_{it})_{t\in T}$ for a given product i, where $t \in T$ covers all time periods in the study.[6]

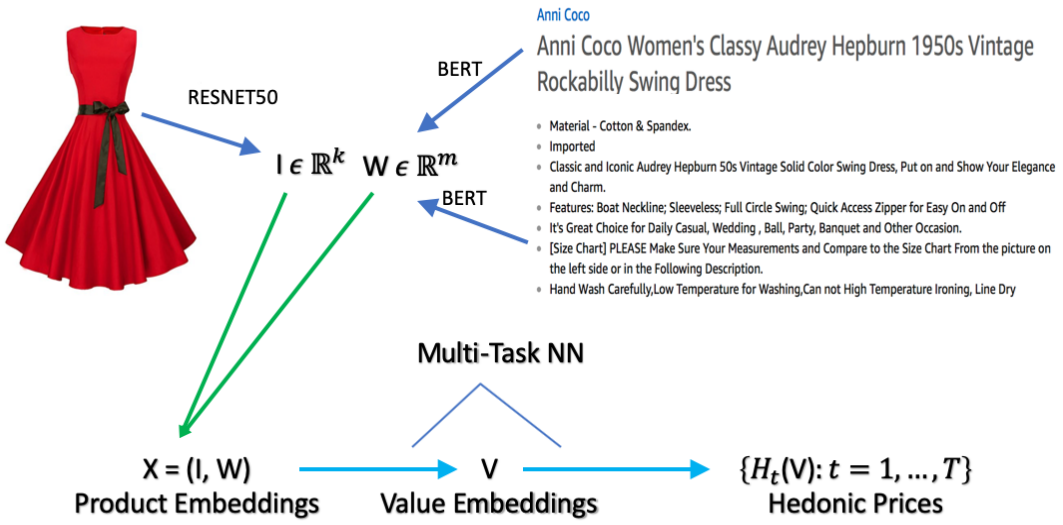

FIGURE 2. Our method for generating hedonic price: The input consists of images and unstructured text data. The first step of the process creates the moderately high-dimensional numerical embeddings I and W for images and text data via state-of-the-art deep learning methods, such as ResNet50 and BERT. The second step of the process takes input $X = (I, W)$ and creates predictions for hedonic prices $H_t(X)$ using deep learning methods with a multi-task structure. Our multitask model creates an intermediate lower dimensional embedding $V = V(X)$, called value embedding, and then predicts the final prices in all periods $\{H_t(V), t = 1, ..., T\}$ using linear functional forms, making it easy to perform inference on the last step using hold-out samples.

1.2. **Deriving Indices using Amazon Data.** We apply these models to Amazon's data for first party sales[7] apparel sales to estimate hedonic prices. The resulting models have

[5]This strategy follows a paradigm called "transfer learning," which has proved a very successful way to use neural networks in new domains (Ng, 2016).

[6]See Section 2 for a comparison to other models, and Appendix C for a discussion of alternative neural network designs.

[7]First party sales are those in which Amazon first buys the product from a trusted seller and then sells it to the consumer.

high predictive accuracy, with R^2 in the hold-out sample ranging from 80 to 90%. Therefore, our approach can 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 hedonic price models provide a good, first-order approximation for real-world prices.

We then proceed to construct Fisher hedonic price indices (FHPI) over 2013-2017, constructing monthly-chained, yearly-chained, and the geometrically combined FHPI (GFHPI).[8,9] We focus discussions on yearly chained FHPI and refer to it as the Fisher hedonic price index unless stated otherwise. We compare this index with

- the matched (repeated sale) Fisher index,
- the posted price Jevons index, which is the geometric mean of relative prices, chained at a daily frequency,
- the BLS Urban CPI index for apparel (CPI), constructed by the Bureau of Labor Statistics,
- 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 FHPI is -.77%, by the Jevons index -2.82%, and by the matched Fisher index -3.46%. In comparison, the annual rate of inflation in apparel prices estimated by the Adobe DPI is -1.3%, and by the CPI is -.04%.

The yearly chained FHPI, our preferred index, has the smallest discrepancy with the CPI. Part of this remaining difference with the CPI could be attributed to the limited

[8]The GFHPI gives equal geometric weight to yearly and monthly chained indices. This brings in some seasonality, allowing the GFHPI to reflect within-year price changes. This index should not be used over long horizons, however, due to chain drift.

[9]This approach was inspired by the GEKS index, which computes the geometric average over all chains; see e.g., Diewert and Fox (2022) for a precise definition. The GEKS index reduces chain drift caused by measurement error in the product shares used to construct the monthly index. We are currently exploring the use of hedonic GEKS indices as follow-up work.

ability of the CPI to address quality change and substitution (as pointed out by Moulton (2018) and others), specifics of product categorization in the apparel segment, and differences in the composition of baskets that consumers purchase via Amazon. The difference may also accrue due to Amazon-specific cost improvements and supply-chain logistics.

In comparison to the FHPI, neither the matched Fisher index nor Adobe's DPI adjusts for change in the quality of goods sold. These indices potentially also suffer from chain drift—the systematic accumulation of errors due to frequent compounding. Compared to the FHPI, the Jevons index does not incorporate quantity weighting, and is also subject to chain drift.[10] Both of these issues cause rather a large discrepancy between the Jevons index and the FHPI.

1.3. **Contributions to the literature.** We view our paper as an original contribution towards the modernization of hedonic price models and their application to large-scale data. In this way, we also contribute to the literature in empirical microeconomics **[sv: should this be macroeconomics?]** dedicated to hedonic price indices and their uses in measuring inflation.

1.3.1. *AI-based hedonic models.* At a high level, our work stems from the observation that hedonic modeling is fundamentally a prediction problem: "how much demand will there be for characteristics X_j at time t, based on the demand for similar products?" As such, it presents an excellent opportunity for AI to help us measure policy-relevant economic variables, such as the rate of inflation (Mullainathan and Spiess, 2017). This is especially significant given the massive amount of unstructured data available in the form of electronic records, which are not readily available to conventional models.

We are unaware of any prior work in this area that develops large-scale hedonic price models from unstructured product text and image descriptions. In addition, our data are unique in that they cover the universe first party transactions in the Amazon Store from 2013 to 2019. From these models and data, we generate interesting findings: we

[10]Chain drift due to measurment error is less severe for the Jevons index—at least theoretically—because the index is computed using very many prices so that within-period errors are small. The Jevons index is a special case of the Tornqvist index with revenue shares for products set to be equal. This reduces another important source of chain drift due to by bouncing shares, as pointed out by Ivancic et al. (2011).

Index	CPI	BPP	DPI	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
Long Chaining	Yes	No	No	Yes

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

demonstrate the power of hedonic models to characterize prices and document the decline in the quality-adjusted Fisher price index for apparel. To the best of our knowledge, there are very few related studies in economics: An independent and contemporaneous work by Zeng (2020) develops a related approach to hedonic prices using scanner data, but uses random forest methods and low-dimensional data instead of neural network based text and image embeddings. An independent and contemporaneous work by Han et al. (2021) explores the use of the image embeddings to characterize typesetting fonts as products, and analyzes the effect of mergers on product differentiation via font producers' design choices. In the coming years, we expect to see see a much wider use of AI-based embeddings for text and images to power empirical research in economics.

1.3.2. *Inflation, e-commerce, and electronic data.* This paper contributes to a rapidly growing literature on using electronic data and techniques to measure inflation and other aggregate quantities. This motivated by the significant growth of e-commerce along with evidence that online prices may fluctuate differently than their offline counterparts, discussed below.

The MIT Billion Prices Project (BPP) constructs Jevons indices using web-scraped retail price data and directly-provided retailer data. The advantages of such data 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 and Rigobon (2011, 2016) study several countries and establish that inflation measures constructed using such online data can differ substantially from official government statistics on consumer prices (the most extreme example being the case of Argentina, see Cavallo, 2013).

A recognized limitation of this approach is that it does not incorporate quantity information. Goolsbee and Klenow (2018) constructed matched Törnqvist indices, including the Digital Price Index (DPI), using Adobe Analytics data from e-commerce clients of Adobe (which notably include quantities as well as prices). This approach overcomes the quantity limitation of the BPP index and thus can account for substitution resulting from consumers minimizing costs. They find that for the US online DPI inflation is substantially lower (for the period 2014-2019) than the official CPI inflation for the categories they study. This is analogous to our findings for apparel, albeit the discrepancy between our preferred index and the CPI is much smaller. The difference with our approach could possibly occur due to chain drift in the DPI created by month-to-month chaining.

There is also a body of research on using scanner data to survey demand and to construct measurements of inflation; of particular note are Handbury et al. (2013), Ivancic et al. (2011), Leicester (2014), Diewert and Fox (2022), and Office of National Statistics (2020). We contribute to this development by providing AI-based hedonic prices that economists can employ alongside the approaches in these studies. This provides a valuable complement since scanner data are generally tilted towards food and beverages sold in grocery stores (Kaplan and Schulhofer-Wohl, 2017).

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). Gorodnichenko and Talavera (2017), Cavallo (2018) and Gorodnichenko et al. (2018) find that online prices change more frequently than offline prices, thereby responding to competition more promptly. These papers also find that online prices exhibit stronger pass-through in response to nominal exchange-rate movements than prices found in official CPI data; such results have important implications for the price stickiness literature and the law of one price. These results also highlight the potential benefit of real-time, electronic data and derived price indices like ours.

Since the initial circulation of this manuscript as a CEMMAP Working Paper (Bajari et al., 2019), there has been substantial and very promising research that adapts our methods to other, publicly available data sources, with the ultimate aim of complimenting or improving national statistics. Of particular note are the working papers of Cafarella et al. (2023) and Ehrlich et al. (2023), both of which discuss the use of ML-based hedonic indices with point-of-sale scanner data.

1.4. **Organization of the Paper.** We organize the rest of the paper as follows. Section 2 defines the hedonic price models and price indices. Section 3 discusses modeling and estimating the hedonic price functions via Neural Networks (NNs). Section 4 provides a non-technical description of the process of obtaining product embeddings via AI tools. Section 5 examines the empirical performance of the AI-based hedonic price functions and constructs the AI-based hedonic price indices. Appendix A provides a technical description of BERT, a large language model used to generate embeddings for the product description. Appendix B gives a technical description of ResNet50, a tool used to generate embeddings of the product's image. Appendix C gives a technical description of various models we've considered and tested, of which the best-performing methods are described in the main text.

1.5. **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 denote parameter values, with the exception of ϵ which denotes the regression error (see below).

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 a predictive model for the price given the product features:

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

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 will use the data from time period t to estimate the function h_t using modern nonlinear regression methods, such as deep neural networks. We contrast this approach with classical linear regression methods as well as other modern regression methods, such as a random forest. The key component of our approach is the generation

of product features X_{it} using neural network 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 detailed in the next section.

There is a substantial body of economic research on hedonic price models. On the theory side, economists have developed a theory of supply and demand in terms of product characteristics (Lancaster, 1966; McFadden, 1974; Rosen, 1974; Gorman, 1980); they have also established existence of the hedonic price function and characterized its behavior under various assumptions (Berry et al., 1995, 2004; Ekeland et al., 2004; Benkard and Bajari, 2005; Chiappori et al., 2010; Chernozhukov et al., 2020); they have developed the use of hedonic prices for bounding changes in consumer 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 their marginal willingness to pay for certain characteristics, see Nesheim (2006) and Greenstone (2017); they have also used hedonic theory to measure the value of non-tradable goods (for example, measuring the effects of air quality or hazardous waste cleanup on housing prices, e.g. Chay and Greenstone 2005 and Stock 1991, or equalizing differences in the labor market, e.g. Brown 1980). Our main use of hedonic prices is to estimate the rates of deflation (or inflation) for apparel products purchased at Amazon.com. This follows prior work on using hedonic models to construct official price statistics (see e.g., Griliches, 1961; Pakes, 2003; Wasshausen and Moulton, 2006; Office of National Statistics, 2020), but with a major deviation: we use product features engineered using deep learning instead of using human experts to tabulate product characteristics, and we also estimate the hedonic price function using deep learning rather than using classical regression methods.

The literature typically specifies three building blocks of theoretical hedonic models: consumer utility functions over the characteristics of products (rather than over products themselves) and over the consumer's characteristics; producer cost functions over characteristics of the good and of the producer; and an equilibrium assumption (or existence is shown as a part of the analysis); see e.g. Pakes (2003) and Rosen (1974). This

determines prices (and quantities) given demand and costs, and establishes the 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:

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

where p is the price of the product to be paid by the customer and m is their income, the first order conditions for the utility maximization problem $\max_{(x,u)} V_0(x,u)+m-H(x,u)$ 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 (for continuously varying attributes). Therefore, under suitable assumptions, a standard argument for identification of the average derivative of a structural function gives

$$\mathrm{E}[\partial_{x_k} H^\star(X_j,U_j)|X_j] = \partial_{x_k} h_t(X_j).^{11}$$

In other words, the average marginal willingness to pay for a given characteristic is equal to the average derivative of the hedonic price map, and is identified by the derivative of the hedonic regression function. We remark here that the expectation is taken over the random variables' distribution at equilibrium; we can not say anything about features of the hedonic price map away from equilibrium.

Given a parametric form of the consumer's utility, preference parameters can be recovered from first-order conditions provided that either (i) $(x,u) \mapsto H^\star(x,u)$ 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 (ii) 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, 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, $H^\star(X,U) = H_0^\star(X) + U$, we have $\alpha_k = \partial_{x_k} h_t(X_j) X_{jk}$ for the consumer who has purchased product j. Hence distributions of taste parameters α for consumers can be recovered under such modeling approaches; see Bajari and Benkard (2005) for further relevant discussion. In this paper

[11] This holds, for example, under independence of observable and unobservable characteristics at equilibrium, see Appendix D.1.

we have a different goal: we will use hedonic prices to construct indices and thereby measure price level changes, following accepted practice in applied economics and in the work of statistical agencies (e.g., Wasshausen and Moulton, 2006; Pakes, 2003; Office of National Statistics, 2020).

2.2. **Price Indices: Hedonic vs Matched.** We focus on hedonic price indices and compare them to matched (repeated sale) price indices. The matched price index tracks changes in the price of a basket of products sold in both the base period and in subsequent time periods. While the matched price method is subject to selection bias (due to product entry and exit), it ensures the index tracks goods from a common pool of products. A major shortcoming of this method is that the common pool of products across time can be small and non-representative; this will be apparent in our data.

The hedonic price index replaces transaction prices with predicted values using a rich set of product characteristics (obtained using a combination of AI and machine learning methods). In principle, the hedonic approach captures changes over time in the value consumers assign to product attributes.[12] Hedonic approaches are especially helpful for predicting the 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 products.

We consider three broad types of 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 weighting 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 $t-\ell$:

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

[12] We refer the reader to Aizcorbe (2014) for further discussion of hedonic price indices and their application.

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, $\mathscr{C}_t$ is the set of all products with transactions at time t, $\mathscr{C}_t \cap \mathscr{C}_{t-\ell}$ is the match set, the set of all products with transactions both at time t and at time $t-\ell$.

Within a representative consumer model, basic economic intuition suggests that matched indices should obey the order restriction: $R^{L,M} \geq R^{P,M}$. Even when aggregate quantities are not described by a representative consumer model, the relation often holds empirically. Similar intuition applies to hedonic indices. One way to aggregate the two indices is to use Fisher's ideal index, which is a superlative index: it measures the 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, 1976).

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 $t-\ell$:

$$R_{t,\ell}^{P,H} = \frac{\sum_{i\in\mathscr{C}_t} H_{it}Q_{it}}{\sum_{i\in\mathscr{C}_t} H_{i(t-\ell)}Q_{it}};\;\; R_{t,\ell}^{L,H} = \frac{\sum_{i\in\mathscr{C}_{t-\ell}} H_{it}Q_{i(t-\ell)}}{\sum_{i\in\mathscr{C}_{t-\ell}} H_{i(t-\ell)}Q_{i(t-\ell)}};\;\; R_t^{F,H} = \sqrt{R_t^{P,H} \cdot R_t^{L,H}}.^{13}$$

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

For an arbitrary index $R_{t,\ell}^{\bullet,\bullet}$, for positive integers t, ℓ, we measure the price changes up to time $t \geq t_0$ as follows. Let $m = \lfloor (t-t_0)/\ell \rfloor$ denote the largest integer no greater than $(t-t_0)/\ell$, and write $t - t_0 = m\ell + r$. We construct a chained index by taking the product

$$R_{t,\ell}^{\bullet,\bullet,C} = R_{t,r}^{\bullet,\bullet} \prod_{\tilde{m}=1}^{m} R_{t_0+\tilde{m}\ell,\ell}^{\bullet,\bullet} \quad \text{where} \quad R_{t,0}^{\bullet,\bullet,C} = 1.^{14}$$

[13] When hedonic prices follow a linear model $H_{it} = \theta_t' V_i$, where V_i are product attributes, we have $\sum_{i\in\mathscr{C}_t} H_{is}Q_{it} = \theta'(\sum_{i\in\mathscr{C}_t} V_i Q_{it})$. Thus, the index may be viewed as measuring the relative price of a quantity-weighted collection of product attributes.

[14] For long-chained indices with $\ell > 1$, this approach compares a given period's prices to the most recent "pivot" $\{t_0, t_0+\ell, t_0+2\ell, \cdots\}$. This is a standard practice, see e.g. Statistics Bureau of Japan (2022, Appendix 4). Resulting indices are numerically very robust to using alternative long-chaining methods.

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, capturing better seasonal price changes for apparel. The second index is less susceptible to the chain-drift problem that arises from an accumulation of errors due to repeated compounding. For this reason, we choose the yearly-chained index to be our preferred index, and refer to it as the Fisher Hedonic Price Index (FHPI). To capture seasonality while mitigating chain drift, we also consider the geometric mean of the monthly-chained and yearly-chained index (GFHPI): $R_t^{GF,H} = \sqrt{R_{t,1}^{F,H,C} R_{t,12}^{F,H,C}}$.

3. Price Prediction and Inference with Deep Neural Networks

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

Our general nonlinear regression model is a composition of several nonlinear, vector-valued functions, called *layers*. It takes the form

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

Here Z_i is the original input, which lies in a very high-dimensional space. Z_i is non-linearly mapped, via the embedding layer e, to an embedding vector X_i which is of moderately high dimension (up to 5120 dimensions). This embedding is again non-linearly mapped to a lower dimension vector $E_i^{(1)}$ by the first hidden layer g_1, and so on, until the final hidden layer g_m. The output of the final hidden layer g_m, given by $V_i := E_i^{(m)}$, 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 output of the final hidden layer, $V_i = E_i^{(m)}$, is called the *value embedding* in our context. It is a moderately high-dimensional summary of the product (up to 512 dimensions). It is derived from product attributes, and directly determines the predicted

hedonic price of the product. Note that the embeddings V_i do not depend on time and thus represent the intrinsic, potentially valuable attributes of the product. However, the predicted price does depend on time t via the coefficient θ_t, reflecting the fact that these intrinsic attributes are valued differently across time.

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

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

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.[15] 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 $\sigma(v) = v$. Within a given layer, individual neurons' activations can be linear or non-linear. The use of a non-linear activation function has been shown to be an extremely powerful tool for generating flexible functional forms, both yielding successful approximations in a wide range of empirical problems and backed by approximation theory. Good approximations can be achieved by considering sufficiently many neurons and layers (e.g., Chen and White 1999; Yarotsky 2017; Kidger and Lyons 2020).

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

The model can be trained by minimizing the loss function

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

where $\eta = (g_1, ..., g_m)$ denotes all of the parameters of the mapping

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

Here we are weighting by the quantity Q_{it}. Regularization can be used to limit fluctuations of predicted prices across time. This is done by adding a penalty to the loss function:

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

[15] The standard architecture has an 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 non-linearly activated.

where the penalty level λ is chosen to yield good performance in validation samples.

Next, we give an overview of how the initial embedding is generated. A multilingual BERT model is used to convert text information into a sub-vector W_i of $E_i^{(1)}$, and likewise a ResNet50 model is used to convert images into another sub-vector I_i of $E_i^{(1)}$ (both models are publicly available; see Section 4 below). These models are trained on auxiliary prediction tasks with auxiliary output A_{T_i} for text and A_{I_i} for image, which can be illustrated diagrammatically as:

$$Z_i = \begin{bmatrix} \text{Text}_i \\ \text{Image}_i \end{bmatrix} \overset{e}{\longmapsto} X_i := \begin{bmatrix} W_i \\ I_i \end{bmatrix} \begin{matrix} \uparrow A_{T_i} \\ \\ \downarrow A_{I_i} \end{matrix} \mapsto E_i^{(1)} \mapsto \cdots \mapsto E_i^{(m)} := V_i \overset{\theta'}{\longmapsto} \{H_{it}\}_{t=1}^{T}. \tag{7}$$

The text and image embeddings W_i and I_i, which form X_i, are obtained by mapping them to auxiliary outputs A_{Tj} and A_{Ij} that are scored on natural language processing tasks and image classification tasks respectively. This step uses data unrelated to prices and its aim is to produce high-quality embeddings W_i and I_i for general text and image data, as we elaborate upon in Section 4. Finally, we further fine-tuned parameters of the mapping generating W_i for price prediction tasks, which yielded some improvements.[16]

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 the price prediction task, we used the Adam algorithm (Kingma, 2014). At a high level, training involves randomly splitting the set of products into three subsets: training (60%), testing (20%) and validation (20%). This split occurs exclusively across products i, while preserving each product's time-series of prices and quantities. The training and validation sets are used by the Adam algorithm to minimize the penalized loss given in Equations (5) and (6), while the testing set is used to measure predictive performance in Section 5. The overall process has many tuning parameters; in practice, we chose them by cross-validation. The most important choices concerned the number of neurons and the number of neuron layers.

[16] We did not attempt to fine-tune image embeddings I_i.

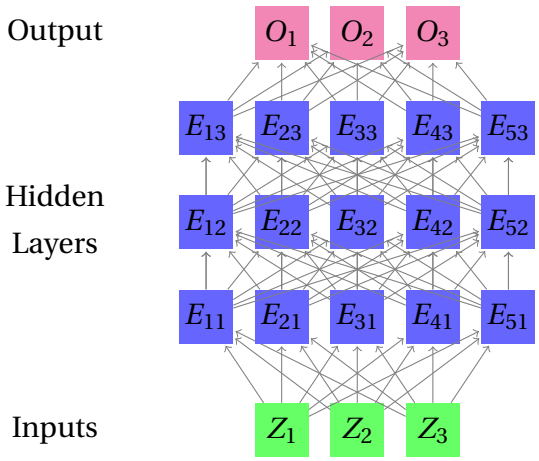

FIGURE 3. Standard architecture of a Deep Neural Network. In the hedonic price prediction network, the penultimate layer is interpreted as an embedding of the product's hedonic value and the output layer contains predicted hedonic prices in all time periods. In comparison, the networks used for text and image processing have very high-dimensional inputs and outputs, with intermediate hidden layers composed of neural sub-networks. The dense embeddings typically result from taking the last hidden layer of the network.

We can 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 outputs, the predicted values. In the first row we see the inputs, and in the second row we see the first layer of neurons. The neurons are connected to the inputs, and the connections represent coefficients. Finally, the last layers of neurons are combined to produce a vector of outputs. The coefficients θ_m are shown by the connections between the last hidden layer of neurons and the multivariate output. Networks with vector outputs are called multi-task networks.

Prediction methods based on neural networks with many layers of neurons are called deep learning methods. Neural networks recently emerged as a powerful and all-purpose method for a wide range of problems—ranging from predictive and classification analysis to various natural language processing tasks. 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 in large data settings. In Section 4 and Appendix, we overview the ideas and details of the neural networks for dealing with text and images.

3.2. **Assessing Statistical Significance and Confidence Intervals.** In many settings, researchers may wish to construct standard errors for predicted prices or for model coefficients. To this end, the last layer of the neural network 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:

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

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

$$\hat{H}_{it} = V_{it}'\hat{\theta}_t.^{17} \tag{8}$$

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

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

Remark 1. The 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 to be frozen conditional on the training and validation samples). Following Chernozhukov et al. (2018), one way to account for this variability is to consider multiple random splits, $s = 1, ..., S$, of the data into test, training and validation

[17]The resulting predictions $\hat{H}_{it}$ can differ from those of our original model; the main purpose of estimation is to construct confidence intervals for predictions and to test significance of coefficients. When the training and test datasets are many orders of magnitude larger than the dimension of V_{it}—as in our setting—the differences are generally small.

subsets (stratified by month). Different splits would result in different value embeddings V_{it}^s, coefficient estimates $\hat{\theta}_t^s$, predicted hedonic prices, and covariance estimates $\widehat{\text{Cov}}(\hat{\theta}_t)^s$, as well as confidence intervals $[U_{it}^s, L_{it}^s]$. Then we can aggregate the estimates and confidence intervals as follows:

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

The adjusted nominal level for the confidence interval for $V_{it}'\theta_s$ is $1-\alpha/2$. Similarly, for judging the 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 exploits the fact that the median of arbitrarily correlated variables whose marginal distributions are standard uniform 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.

4. Image and Text Embeddings Via Deep Learning

Typically, customers view products as depicted in Figure 1, where details such as product title, description, and images are presented. Our task is to convert the product characteristics, such as the product title, description, and image, into numerical vectors, which can be used for constructing hedonic prices, as shown in Figure 2 given in the introduction. 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 embeddings and give a non-technical description of the main ideas. We will mostly focus on the transformer-based large language model – BERT, one of the most successful text embedding algorithms. We give a more technical review of BERT and earlier algorithms (Word2Vec, ELMO) in the Appendix.

4.1.1. *High-Level Objectives.*

First, we would like to stress that the high-level objective of the text-embedding algorithms is to construct a concise (low-dimensional) representation of the dictionary of words and derive a concise representation for text sentences, such as product titles and descriptions.

Conceptually, the j-th word in the product description can be represented by a binary encoding of a very high dimension d. Still, this representation is not very useful because it is too sparse, and it is not able to explore word similarity to compactly approximate the dictionary. Indeed each distinct word has a distance of 1 from another distinct word.

Instead, we aim to represent words by vectors of much lower dimension r, such that the distance between similar words is small. Denote such potential 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 the 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.

Text embedding algorithms aim 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, such as predicting an omitted word in a sentence from surrounding words or detecting when two sentences are presented in reverse order, using many examples based on the corpus of the published text (ranging from “small" data, such the entirety of Wikipedia, to a large fraction of all digitized text, used in training GPT-3). These tasks are not related to hedonic prices—but precisely because they are not related, one can generate extremely large data sets of examples on which the model can be trained.

Once the embeddings for words have been obtained, we can generate the embedding for the title or description of product i, containing the embedded words $\{U_{j,i}\}_{j=1}^{J}$ by taking averages:

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

where λ_j's are weights given to the jth word. A simple choice is fixed weight $\lambda_j = 1/J$, but one can also use data-dependent weights (see the Appendix for details). One may also use the “brute force" approach and concatenate the embeddings: $W_i = \text{vec}(\{U_{j,i}\}_{j=1}^{J})$. This leads to rJ-dimensional embedding, but as long as rJ is not overly large, it remains practical.

Once we have obtained the embeddings, how do we judge whether the text embedding is successful? The most obvious check we can do, in our context, is to see if the embeddings are useful for hedonic price prediction. And find that they are extremely useful, as we report later in the paper. Furthermore, we can also check qualitatively if words that have similar meanings have similar embeddings. Ordinarily, this is done through the correlation or cosine-similarity, more precisely:

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

According to our human notion of similarity, the more similar the words are (controlling for context), the higher the value the formal similarity measure should take, up to a maximum value of 1. Even the first generation of text embedding algorithms was able to achieve remarkable qualitative performance at this task. We give such examples in the Appendix, and the latest generation of the algorithms only improved in their ability to understand language. In what follows, we present a non-technical discussion of BERT, a state-of-art language understanding model; in the Appendix, we present a more technical discussion.

BERT at a high level. BERT (Bidirectional Encoder Representations from Transformers, Devlin et al. (2018); Vaswani et al. (2017)) is a transformer-based model developed by Google. The model is trained using a transfer learning approach, meaning auxiliary prediction tasks—predicting a masked word in a sentence or predicting the order of two sentences—that are not connected to the final tasks, such as predicting hedonic prices in our case. The auxiliary tasks are selected such that the amount of examples is very large, and such that performing the tasks accurately reflects a high level of comprehension. Fine-tuning BERT is done on Amazon's product descriptions for apparel. In our context, we further perform light fine-tuning on the hedonic prediction tasks.

BERT is particularly good at understanding the meaning of words in context, and this property is generally attributed to the transformer blocks present in the neural network (Vaswani et al., 2017). The transformer blocks in BERT allow the model to understand the context of a word by looking at the words that come before and after it, rather than just relying on the individual word. The transformer blocks comprise two main components: a self-attention mechanism and a standard feed-forward neural network. The self-attention mechanism allows the model to weigh the importance of different words in a sentence when making predictions, producing so-called "attention weights." In fact,

BERT uses several self-attention mechanisms in parallel, thus allowing the model to "attend to" different parts of the input simultaneously. The feed-forward neural network, also known as a fully connected layer, takes the output from the self-attention mechanism and applies a series of non-linear transformations. This component allows the model to learn more complex transformations of the input. Both components are applied to the input in parallel, and then the outputs from both are linearly summarized. This summary is then used as the input for the next transformer block.

BERT proceeds to produce the text embeddings by first tokenizing the input text into individual words or sub-words. These tokens are then passed through the BERT model. The transformer blocks process the tokens in parallel and learn to represent each token in a vector space, called an embedding. Indeed, the last hidden layers of the network are the embeddings. During the training phase, BERT learns a set of parameters that can be used to generate embeddings for any input text. The embeddings are then fine-tuned during the fine-tuning phase to perform a specific natural language understanding task; in our case, we fine-tune them on price prediction tasks. The embeddings generated by BERT are contextual embeddings, which means that the embedding for a word depends on the context in which it appears in the input text. This is in contrast to non-contextual embeddings, such as Word2Vec, which assign the same embedding to a word regardless of its context. We present more technical details in Appendix A.

It is helpful to compare BERT to GPT (3 and 4), which has received widespread media attention. GPT also uses transformer blocks, but it does so in an autoregressive fashion. It uses supervised learning, which means it is trained on specific tasks to generate human-like text. GPT is pre-trained on a massive amount of text data. It can be fine-tuned for various natural language generation tasks such as text summarization, text completion, and translation. While GPT is a tool for generating human-like language, BERT is a tool for understanding language. However, since GPT also must understand language well enough to generate conversation, it would be interesting to see if it can provide embeddings that can outperform those generated by (fined-tuned) BERT. This is a direction for future work.

4.2. **Image Embeddings using ResNet-50.** ResNet-50 is a convolutional neural network architecture designed for image classification tasks (He et al., 2016). The '50' in the name

refers to the number of specific structured blocks of neurons in the network. The key innovation in ResNet is the use of residual connections, which allows for a very deep network structure (i.e., a highly flexible model) without suffering from vanishing gradients, a problem that typically limits the ability to learn from data. A residual connection is a shortcut connection that bypasses one or more layers and directly connects the input to the output of a specific layer. This approach enables the model to learn simple residual functions added to the original input, rather than approximating the desired output with many non-linear layers. Consequently, it becomes possible to train much deeper networks while maintaining good performance. ResNet-50 is trained on the ImageNet dataset, which contains over 14 million images and more than 22,000 classes, and it has been trained to classify object images in 1000 classes with high accuracy. At the time of its release, the ResNet50 model achieved the best results in image classification, particularly for the ImageNet and COCO datasets. There are recent advances in computer vision—vision transformers—that import the transformer architecture from large language models such as BERT. They also achieve a near state-of-art performance, but we have not yet explored their use in our context.

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 a meaningful summary, or embedding, of the image. The reason is that the last hidden layer is used for object classification using a simple logistic model, so it should represent the object type accurately. This information is clearly useful for our purposes.

5. AI-Based Hedonic Price Models and Price Indices for Apparel

Having constructed numeric embeddings which capture demand-relevant characteristics of the product as expressed in its image and text description, we may set about estimating the hedonic price function. In this section, we compare the performance various estimation strategies. We then report derived indices for apparel under our preferred method and contrast it with other major indices.

5.1. **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 products, with

tens of millions of products sold in the Amazon marketplace.[18] Our study covers the period from 01/2013 to 12/2018. The transaction prices of a product i 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.

To estimate the hedonic price function, we collected the most recent description (Text_i) and image (Image_i) available for each product as of July 2019—shortly after the end of the study. Descriptions include the product's title, brand name, a list of high-level bullet points, and a description provided by the seller.

These images and descriptions are occasionally updated. We remark that changes to images and descriptions do not reflect changes to the underlying product, which would require creation of a new product identifier (see Section 5.1.1 below). Thus, the captured descriptions and images represent each product throughout the study's duration; changes to the description or image do not a product's underlying valuable attributes, hence they should not change its hedonic price.

One key characteristic of our data is the very high turnover of products from month to month, as shown in Figure 4. Moreover, Figure 5 shows that there is considerable growth in the selection of products. As discussed in the introduction, these properties motivate the use of the hedonic approach for gauging changes in price levels.

This analysis used a dataset of approximately 20 terabytes, owing to the number of products and the length of the time period covered; cloud computing tools based upon Apache's Hadoop and Spark were highly effective for holding and processing the data

[18]The quantity information is proprietary, but we note 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, hedonic prices and hedonic price indices derived using quantity weight can be approximated to various extents by publicly available price information, product information and images, and product rank information.

(both in terms of cost and computing time).[19] In this case, these cloud computing resources were provided by Amazon Web Services (AWS). Computation of the product embeddings and estimation of the hedonic price functions were carried out on a GPU cluster, also provided by AWS.

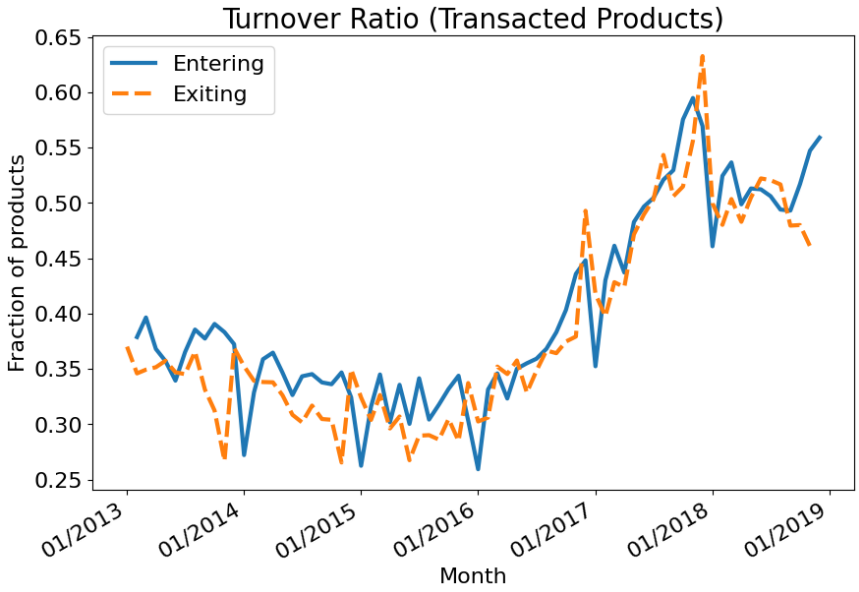

FIGURE 4. Turnover Rate for Products. The Figure shows the share of products with transactions in a given month and no transactions in the previous month (blue line), as well as the share with transactions in a given month and no transactions in the next month (orange line).

5.1.1. *Defining products.* In conducting our study, we have chosen to define "products" at the most granular level. In particular, different sizes, colors, and versions of an article of clothing are treated as distinct products, even if they are share the same brand and item name. This gives the neural network model flexibility to group products and adapt to differences in demand between, say, more- or less-popular colors of a given item, and limits the role of human judgement.[20] This level of granularity is particularly natural in our context, since product variations are typically listed at different prices in

[19]For example, training our model takes under 24 hours on a cloud computer with 8 GPUs, corresponding to less then $500 in cloud computing fees using the Sagemaker environment provided by Amazon Web Services. The largest expense was data storage, which came to roughly $460 per month.

[20]Other choices are of course possible. See Aizcorbe (2014, Sec. 4.1) for further discussion on how researchers may choose to define the product.

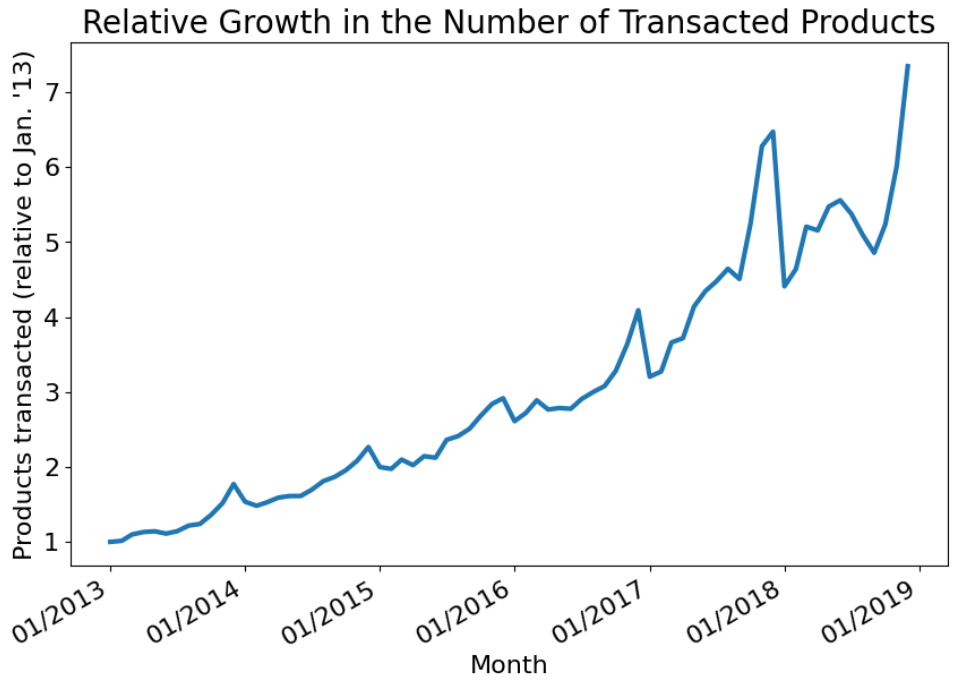

FIGURE 5. Products transacted per month relative to the number of products transacted in January, 2013.

Amazon's marketplace. It does, however, influence the interpretation of Figures 4 and 5 as it affects the measurement of product variety and product turnover in our data. Using an auxiliary catalogue of all product variations, we verify in Appendix D.2 that the high out-of-sample accuracy reported later in this section is not driven by variations of essentially similar products: we observe similar accuracy when restricting to products from the hold-out sample which are not a variation of any product in the training data.

5.2. **Out-of-sample Performance for Predicting Prices.** In Table 2, we first examine the predictive performance, recording the R^2 for predicting prices in the hold-out sample of products. In addition to the multi-task price prediction neural network using text and image embeddings W (from BERT) and I (from ResNet50), which is discussed extensively in Sections 3 and 4, we consider the following alternative methods for estimating the hedonic price function:

- linear regression using features generated by a collection of binary encodings of the product catalog;

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

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

- linear regression using neural-network embedding features W and I generated by BERT and ResNet50 as discussed in Sec. 4;
- gradient-boosted trees using the aforementioned features W and I;[21]
- a "single-task" neural network regression using embeddings W and I, corresponding to an alternative neural network architecture[22]

Results from the comparison are seen in Table 2. When we switch from the basic catalog features to the embeddings W and I, we obtain a first major improvement—even using linear regression in the final step. We obtain a second major improvement when we switch from linear regression to a scalable implementation of tree-based methods (gradient-boosted trees). Finally, we obtain the final major improvement as we switch from a random forest to a multi-task neural network. Thus, the neural network model easily achieves better predictive performance than other methods.

Figure 6 also presents the month-to-month performance of the various models. The out-of-sample R^2 for the best multi-task neural network model ranges between 80% to 90%. Multi-task neural networks (NNs) uniformly dominate single-task NNs, which in turn uniformly dominate boosted tree models and linear models. We also explore the

[21]Gradient-boosted trees (see Chen and Guestrin, 2016) are a relative of random forests; they have been shown to perform well in general prediction tasks and are easier to estimate in very large datasets.

[22]Roughly, single-task learning corresponds to estimating a separate model in each time period. See Appendix C for a formal definition.

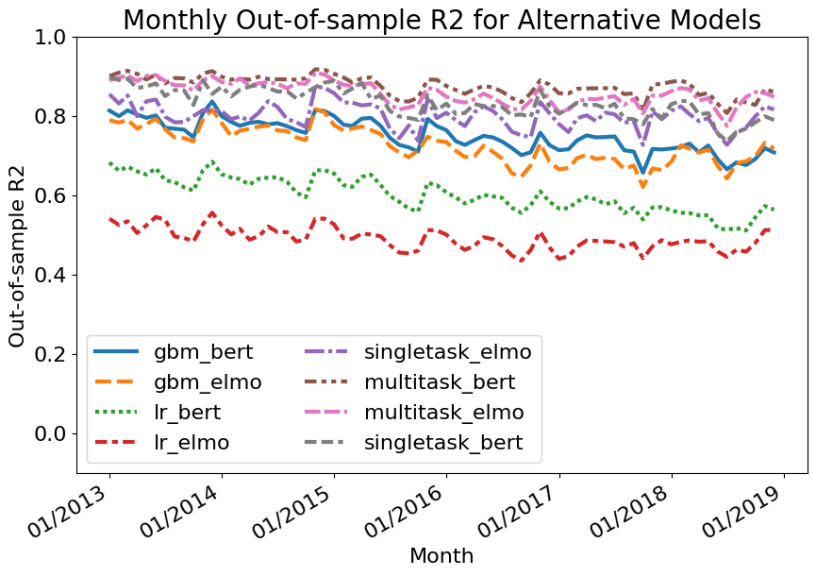

FIGURE 6. The out-of-sample performance of the empirical hedonic price function obtained using our neural network every month since January, 2013, in comparison to alternative models. Multi-task neural networks dominate single-task neural networks (discussed in Appendix C). Neural network models dominate boosted tree models (GBM), which in turn dominate linear models (LM).

performance of BERT-based embeddings of the product description, W, with an alternative embedding of the product description produced using the ELMO network[23] The BERT-based multi-task NN nearly uniformly outperforms ELMO-based multi-task NN, although the performance of the two models is generally similar.

A potential limitation the multi-task model in comparison to single-task models, however, is that it requires retraining the full model as each new year's data becomes available. Moreover, the performance of the best multi-task models is better at the beginning of the studied period and worsens at the end of the period, possibly due to the increasing variety and number of products being transacted (this is shown in Figure 5). It is worth mentioning that the out-of-sample R^2 agrees with validation R^2 we obtained in training (not reported), which suggests that our training approach successfully limits overfitting.

[23]ELMO is a large language model based upon a recursive neural network architecture. Although it does not utilize the transformer architecture, it performs nearly as well as BERT in our setting. We review details of ELMO in Appendix A

5.2.1. *Examples of hits and misses.* It is instructive to inspect the performance of the best NN model using particular examples. Here we take one example where the prediction worked well and another where the prediction worked poorly, see Figures 7 and 8. For the first example, a sweater, the neural network model predicts the price of this item at about 100. The time-averaged average price as shown on camelcamel.com—a third-party website which tracks prices on Amazon.com—is 97, with the price ranging from 39 (corresponding to liquidation events) to 120. For the second example, a designer dress, the neural network 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 large inaccuracy, the price history for this item, again as seen on camelcamel.com, suggests that there were periods when the price ranged between 206 and 2800, with an average sale price of 464, which is not as far off from our prediction. An important feature that is possibly missed by these embeddings is “cruising around custom rose gold buttons," although it is hard to verify this feature's importance objectively.

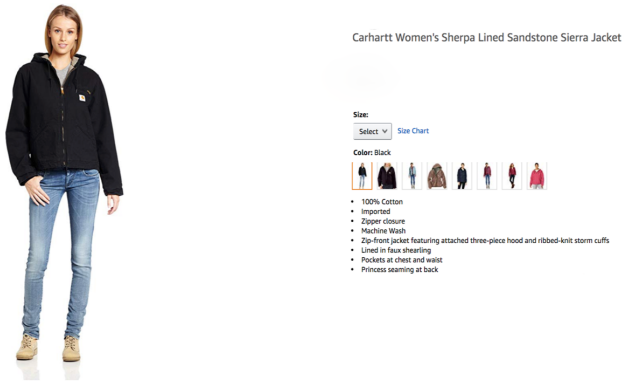

FIGURE 7. An example of accurate price prediction for a women's jacket (black, size medium): The neural network model predicts the price of this item at about $100. The average price on the price aggregator `camelcamel.com` is $97, with the offer price ranging from $39 to $120.

5.3. **Statistical Significance and Inference.** Next we examine the hedonic price model's statistical significance using the methods in Section 3.2.[24] We illustrate this approach in Figure 9, which reports the estimated coefficients on value embeddings for the month

[24]Note that the procedure involves re-estimating the final model weights, θ. Thus, this section considers slightly different coefficient and hedonic price estimates than those discussed elsewhere in the paper.

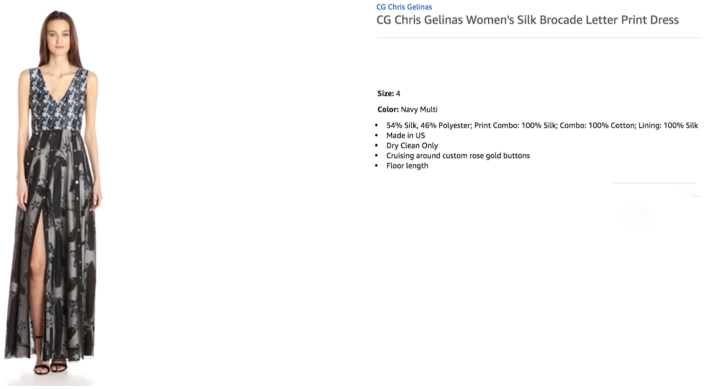

FIGURE 8. An example of inaccurate prediction: The neural network model predicts the price of this item at about 300, but the recent prices were around 2400. While this seems a miss, the price history for this item (whose identifier is `B06XR39DJ1`), as recorded by the Amazon.com price aggregator `camelcamel.com`, shows that the price for this item ranged between $206 and $2800, with an average price of $464.

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.

Product	Month	P_{it}	$\widehat{H}_{it}$	SE_{it}	$[L_{it}, U_{it}]$	$[L_{it}(P_{it}), U_{it}(P_{it})]$
W M Black Jacket (B01DJ34PD2)	Dec. 2018	118.8	114.6	0.05	[114.5, 114.7]	[102, 126]
W XS Blue Jacket (B01MSBDGTL)	Dec. 2018	119.9	110.25	0.06	[110.1, 110.4]	[98 , 122]

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

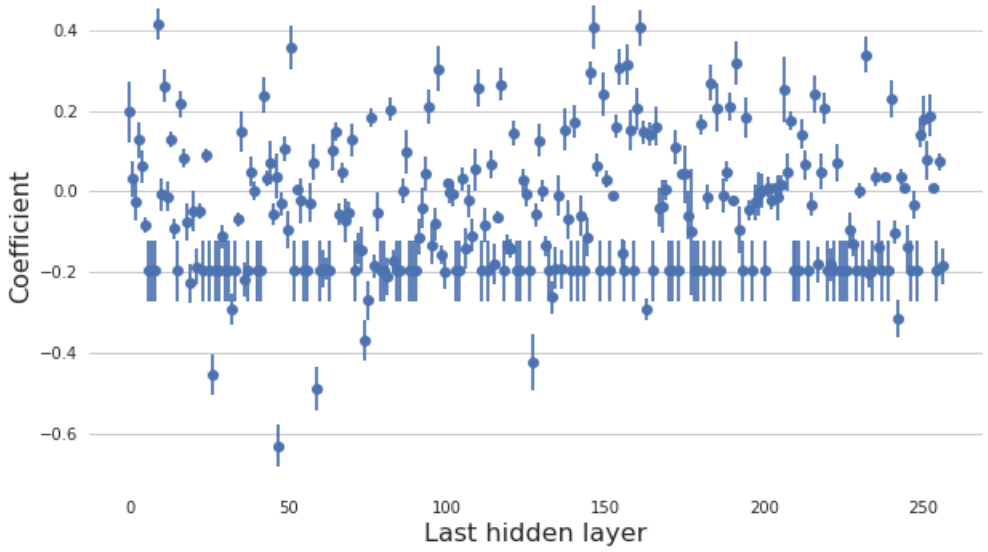

FIGURE 9. Statistical Significance of Value Embeddings via Linear Regression Model Applied to the Test Sample. The Figure shows the point estimates and the pointwise 95% confidence intervals on the coefficients of the value embeddings, as estimated by the linear regression model, applied to the hold-out sample. Note that more than 90% of the coefficients are statistically significant at the 10^{-5} level.

5.4. **Hedonic Price Indices for Apparel.** Here we will use the definitions of hedonic indices introduced in Section 2. For the computation of the hedonic indices we use the best predictive model from the preceding comparison, which we have described in detail in Sections 3 and 4.

In Table 4, we present our main result—the estimates of an average annual rate of inflation in apparel over the period spanning from 2014 to 2019, via Fisher hedooic price indices (yearly-chained, monthly chained, and the geometric mean of the two), the Jevons Posted price index, the Adobe DPI (Adobe Inc., 2024), and the urban CPI for apparel (Bureau of Labor Statistics, 2021):

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

Apparel Indices	*Change in Price Index*
Fisher Hedonic, Yearly Chaining (FY)	-0.98%
Fisher Hedonic, Monthly Chaining (FM)	-5.27%
Fisher Hedonic, Geometric Mean ($\sqrt{FY \cdot FM}$)	-2.28%
Fisher Matched, Monthly Chaining (FI)	-3.12%
Jevons Posted Price Index, Daily Chained (JPI)	-3.01%
Adobe Digital Price Index, Monthly Chained (DPI)	-2.02%
U.S. Urban Apparel (BLS)	-0.31%

TABLE 4. Estimates of Average Annual Rate of Inflation in Apparel over five years, 2014-2019: Fisher Hedonic Index, Fisher Matched Index, Jevons Posted Price Index, Adobe DPI, and the BLS urban CPI.

E.1 The main index – the yearly-chained FHPI – suggests an average price decline in 2014-2019 of .98%. In contrast, the BLS CPI for apparel suggests that there was a somewhat smaller decline in the price level.

E.2 The yearly chained FHPI declines at a slower rate than the matched Fisher index, Adobe DPI, and the monthly-chained FHPI, highlighting the importance of using hedonic adjustment and long chaining.

Indeed, the matched index potentially contains a matching bias created by the high turnover of products illustrated in Figure 4. Moreover, as we emphasized in the introduction, the use of hedonic prices allows us to perform “long" chaining, thereby mitigating the chain drift problem.

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, BLS also uses their own hedonic models to perform quality adjustments.[25] Moulton (2018) discusses methodological points and estimates that the upward bias in the overall (non-apparel specific)

[25]Note that our goal here is also not to replicate the BLS CPI index but to construct a modern hedonic version of a classical Fisher index. We also note that the replication is simply precluded in our setting because we can not assign extremely many products manually to the subcategories used by BLS.

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 a linear model instead of the neural network (not reported), and conclusions seem qualitatively robust with respect to whether the linear model or nonlinear neural network model is used. However, the neural network models, which do exhibit superior predictive ability, result 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 on the neural network 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 neural network models are more noisily estimated than the linear models, but they are much less biased, giving a better predictive performance as a result. Still, after taking weighted averages of prices, there are only moderate differences between the indices based on the neural network and those based on linear models.

In Figure 10, we demonstrate the dynamics of the year-chained FHPI and compare it with the Jevons index and the monthly chained FHPI.

The Jevons index is a geometric mean of the posted price relatives and does not incorporate quantity weighting. It underlies the Billion Price Project (Cavallo, 2018) and is also employed by statistical agencies as a complementary index (Office of National Statistics, 2020). The index is convenient and easy to construct because it only relies on publicly posted offer prices and does not use quantity information. The index measures the average "within" price level change calculated on a much larger universe of products (with and without transactions). In contrast, the Fisher index reflects both the "within" price change and the "between" price change since it also reflects the substitution arising from utility-maximizing behavior by customers with budget constraints. The key empirical observations are as follows.

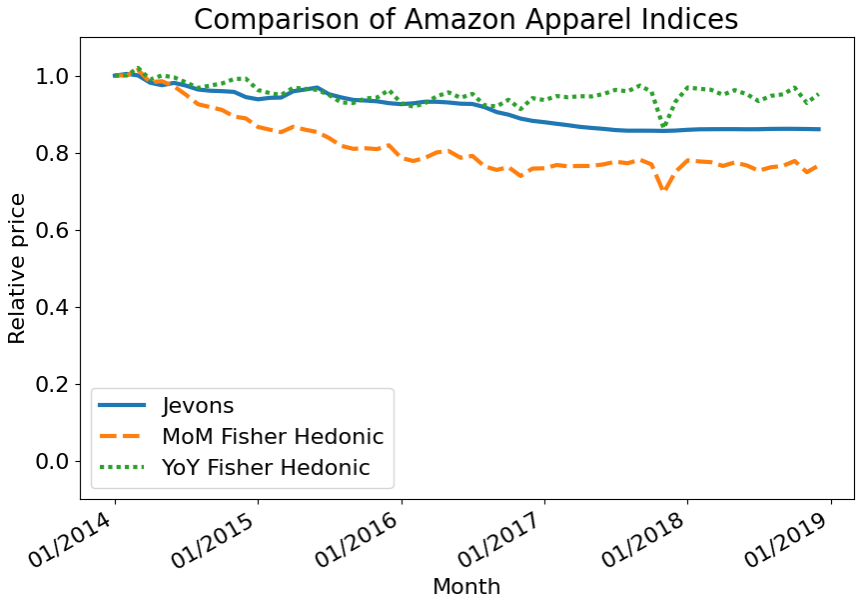

FIGURE 10. Dynamics of the yearly-chained Fisher Hedonic Price Index (dotted line), the Jevons Price Index (solid lined), and the monthly-chained Fisher Hedonic Price Index (dashed line).

E.3 The Jevons index exhibits steady declines over the studied period, with a total decline of more than 10%; the rate of decline is considerably larger and much more implausible than that of the yearly chained FHPI.

This may suggest that either the chain drift or the substitution effects are sufficiently large to matter. Without the ability to do long chaining and (or) without some proxy for quantity information (such as sales rank data), it is doubtful that Jevons-type indices can accurately approximate inflation. Still, the Jevons index continues to be a very important, simple way to gauge large changes in inflation, as the BPP (based on the Jevons index) has amply demonstrated (Cavallo and Rigobon (2016)). [26]

[26]Note that given a reference series, such as the CPI, one can always use a post-processing or "now-casting" approach to modify Jevons or any other digital index R_g to track the reference index R_f as well as possible. For example, one can construct the best linear predictor $\hat{a}+\hat{b}\log R_g$ of $\log R_f$ using the historical data on (R_g, R_f) in a given time frame and then use $R_g^\star = \exp(\hat{a}+\hat{b}\log R_g)$ as the post-processed index. This further highlights the potential usefulness of the Jevons and other digital indices, even though their raw versions exhibit large biases. In our discussions, we focus on raw indices without post-processing, with the understanding that post-processing can always be employed as needed.

Finally, when we compare yearly-chained FHPI to monthly-chained FHPI, the key observation is as follows:

E.4 The monthly-chained FHPI exhibits a strong, very implausible 20% decline in the price level, in sharp contrast to the yearly-chained FHPI.

This observation highlights the importance of mitigating the “chain drift" problem, which appears to be quite severe for the frequently compounded series such as the monthly chained FHPI. For this reason the monthly chained FHPI and other frequently chained indices (matched Fisher and daily-chained Jevons) are highly unlikely to approximate inflation accurately, especially over longer horizons. Of course, as we stressed before for Jevons, these indices can be useful over short horizons to gauge large changes in the price level.[27]

6. CONCLUSION

We develop empirical AI-based models of hedonic prices and derive hedonic price indices for measuring changes in consumer welfare. To achieve this, we generate product attributes (or 'features') from text descriptions and images using deep neural networks. We then use these features to estimate the hedonic price function, again using deep learning. More precisely, we convert text information about the product to numeric product features using the BERT large language model, fine-tuned on Amazon's product descriptions, and convert the product image to numerical product features by a pre-trained ResNet50 neural network model. We use a multi-task neural network to produce estimated hedonic prices over many periods using the aforementioned features. All the ingredients to the method rely on publicly available, open-source software components. We apply the models to Amazon's proprietary data on first-party sales in apparel. Resulting models have high predictive accuracy for several product categories, with the R^2 ranging from 80% to 90%. Our main hedonic index estimates the rate of inflation in apparel to be moderately negative, somewhat lower than the rate estimated by the CPI.

We find the performance of AI-based hedonic models remarkable for two reasons. First, the high predictive accuracy of our models suggests that the theoretical hedonic price models from economics provide a good approximation of real-world equilibrium

[27] They can also be employed in a nowcasting fashion, as explained in the previous footnote.

prices. Second, our methods are AI-powered algorithms that are scalable to many products and avoid the significant manual effort required to construct more traditional hedonic price indices. The good performance of these methods suggests that AI-powered methods, particularly our embeddings, can be used to build real-time hedonic price indices using electronic data and facilitate price research in many settings. These methods can also power economic research in many other areas—for example, labor economics, where embeddings can be used to characterize workers' job-relevant characteristics, or industrial organization, where embeddings can be used to characterize firms and their products. Since all main ingredients of our approach are open-source, publicly-available technology, future scientific pursuits in this direction can readily use these tools or other similar open-source products.

Based on our experience, we believe there are natural directions for further research. First, we found that while images used alone help predict prices, they add very little to prediction accuracy once the text embeddings are included in the model. Therefore, finding better ways or models to leverage image data is an important unresolved problem. Second, an important further research direction is model explainability. When the price of the product changes, we'd like to explain the price change in terms of changing valuations of key attributes. Similarly, when comparing the prices of two products, we would like to attribute the price difference to the valuations of key attributes. Given the non-linearity of the AI-based hedonic models, explainability is not a simple problem. We hope interested researchers take notice of these open problems.

7. Acknowledgements

We thank the participants at the 2018 Federal Economic Statistics Advisory Committee meeting, the 2019 Allied Social Science meetings, the 2019 Federal Reserve Board conference on "Nontraditional Data, Machine Learning, and Natural Language Processing in Macroeconomics", the 2019 Brookings Conference "Can big data improve economic measurement?" and the 2021 CEMMAP conference "Measuring prices and welfare," and seminars at Berkeley, European Bank of Reconstruction and Development, MIT, UCL, and York. We are grateful to Andrew Chesher, Greg Duncan, Kevin Fox, John Haltinwager, James Heckman, Daniel Miller, Mathew Shapiro, Bernhard Schölkopf, James Stock, and Weining Wang for helpful comments during the various stages of this project.

All authors were fully or partially employed by Amazon.com while conducting the research.

REFERENCES

Adobe Inc. Adobe digital price index. https://business.adobe.com/resources/digital-price-index.html, 2024. [Accessed 16-10-2024].

Ana Aizcorbe. *A practical guide to price index and hedonic techniques*. Oxford University Press, USA, 2014.

Patrick Bajari and C Lanier Benkard. Demand estimation with heterogeneous consumers and unobserved product characteristics: A hedonic approach. *Journal of political economy*, 113(6):1239–1276, 2005. Publisher: The University of Chicago Press.

Patrick Bajari, Zhihao Cen, Victor Chernozhukov, Manoj Manukonda, Jin Wang, Ramon Huerta, Junbo Li, Ling Leng, George Monokroussos, and Suhas Vijaykunar. Hedonic prices and quality adjusted price indices powered by ai. *CEMMAP, London*, 2019.

C. Lanier Benkard and Patrick Bajari. Hedonic price indexes with unobserved product characteristics, and application to personal computers. *Journal of Business & Economic Statistics*, 23(1):61–75, 2005. Publisher: Taylor & Francis.

Steven Berry, James Levinsohn, and Ariel Pakes. Automobile prices in market equilibrium. *Econometrica: Journal of the Econometric Society*, pages 841–890, 1995. Publisher: JSTOR.

Steven Berry, 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):68–105, 2004. Publisher: The University of Chicago Press.

Charles Brown. Equalizing differences in the labor market. *The Quarterly Journal of Economics*, 94(1):113–134, 1980.

U.S. Bureau of Labor Statistics. CPI for all urban consumers, 2014-2021 [data set]. http://data.bls.gov, 2021. [Accessed 16-10-2024].

Michael Cafarella, Gabriel Ehrlich, Tian Gao, John C Haltiwanger, Matthew D Shapiro, and Laura Zhao. Using machine learning to construct hedonic price indices. Technical report, National Bureau of Economic Research, 2023.

Alberto Cavallo. Online and official price indexes: Measuring Argentina's inflation. *Journal of Monetary Economics*, 60(2):152–165, 2013.

Alberto Cavallo. Are Online and Offline Prices Similar? Evidence from Large Multichannel Retailers. *American Economic Review*, 107(1):283–303, January 2017. ISSN 0002-8282. doi: 10.1257/aer.20160542. URL https://www.aeaweb.org/articles?id=10.1257/aer.20160542.

Alberto Cavallo. Scraped Data and Sticky Prices. *The Review of Economics and Statistics*, 100(1):105–119, March 2018. ISSN 0034-6535. URL https://doi.org/10.1162/REST_a_00652.

Alberto Cavallo and Roberto Rigobon. The distribution of the size of price changes. Technical report, National Bureau of Economic Research, 2011.

Alberto Cavallo and Roberto Rigobon. The billion prices project: Using online prices for measurement and research. *Journal of Economic Perspectives*, 30(2):151–78, 2016.

Kenneth Y Chay and Michael Greenstone. Does air quality matter? evidence from the housing market. *Journal of political Economy*, 113(2):376–424, 2005.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Xiaohong Chen and Halbert White. Improved rates and asymptotic normality for nonparametric neural network estimators. *IEEE Transactions on Information Theory*, 45 (2):682–691, 1999. Publisher: IEEE.

Victor Chernozhukov, Mert Demirer, Esther Duflo, and Ivan Fernandez-Val. Generic machine learning inference on heterogeneous treatment effects in randomized experiments, with an application to immunization in india. Technical report, National Bureau of Economic Research, 2018.

Victor Chernozhukov, Alfred Galichon, Marc Henry, and Brendan Pass. Identification of hedonic equilibrium and nonseparable simultaneous equations. *Journal of Political Economy*, 2020.

Antonio G Chessa and Robert Griffioen. Comparing price indices of clothing and footwear for scanner data and web scraped data. *Economie et Statistique*, 509(1):49–68, 2019. Publisher: Persée-Portail des revues scientifiques en SHS.

Pierre-André Chiappori, Robert J McCann, and Lars P Nesheim. Hedonic price equilibria, stable matching, and optimal transport: equivalence, topology, and uniqueness. *Economic Theory*, 42(2):317–354, 2010. Publisher: Springer.

AT Court. Hedonic price indexes with automotive examples. In *The dynamics of automobile demand*, pages 98–119. General Motors Corporation New York, 1939.

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

W Erwin Diewert. Exact and superlative index numbers. *Journal of econometrics*, 4(2): 115–145, 1976. Publisher: Elsevier.

W Erwin Diewert. Index number issues in the consumer price index. *Journal of Economic Perspectives*, 12(1):47–58, 1998.

W Erwin Diewert and Kevin J Fox. Substitution bias in multilateral methods for cpi construction. *Journal of Business & Economic Statistics*, 40(1):355–369, 2022.

Gabriel Ehrlich, John Haltiwanger, Ron Jarmin, David Johnson, and Matthew D Shapiro. Minding your ps and qs: Going from micro to macro in measuring prices and quantities. In *AEA Papers and Proceedings*, volume 109, pages 438–443. American Economic Association 2014 Broadway, Suite 305, Nashville, TN 37203, 2019.

Gabriel Ehrlich, John C Haltiwanger, Ron S Jarmin, David Johnson, Ed Olivares, Luke W Pardue, Matthew D Shapiro, Laura Zhao, et al. Quality adjustment at scale: Hedonic vs. exact demand-based price indices. Technical report, National Bureau of Economic Research, 2023.

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

Austan D Goolsbee and Peter J Klenow. Internet rising, prices falling: Measuring inflation in a world of e-commerce. In *Aea papers and proceedings*, volume 108, pages 488–92, 2018.

William M Gorman. A possible procedure for analysing quality differentials in the egg market. *The Review of Economic Studies*, 47(5):843–856, 1980.

Yuriy Gorodnichenko and Oleksandr Talavera. Price Setting in Online Markets: Basic Facts, International Comparisons, and Cross-Border Integration. *American Economic Review*, 107(1):249–282, January 2017. ISSN 0002-8282. doi: 10.1257/aer.20141127. URL `https://www.aeaweb.org/articles?id=10.1257/aer.20141127`.

Yuriy Gorodnichenko, Viacheslav Sheremirov, and Oleksandr Talavera. Price Setting in Online Markets: Does IT Click? *Journal of the European Economic Association*, 16(6): 1764–1811, December 2018. ISSN 1542-4766. doi: 10.1093/jeea/jvx050. URL `https://doi.org/10.1093/jeea/jvx050`.

Michael Greenstone. The continuing impact of sherwin rosen's "hedonic prices and implicit markets: product differentiation in pure competition". *Journal of Political Economy*, 125(6):1891–1902, 2017.

Z Griliches. Hedonic price indexes for automobiles: an econometric of quality change. Price Statistics Review Committee, 1961.

Robert M. Groves and Brian A. Harris-Kojetin, editors. *Innovations in Federal Statistics: Combining Data Sources While Protecting Privacy*. The National Academies Press, Washington, DC, 2017. ISBN 978-0-309-45428-5. doi:

10.17226/24652. URL https://nap.nationalacademies.org/catalog/24652/innovations-in-federal-statistics-combining-data-sources-while-protecting-privacy.

Sukjin Han, Eric H Schulman, Kristen Grauman, and Santhosh Ramakrishnan. Shapes as product differentiation: Neural network embedding in the analysis of markets for fonts. *arXiv preprint arXiv:2107.02739*, 2021.

Jessie Handbury, Tsutomu Watanabe, and David E Weinstein. How much do official price indexes tell us about inflation? Technical report, National Bureau of Economic Research, 2013.

Kaiming He, 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*, pages 770–778, 2016.

ILO, World Bank, et al. *Consumer price index manual: Theory and practice*. International Labour Organization, 2004.

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

Ron S Jarmin. Evolving measurement for an evolving economy: thoughts on 21st century us economic statistics. *Journal of Economic Perspectives*, 33(1):165–184, 2019.

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

Patrick Kidger and Terry Lyons. Universal approximation with deep narrow networks. In *Conference on learning theory*, pages 2306–2327, 2020. tex.organization: PMLR.

DP Kingma. Adam: a method for stochastic optimization. In *Int Conf Learn Represent*, 2014.

Kelvin J Lancaster. A new approach to consumer theory. *Journal of political economy*, 74(2):132–157, 1966.

David Lebow. *Modernizing the Consumer Price Index for the 21st Century: National Academies of Sciences, Engineering, and Medicine: 2022*. Springer, 2023.

Andrew Leicester. The potential use of in-home scanner technology for budget surveys. In *Improving the measurement of consumer expenditures*, pages 441–491. University of Chicago Press, 2014.

Daniel McFadden. The measurement of urban travel demand. *Journal of public economics*, 3(4):303–328, 1974. Publisher: Elsevier.

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

James Milton and Jeanine Treffers-Daller. Vocabulary size revisited: the link between vocabulary size and academic achievement. *Applied Linguistics Review*, 4(1):151–172, 2013.

Brent R Moulton. The measurement of output, prices, and productivity. *Report, Productivity Measurement Initiative under The Hutchins Center on Fiscal and Monetary Policy, Brookings Institution, Washington, DC*, 2018.

Sendhil Mullainathan and Jann Spiess. Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106, 2017.

Lars Nesheim. Hedonic price functions. *CEMMAP, London*, 2006.

Andrew Ng. Nuts and bolts of building ai applications using deep learning. *NIPS Keynote Talk*, 2016.

U.K. Office of National Statistics. Using statistical distributions to estimate weights for web-scraped price quotes in consumer price statistics. Technical report, U.K. Office of National Statistics, September 2020.

Ariel Pakes. A reconsideration of hedonic price indexes with an application to PC's. *American Economic Review*, 93(5):1578–1596, 2003.

Sherwin Rosen. Hedonic prices and implicit markets: product differentiation in pure competition. *Journal of political economy*, 82(1):34–55, 1974.

Statistics Bureau of Japan. Explanation of the consumer price index, Jan 2022. URL `https://www.stat.go.jp/english/data/cpi/1590.html`.

James H Stock. Nonparametric policy analysis: an application to estimating hazardous waste cleanup benefits. *Nonparametric and Semiparametric Methods in Econometrics and Statistics*, pages 77–98, 1991.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

David Wasshausen and Brent Moulton. The role of hedonic methods in measuring real gdp in the united states. *BEA Papers*, 67, 2006.

Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017. Publisher: Elsevier.

Shipei Zeng. Hedonic imputation with tree-based decision approaches. *Working Paper, School of Economics, University of New South Wales, December*, 2020.

APPENDIX A. OVERVIEW OF TEXT EMBEDDING MODELS

Here we provide a more technical overview of the text embedding models used in this paper.

A.1. **First generation: Word2Vec Embeddings.** We first recall some basic ideas underlying the Word2Vec algorithm (Mikolov et al., 2013). Here we rely on notation in Section 4. The goal is to find the $r \times d$ matrix $\omega = \{u_j\}_{j=1}^{d}$, representing d words in r dimensional space. The columns are the embeddings for the words.

We can think of a word appearing in sentence as random variable T; and we can let U denote the corresponding embedding. Word2Vec trains the word embeddings by predicting 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:

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

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:

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

where the summation is over many examples $\mathscr{S}$ of subsequences s. r

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) → Fashion, (Fashion,Multi-pocket)→ Women's, (Women's,Cotton) → Multi-pocket, and so on.

We can examine the quality of word embedding by assessing predictive performance for price prediction tasks. We can also qualitatively inspect whether embeddings capture word similarity. For example, we found that the embeddings for "necktie" and "bowtie" are most cosine-similar to the word "tie." The embeddings also seem to induce an interesting vector space on the set of words, which seems to encode analogies well. For example, the embedding for the word "briefcase" is most cosine-similar to the artificial latent word

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

Examples like this and others reported in Mikolov et al. (2013) supported the idea of summing the embeddings for words in a sentence to produce an embedding for sentences.

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 reverse guessing as well:

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

where θ is a parameter vector. Recursive neural networks with 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, 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

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

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 11.

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 the 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."

A.1.1. *Training.* In Figure 11, 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:

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

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

A.1.2. *Producing embeddings.* To produce embeddings from the trained network, each word t_k in a sentence $m = (t_1,\ldots,t_k)$ 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

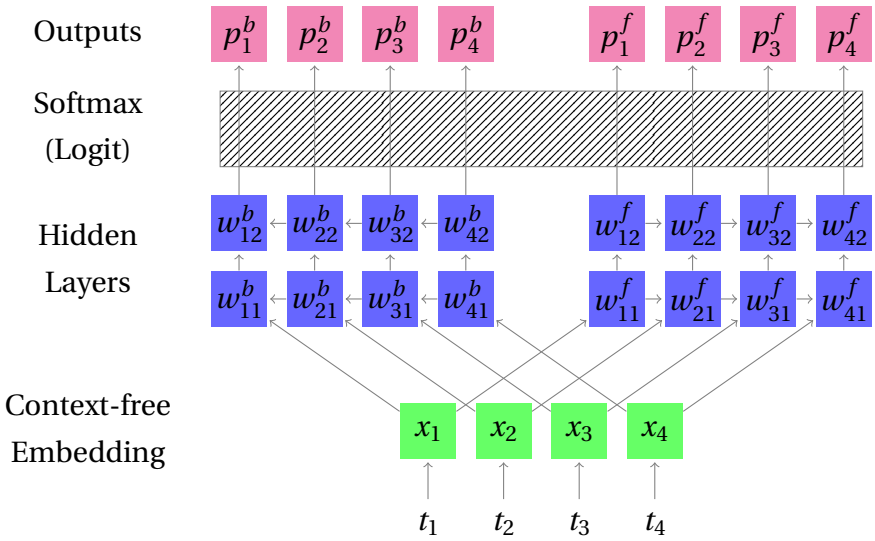

FIGURE 11. ELMO Architecture. This is an ELMO network for a string of 4 words, with $L = 2$ hidden layers. Here, the softmax layer (multinomial logit) is a single function mapping each input in $\mathbb{R}^d$ to a probability distribution over the dictionary Σ.

network performing the final task. In principle, however, the whole network could in principle be plugged in to the network performing the final task and allowed to update.

A.2. **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.

A.2.1. *Computing The Attention.* We begin with n word 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 Attention mapping is applied on X directly:

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

where

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

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 a weighted average of embeddings for 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.

A.2.2. *Generating product embeddings.* Depending on specific tasks and resources, Devlin et al. (2018) suggested using 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,

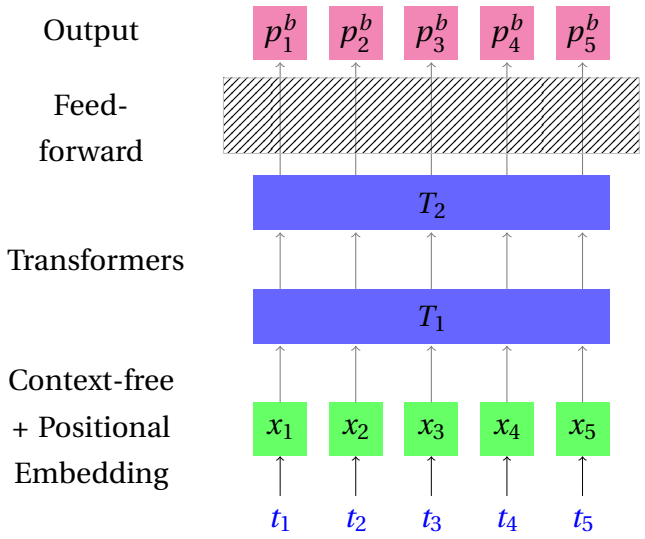

FIGURE 12. BERT Architecture

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.

A.3. **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 toward 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 architecture. 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.

APPENDIX B. OVERVIEW OF IMAGE EMBEDDINGS VIA RESNET50

The central idea of the ResNet 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 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 sub-maps entering the composition of the entire network). Putting together many blocks like these sequentially, we obtain the overall architecture depicted in Figure 13.

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 errors. 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.

APPENDIX C. DETAILS OF DIFFERENT ARCHITECTURES FOR PREDICTION USING EMBEDDINGS

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

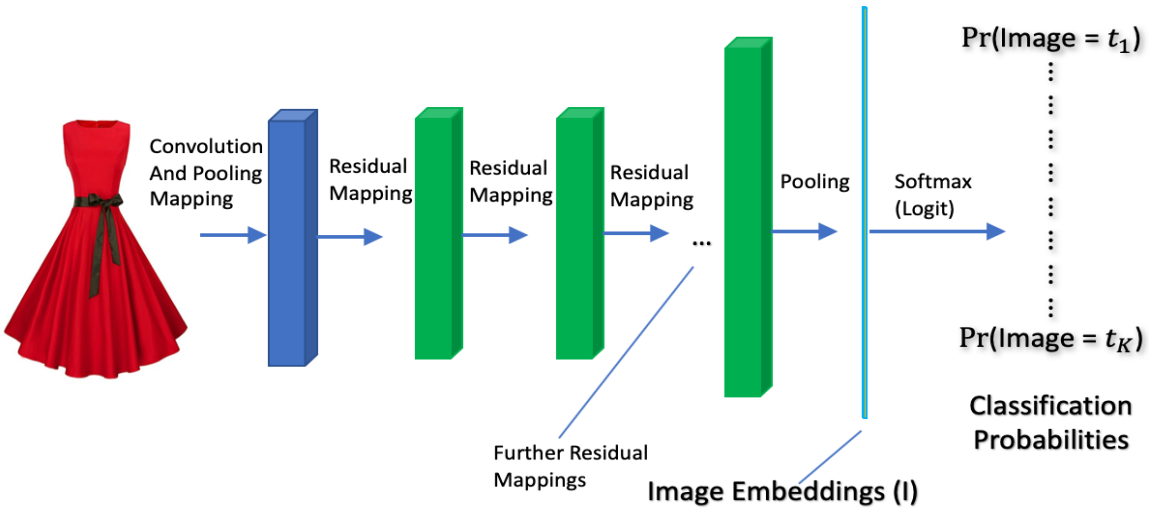

FIGURE 13. The ResNet50 operates on numerical 3-dimensional arrays representing images. It first does some early processing by applying convolutional and pooling filters, then it applies many residual block mappings, producing arrays shown in green. The penultimate layer produces a high-dimensional vector I, the image embedding, which is then used to predict the image type.

C.1. **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 pre-trained ELMO embedding. For images, we used pre-trained ResNet 50 embedding. The structure is shown in Figure 14. 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.

C.2. **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 the title, brand, description, and bullet points (of dimension $768 \cdot 4 = 3072$).

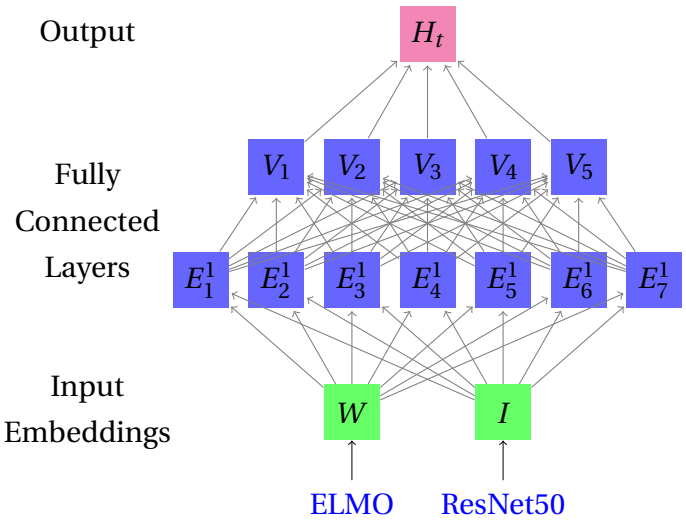

FIGURE 14. SingleTask + ELMO model. Product text is mapped to W and the image is mapped to I of dimensions 256 and 2048. For illustration purposes, we only show two hidden layers with dimensions 7 and 5 respectively. In practice, we use three layers with dimensions 2048, 1024, and 256. The output is price for one time period t.

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.

C.3. **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 $d_T = 768$.

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 16 is a simple illustration of the end-to-end BERT + Multitask model structure.

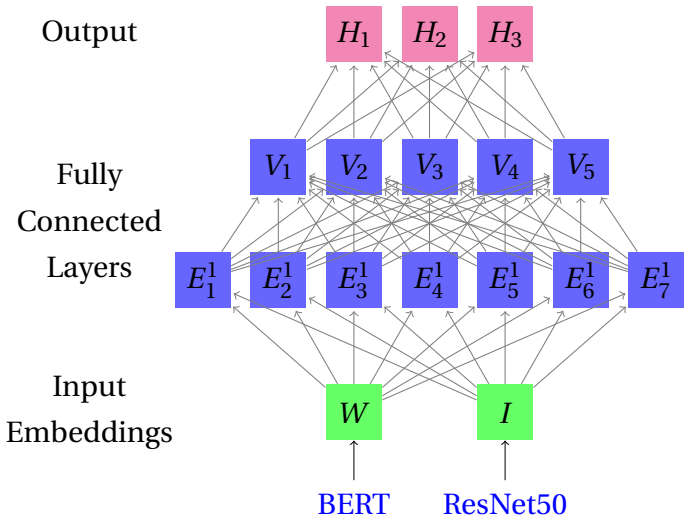

FIGURE 15. MultiTask + BERT. Product text is mapped to W and image is mapped to I of dimensions 3072 and 2048 respectively. For illustration purpose, we only show two hidden layers with dimension 7 and 5 respectively, and output is of dimension 3. In practice, we use three layers with dimension 2048, 1024, and 256, and the output is a price vector over $T = 72$ time periods.

We are experimenting with different numbers of fine-tuned layers. Some initial results show that fine-tuning more layers improve 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.

APPENDIX D. ADDITIONAL DETAILS

D.1. **Identification of average marginal willingness to pay via the hedonic price function.** We briefly sketch identification of the average derivative of a structural function under suitable assumptions. For illustration, we assume with some loss of generality that unobservable product characteristics U are independent of observable characteristics X at equilibrium.

This argument supports our characterization (made in Section 2) that the average partial derivative of the hedonic price function (with respect to the equilibrium distribution of observed and unobserved characteristics) is equal to the average marginal willingness to pay for a particular characteristic.

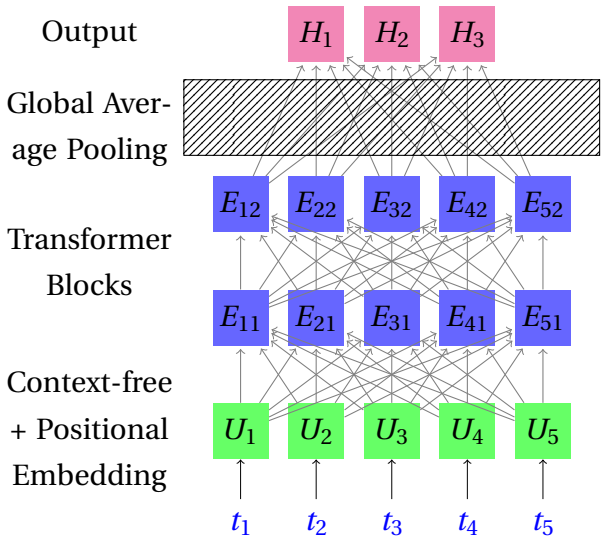

FIGURE 16. MultiTask + Fine-tuned BERT. Product text (sentence) is tokenized and padded to X, where components represent the context-free input embedding plus a positional encoding for a token (word). Then the input is fed into a BERT model which consists of 12 layers of transformer blocks and outputs T dimensional price vector. For illustration purposes, we only show 5 tokens and two transformer blocks.

In the hedonic equilbrium (see e.g. Ekeland et al., 2004), quantities $q_t(x,u)$ and prices $h_t(x,u)$ are well-defined as functions of observed characteristics $x \in \mathscr{X}$ and unobserved characteristics $u \in \mathscr{U}$, for $\mathscr{X}, \mathscr{U} \subset \mathbb{R}^d$. Moreover, at equilibrium, the distribution of characteristics $(X,U) \sim F_t$ is fixed. We denote by F_t^X and F_t^U the associated marginal distributions.

In order to make the sketch, we assume independence of U and X under the distribution F_t. We further assume that the function h_t is bounded and that the functions $\{\nabla_x h_t(x,u)\}_{u \in \mathscr{U}}$ are equicontinuous.

By independence of X and U under F_t, the average structural function

$$h_t(x) = \int h_t(x,u) dF_t^U(u)$$

coincides with the hedonic regression function, in the sense that $h_t(X) = \mathbb{E}_{F_t}[h_t(X,U)|X]$ almost surely. To prove this claim, let ψ denote any bounded, X-measurable test function, and note that by independence the joint distribution induced by F_t coincides with

that of the product measure $F_t^X \otimes F_t^U$. Thus,

$$\begin{aligned}\mathbb{E}_{F_t}[\psi(X)h_t(X,U)] &= \int h_t(x,u)\psi(x)\,dF_t(x,u)\\ &= \int h_t(x,u)\psi(x)\,d\{F_t^X \otimes F_t^U\}(x,u).\end{aligned}$$

By Fubini's theorem, using boundedness of ψ and h_t, this is the same as

$$\begin{aligned}&= \int \left[\int h_t(x,u)\,dF_t^U(u)\right]\psi(x)\,dF_t^X(x)\\ &= \int h_t(x)\psi(x)\,dF_t^X(u) = \mathbb{E}_{F_t}[h_t(X)\psi(X)].\end{aligned}$$

By the definition of the conditional expectation, we conclude that

$$h_t(X) = \mathbb{E}_{F_t}[h_t(X,U)|X].$$

Under standard conditions (say, if the functions $\{\nabla_x h_t(x,u)\}_{u\in\mathscr{U}}$ are equicontinuous), we may then differentiate under the integral to obtain

$$\frac{\partial}{\partial x_k}h_t(x) = \frac{\partial}{\partial x_k}\int h_t(x,u)\,dF_t^U(u) = \int \frac{\partial}{\partial x_k}h_t(x,u)\,dF_t^U(u).$$

Thus, after integrating both sides with respect to dF_t^X, we find that the average derivative of the hedonic price function (left-hand side) coincides with the average marginal willingness to pay for characteristics x_k (right-hand side).

D.2. **Impact of product variations.** In Amazon's data, products are distinguished by their so-called "Amazon standard identification number," or ASIN. These numbers are assigned at the finest-possible granularity; for example, multiple editions of the same book are assigned different ASINs, as are different colors or sizes of the same jacket. Such "child ASINs" are then grouped together into "parent ASINs" which include every size and color of a given jacket, or every edition of a particular book.

As we discussed in Section 5.1.1 in the main text, we define a distinct product for each "child ASIN." This seems sensible for two reasons. For one, in the apparel sector, different sizes and colors can often sell for different prices. For another, this choice gives the neural network freedom to capture important price differences between similar products when data are sufficiently rich, while preserving the possibility of lumping together similar products when fewer transactions are available. Such trade-offs will naturally occur due to regularization of the network's weights.

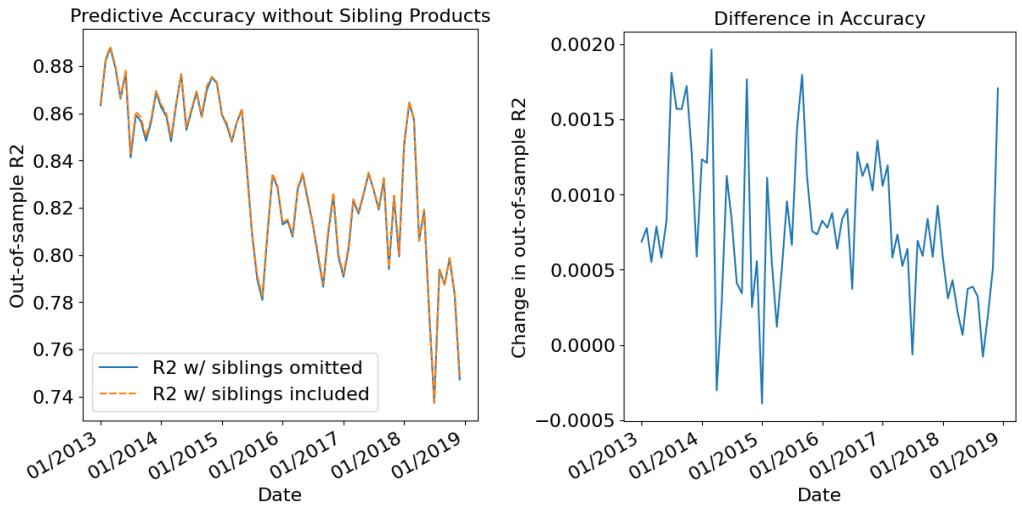

FIGURE 17. Predictive accuracy with and without sibling products. On the left, we have plotted the R^2 obtained both with (orange, dashed) and without (blue, solid) including products that have a sibling in the training dataset (e.g., a variation in size or color, or a newer version). The lines differ imperceptibly—the difference between the two is plotted on the right hand side; it hovers around 0.001.

In principle, the low out-of-sample error we report could be driven by good performance for such "sibling" products that happen to be split across training and test datasets. We first remark that good performance in these cases is still helpful, since product entry and exit often occurs for some colors/sizes and not others. Fortunately, Amazon tabulates "sibling" products (e.g. variations in size and color, or newer versions of existing products). We found that of the roughly 2.6 million products in the test dataset, only 2.93% (roughly 60 thousand) of those products had a relative in the training dataset, limiting the potential impact of these examples on the reported accuracy.

As a final check, we plotted the difference in R^2 within each period when all products in the test dataset which have a relative in the training dataset are removed from consideration. We found that the R^2 was reduced by a negligible amount, on the order of 0.001. This is detailed in Figure 17.