$\mathcal{D}$-Extract: Extracting Dimensional Attributes From Product Images

Pushpendu Ghosh
Amazon
gpushpen@amazon.com

Nancy Wang
Amazon
wangzxi@amazon.com

Promod Yenigalla
Amazon
promy@amazon.com

Abstract

Product dimension is a crucial piece of information enabling customers make better buying decisions. E-commerce websites extract dimension attributes to enable customers filter the search results according to their requirements. The existing methods extract dimension attributes from textual data like title and product description. However, this textual information often exists in an ambiguous, disorganised structure. In comparison, images can be used to extract reliable and consistent dimensional information. With this motivation, we hereby propose two novel architecture to extract dimensional information from product images. The first namely Single-Box Classification Network is designed to classify each text token in the image, one at a time, whereas the second architecture namely Multi-Box Classification Network uses a transformer network to classify all the detected text tokens simultaneously. To attain better performance, the proposed architectures are also fused with statistical inferences derived from the product category which further increased the F1-score of the Single-Box Classification Network by $\approx 3.78\%$ *and Multi-Box Classification Network by* $\approx 0.9\%$*. We use distance supervision technique to create a large scale automated dataset for pretraining purpose and notice considerable improvement when the models were pretrained on the large data before finetuning. The proposed model achieves a desirable precision of* 91.54% *at* 89.75% *recall and outperforms the other state of the art approaches by* $\approx 4.76\%$ *in F1-score*[1].

1. Introduction

Product detail page in an e-commerce website consists of information in the form of title, product description, reviews and product images. These websites continuously innovate and improvise scalable methods to use the seller uploaded details and extract structured information like brand, color, dimension etc., hereafter called product attributes. These product attributes are used as signals in search ranking, filters for refining the search results, comparison of products, finding product substitutes etc. Poor quality product attributes like missing or inconsistent values can cause confusion for customers, leading to lost sales and increased returns for these websites.

Figure 1: Example of a product image with dimensional attributes. According to the established definition (Section 2), the table's length, width and height are 23.5 inch, 68 inch and 36.5 inch respectively.

Particularly for home and furniture products, dimension information is one of the most decisive attribute for the customers. Individually going through each product to find that one product that meet their requirement becomes extremely cumbersome and time consuming. Searching products based on dimension can be made possible only through high coverage of accurate dimension attributes in the catalog. Also these extracted dimensions can be highlighted in the product detail page to enhance information visibility. Thereby, instilling assurance, satisfaction and convincing the customer to buy the perfect sized products for their spaces. Increased discoverability of the appropriate sized product along with a well informed buying decision thereby result in an enhanced shopping experience.

The usual input source of extracting dimensional attributes is unstructured text whose reliability is questionable as this textual information added by sellers is subject to individuality and hence often not standardized and definitive. For instance, some sellers follow the order: width $\times$ depth/length $\times$ height; whereas some sellers men-

[1]Data: https://github.com/amazon-science/dimension-extraction-dataset

tion it as: length $\times$ breadth/width $\times$ height, which creates confusion between what to be considered as length or width. As shown in Figure 1, product images on the other hand, can be used to extract much reliable information in a standardised way. So to make the catalog more consistent, complete and correct, we propose an end-to-end system to extract dimension attributes from product images.

1.1. Related Works

The last decade has witnessed numerous literature [5] [13] which aim to extract attributes from textual data using classical machine learning. Recent researches like OpenTag [19] applies Bidirectional-LSTM followed by conditional random field; similarly LaTeX-Numeric [10] extracts numeric attributes from textual data by solving a NER problem using BiLSTM-CNN-CRF model [9]. With improvements in computer vision techniques, there are a few exemplary works which use product images to extract attribute values primarily from fashion based products. Baloian *et al*. [2] detects colour and texture from product images using kNN classifier over the features extracted from intermediate layers of pre-trained Resnet architecture. Adhikari *et al*. [1] and Parekh *et al*. [11] also extract style, occasion, pattern attributes from product images using image classifiers like Resnet-34 and EfficientNet-B0 respectively. On similar grounds, Zhou *et al*. [20] uses features from sections of the product images to improve recommendations. These works highlight the scope of the ample information that can be extracted from product images. Since information exist in both image and textual data, more multimodal approaches for information extraction have recently been devised. Zhu *et al*. [21] fundamentally solve the NER problem by tagging each word in the product description as an attribute value but with the inclusion of ResNet image features using Cross-Modality Attention. Inspired by visual question answering, PAM [8] is an intricate multimodal transformer architecture to extract 'Item Form' and 'Brand' attribute values from product text and images. It applies OCR (Optical Character Recognition) to extract words from images, uses Faster-RCNN to extract important features from the image and also consider BERT embeddings of each token in the product title. We propose a much simpler but effective image based dimension attribute extraction solution just using images and OCR results; unlike PAM which also uses unstructured text like titles. To the best of our knowledge, the proposed model is the first to extract dimensional attributes from product images and we show very high precision and recall in doing so.

2. Problem Definition

A product $p \in P$ in an e-commerce website is depicted through a set of multiple images $I^p = \{I^p_1, I^p_2, ..., I^p_N\}$ and belongs to a product category $c \in C$, where C is the set of all product categories. Let $I^p_D \subset I^p$, be the subset of images which have dimensional information in it. When the product's functional side is in the front view, we define length as front to back measurement, width as left to right measurement and height as top to bottom measurement, as shown in Figure 1. The task is to extract dimensional attributes like length, width and height of all products $p \in P \mid n(I^p_D) \neq 0$.

3. Methodology

In this paper, we put forth an end to end system to detect dimensional attributes of the product from its images. The system comprises of a *Filter classifier*, which boost the system's efficacy by filtering out images without dimensional attributes; an *OCR engine* to extract texts from the image; a *Parser* to parse measurement values from the words detected by OCR and finally a model (*Bounding-Box classifier*) to classify the OCR-texts as a dimensional attribute.

3.1. Filter Classifier: Filtration of images without dimensional information

Since one product may have multiple images and not all images have dimensional description. Hence, application of the extraction algorithm on all images is highly inefficient and unnecessary. To reduce the OCR induced cost, we trained a MobileNetV3-L architecture to predict a binary output if the input image carries dimensional information in it. It is fed with a high resolution input image (600 pixels) so that it does not lose out the tiny text and thin dimensional axes/lines in the image, which are very important features to classify dimensional images.

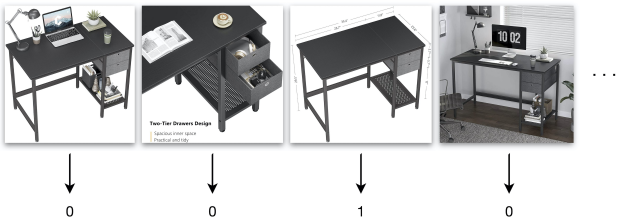


Figure 2: Filter Classifier filtering out the non-dimensional images of a product.

3.2. OCR engine: Detect Text from Images

We deploy an OCR engine to detect text from product images. The OCR engine is able to detect words from the image within $\pm 90^\circ$ orientation of the horizontal axis. Given an image with texts, the OCR engine returns the detected words along with the coordinates of their bounding-box. Next, we find bounding-boxes which are in vicinity of each other and upholding the transitive property; merge them by calculating union of the bounding-boxes and concatenating the words in appropriate order. For example, we

merge the group of nearby bounding boxes which detected "20", "½", and "inch" into one bounding box with value "20 ½ inch". At the end, for simplicity, we only consider the coordinates of their centroid as the location coordinates of the bounding box.

3.3. Parser: Extraction of measurement value from text tokens

We post-process the OCR tokens by tokenizing them, detecting errors and correcting them by replacing the faulty token with a valid vocabulary token [7]. For example, correcting a detected "5O inoh" to "50 inch". Next we use a regex based parser to extract a numerical value along with its unit of measurement (UOM) which corresponds to length entity like centimeter, inch, etc. OCR sometimes fail to detect the text corresponding to the UOM in a bounding-box, for *e.g.*, the double quotes character (") corresponding to inches is often missed by OCR. To resolve such cases of missed out UOM, we deduce the most probable UOM using the other detected OCR tokens in the image and assign it to the bounding-box in question, provided the measurement value does not become an outlier. Bounding boxes in which the parser did not detect any measurement value were ignored.

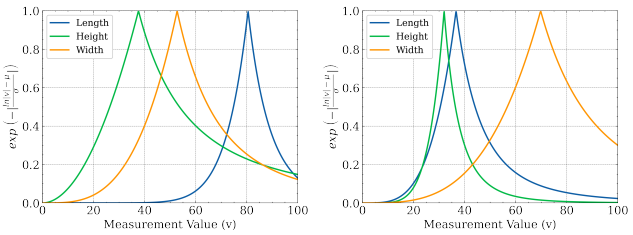
3.4. Bounding-Box Classifier: Classify bounding-box as Length, Width, Height or None

For a product p with an image I_d^p having dimensional information, the model so far detected k bounding-boxes, $\{bb_1, bb_2, ..., bb_k\}$ where each bounding-box bb_i is represented by the pixel coordinates of the bounding-box: $(x_{min}, x_{max}, y_{min}, y_{max})$ and corresponds to a value of measurement: v_i in inches. In this section, we aim to classify each bounding-box to be either Length, Width, Height or None (does not belong to any LWH attributes).

3.4.1 Statistic Inference from the product category

Dimension attributes are highly dependent on the product category that they belong to. For instance, length of a mattress usually lie between 70 inches to 90 inches, whereas their height generally never exceeds 10 inches. Hence, the product category along with the measurement value detected inside the bounding-box have quite a high influence on the true attribute. To integrate this product-category information, one way was to train different models for each product-category, but this would lead to model proliferation. Rather, we devise a novel technique to feed the product category information to our model.

First, we convene all possible values of length, width and height from the catalog (in inches), find their logarithmic values and fit a normal distribution, $\mathcal{N}_{(c,\alpha)}$ with mean $\mu_{(c,\alpha)}$ and standard deviation $\sigma_{(c,\alpha)}$, corresponding to each



(a) Product Category: Bed (b) Product Category: Sofa

Figure 3: Plot of $\bar{z}_\alpha$ with respect to measurement value v

attribute $\alpha \in \{\text{Length, Width, Height}\}$ and every product category $c \in C$. We consider these $3 \times n(C)$ distributions as a metadata. Next, for each bounding-box, given the product category c and the detected measurement value v, we calculate $\bar{z}_\alpha, \forall \alpha \in \{\text{Length, Width, Height}\}$ using Equation 1 and Equation 2.

$$z_\alpha(v, c) = \frac{ln(v) - \mu_{(c,\alpha)}}{\sigma_{(c,\alpha)}} \tag{1}$$

$$\bar{z}_\alpha = e^{-|z_\alpha(v,c)|} \tag{2}$$

As shown in Figure 3, it is worth noting that $\bar{z}_\alpha$ captures the likelihood of v to be considered as the α attribute. If v is an outlier to $\mathcal{N}_{(c,\alpha)}$, $|z_\alpha|$ is high and hence $\bar{z}_\alpha(v, c)$ is small in magnitude, which however is reversed when v is near to the peak of $\mathcal{N}_{(c,\alpha)}$. For instance, a measurement value of 35 inch in a bed product category, results to $(\bar{z}_L, \bar{z}_W, \bar{z}_H) \approx (0.00039, 0.25984, 0.86975)$ and since $\bar{z}_L$ is quite small, the model can easily rule out the possibility of 35 inch being the length of the bed and add more weight to the odds of it being a height. Similarly if $v = 10$ inch is detected from an image in bed product category, the values $(\bar{z}_L, \bar{z}_W, \bar{z}_H) \approx (0.00000, 0.00423, 0.07548)$, can easily indicate that the bounding-box is not associated with any attribute, and hence would likely get classified as None. These values help the model understand if v stands as an outlier to the category's α attribute.

3.4.2 Single-Box Classification network

The location of the text token with respect to the image is a vital feature for deciding the attribute associated with the bounding-box. For instance, the height information is usually present next to a vertical line or arrow in the image. Hence we intend to use the co-ordinates of the centroid along with the image as a input for classification. Given a bounding-box $bb_i \equiv (x_i, y_i, v_i)$ corresponding to a 3-channel RGB image $I \equiv (I_R, I_G, I_B)$, we construct a 3-channel input using Equation 3, 4 and 5 respectively. The motivation is to assist the model with information to understand the importance of orientation of various lines in

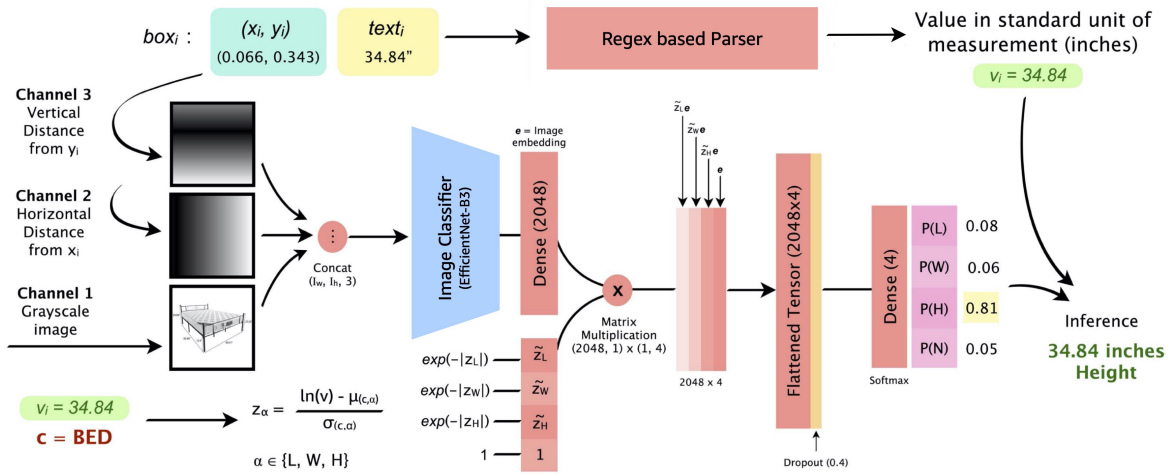


Figure 4: Architecture of Single Box Classification network.

the image and their proximity from the bounding box. This feature generation technique enables the model to add more attention to image features near the bounding box.

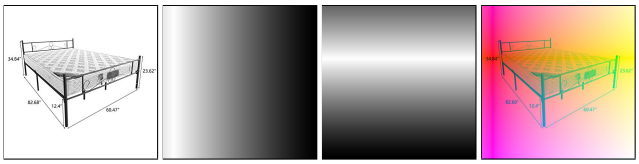
$$f_1(x,y) = 0.299 \times I_R(x,y) + 0.587 \times I_G(x,y) + 0.114 \times I_B(x,y) \quad (3)$$

$$f_2(x,y) = |x - x_i| \quad (4)$$

$$f_3(x,y) = |y - y_i| \quad (5)$$

where x and y correspond to the horizontal and the vertical pixel coordinates normalized by the width and height of the image respectively, such that, $0 \leq x \leq 1$ and $0 \leq y \leq 1$. $f_1(x,y)$ defines the grayscale conversion of the RGB image at the pixel (x,y), while $f_2(x,y)$ and $f_3(x,y)$ encodes the horizontal and vertical distance of the pixel (x,y) from the bounding box. In this use case, the colour of image isn't of much significance, as colors of the item are not related to the dimension related attribute values and hence we chose to convert the RGB images to its grayscale variant. Also this particular choice of input creation creates 3-channel input and hence enabling us to initialize our models using Imagenet weights for faster model convergence.

Architecture: The input image is encoded as a 3-channel feature as described above, which is then passed through an image classifier (EfficientNet-B3), which outputs an embedding e of size $(e_d, 1)$. Next we use $\bar{z} = [\bar{z_L}, \bar{z_W}, \bar{z_H}, 1]$ of size $(1, 4)$ as defined in 3.4.1 to compute the product of e and $\bar{z}$, i.e., $e\bar{z} = (\bar{z_L}e, \bar{z_W}e, \bar{z_H}e, e)$. The resulting vector of size $(e_d, 4)$ is flattened and followed by a dropout layer and a dense layer with $|\alpha| + 1$ nodes, each corresponding to the attributes (length, width, height) and None category. The hypothesis is to spike the set of neurons corresponding to $\bar{z_\alpha}e$, when the measurement value v highly correlates with



(a) Channel 1 (b) Channel 2 (c) Channel 3 (d) Stacked

Figure 5: 3-channel Feature Generation using the original RGB Image and the coordinates of the bounding-box. The reddish color in the stacked image highlights the pixels near the query bounding box.

the product category's α attribute. The last column of $e\bar{z}$ is e, which acts like a shortcut connection of the image embeddings. This makes the model robust to outlier products like kids-chair (smaller than usual chair) or bunk-bed (taller than usual beds) where even though the first three columns of $e\bar{z}$ are approximately zero, the model can directly predict the attribute using the image embedding e.

3.4.3 Multi-Box Classification Network

Architecture: Motivated by Dosovitskiy *et al.* [4], the input image $I \in \mathbb{R}^{H \times W \times C}$ is split into a sequence of patches of size $P \times P$ pixels, resulting in $N = HW/P^2$ number of patches. These patches are flattened and mapped to a D dimensional vector (patch embeddings) using a trainable linear projection. Position embeddings derived from the position of the image patch, are added to the patch embeddings to retain positional information. We use standard learnable 1D position embeddings as done by Dosovitskiy *et al.* [4]. Simultaneously, every bounding box detected in the image denoted by $bb_i = (x_i, y_i)$, are encoded into embeddings of size D by concatenating the

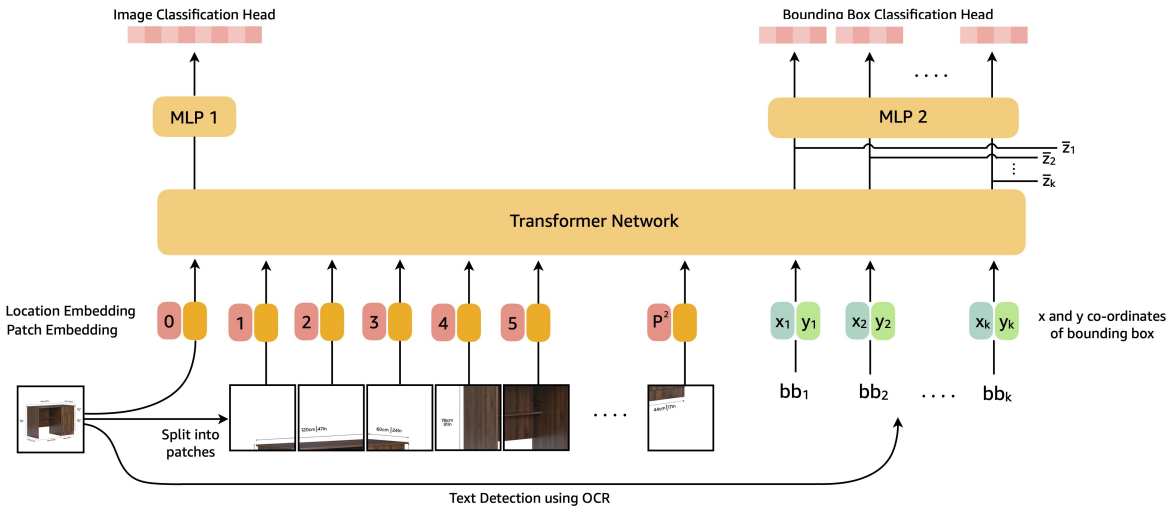


Figure 6: Architecture of Multi-box Classification Network

embeddings of x_i and y_i, each having dimension $D/2$. Later, N patch embeddings along with k bounding box embeddings are passed to a multi-layered bidirectional transformer network. The output embeddings of the bounding boxes are later multiplied with $\bar{z_i}$, where $\bar{z_i}$ is the statistical likelihood of v_i to be considered as one of the attributes as discussed in 3.4.1. The resultant vector is followed by a dropout layer and finally connected with a dense layer with $|\alpha| + 1$ nodes.

Pre-training tasks: To improve the transformer learning process, we follow two pre-training tasks.

1. Image Classification Task: The transformer network is trained to classify the input images and predict the product category that it belongs to. As shown in Figure 6, the embedding of the extra learnable placeholder is passed through an MLP network with one hidden-layer and a softmax activation. This increases the models knowledge about product classification using images and helps it to generalise any task for multiple product categories.

2. Text Detection Task: To better fuse the image patches with the bounding box location co-ordinates, we pass random co-ordinates (where no text is present in the image) to randomly chosen $(p = 0.5)$ bounding box placeholders and the rest with correct location co-ordinates where the OCR engine detected a text. Finally, the output embeddings of the bounding boxes are passed to an another MLP network with a sigmoid activation layer and trained using binary cross-entropy loss to classify if the image segment at the corresponding input co-ordinates has text or not.

4. Experiments and Results

4.1. Dataset

We create two datasets, one for training the image filtering classifier and another for training the bounding-box classifier, for which we considered products across 68 categories. To avoid incorrect and missing annotations, outliers were removed from the catalog and we only considered products which have all the dimensional attributes filled.

1. `FC_AUTO_TRAIN`: To train the Filter Classifier, we considered $\approx$ $600k$ products with $\approx$ $4.5M$ images from amazon.co.uk and apply OCR and the parser to detect text with dimensional information inside the image. Out of the $\approx$ $4.5M$ images, measurement values were detected from only $\approx$ $280k$ images which were labelled as 1. Assuming that the remaining images do not have dimensional information, we sample $\approx$ $300k$ images and labelled them as 0.

2. `BBC_AUTO_TRAIN`: To procure a large scale automated train data for pre-training the bounding box classifier, along with $\approx$ $600k$ products from amazon.co.uk, we also considered $\approx$ $5.1M$ products from various other marketplaces, which had $\approx$ $40M$ images. Next, we filter out the images without dimensional description using the Filter classifier trained using `FC_AUTO_TRAIN` and end up having $\approx$ $2.4M$ images. We apply OCR to every image $I \in I_D$ to receive k bounding-boxes $bb_{i,I}$ and detect corresponding value of measurement $v_{i,I}$ using the parser, $\forall i \in 1, 2..., k$. Next, we determine the relative error $\delta_{\alpha,i,I}$ using Equation 6, where v_α is the true measure-

ment value of the product's α attribute in the catalog, where $\alpha \in L, W, H$.

$$\delta_{\alpha,i,I} = \frac{|v_{i,I} - v_\alpha|}{v_\alpha} \tag{6}$$

If $\min_{\alpha \in \text{L,W,H}} \delta_{\alpha,i,I} \leq 10^{-3}$, we tag $bb_{i,I}$ as $\operatorname{argmin}_{\alpha \in \text{L,W,H}} \delta_{\alpha,i,I}$, else we tag $bb_{i,I}$ as None.
Lastly, with few post processing, and basic outlier removal, we create a dataset with $\approx 4.4M$ bounding-boxes corresponding to $\approx 1.3M$ images.

3. `BBC_GDS`: To finetune and evaluate the models with high quality data, we manually annotated 200 dimensional images for every product type and split 150 for training (`BBC_GDS_TRAIN`) and 50 for testing purpose (`BBC_GDS_TEST`). This dataset is published online[2].

4.2. Training and Implementation Details

Single Box Classification Network: Ideally the single box classification network can be constructed using any image classifier. In our experiments, we chose EfficientNet-B3, due to its high performance and efficiency [17]. Moreover, we tested various state of the art image classifiers and empirically found that EfficientNet-B3 was optimal in terms of both computation and performance. The whole architecture was trained using Adam optimizer with step-decay learning rate varying from 10^{-3} to 10^{-5}.

Multi Box Classification Network: As discussed in subsection 3.4.3, the multi-box classifier network requires a multi-layered bidirectional transformer network to share information between the image features and the bounding-box location features. The transformer encoder network of the Multi Box Classification network has 12 layers, input image patch size as 16, a hidden size of 768, and 12 self-attention heads. This network was trained using Adam with weight decay optimizer (AdamW) with step-decay learning rate varying from 10^{-3} to 10^{-5}, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

4.3. Model performance

4.3.1 Filter Classifier

This model plays a vital role of optimizing the extraction process. High quality OCR engine consumes $\approx 100ms$ per image and incurs heavy computation cost. Also, since most images ($\approx 97\%$) do not have any dimensional attributes, applying the extraction process to all images of the product is ineffective and not scalable. To tackle the drawback, the filter classifier comes to our aid, which processes an image in just $\approx 4ms$ (on 1 Tesla V100) making the process highly efficient (14x). The filter classifier, backed by a state of the art image classifier works at 82.21% precision and 98.06% recall in the test set of Dataset-1. To understand the impact of Filter classifier on the overall system's precision and recall, we compare the setup of skipping the Filter classifier against when using it. As shown in Table 1, skipping the filter classifier improves the recall but also decreases the precision, as the number of incorrect extraction from non-pure dimensional images (Figure 7) increases.

Setting	Precision	Recall	F1-score
Filter classifier skipped	0.9103	0.9010	0.9056
Filter classifier applied	0.9154	0.8975	0.9064

Table 1: Effect of not applying Filter classifier, i.e., using every image for attribute extraction.

(a) Product Image (b) Magnified Image

Figure 7: A sample dimensional image with dimensional text but no visual mapping or dimension spans. These kind of images constitute the majority of the filter classifier's false negatives, and likewise the attribute classifier also perform poorly on these images due to ambiguity.

4.3.2 Comparision with benchmark models/systems

To check the effectiveness of the proposed model and the improvement made by fusing the statistical inferences, we compare the two types of bounding-box classifiers with S.I (proposed architecture) and without S.I, along with various other state of the art approaches.

1. We tried CV based algorithmic approaches which included, detecting lines using Hough transform, computing the angle of the lines closest to the bounding box containing the dimension value and lastly using a set of hard-coded rules, logic and checks for every product type, we complete the extraction.

2. LayoutLM [18] is a state of the art model to extract information from scanned documents using text detected by OCR and layout information.

[2]https://github.com/amazon-science/dimension-extraction-dataset

Model	Precision	Recall	F1-score	GFLOPs	Parameters
CV based algorithmic approach	0.8092	0.3088	0.4470	-	-
LayoutLM [18], Trained using $\mathcal{B}$	0.5624	0.3303	0.4162	19.2	113M
LoRRA VQA [14] [15], Trained using $\mathcal{B}$	0.8907	0.8219	0.8549	88.1	-
DeepLab V3+ [3], Trained using $\mathcal{B}$	0.9262	0.8006	0.8588	20.8	12M
YOLOv3 [12], Trained using $\mathcal{B}$	0.8974	0.8045	0.8484	156.4	62M
YOLOv5 [6], Trained using $\mathcal{B}$	0.8792	0.7971	0.8361	49.1	21M
Single Box Classification Network without S.I.				1.8	12M
Trained only using BBC_GDS_TRAIN	0.8716	0.8230	0.8466		
Trained using $\mathcal{B}$	0.8925	0.8459	0.8686		
Single Box Classification Network with S.I.				1.8	12M
Trained only using BBC_GDS_TRAIN	0.9061	0.8302	0.8665		
Trained using $\mathcal{B}$	0.9154	0.8975	**0.9064**		
Multi Box Classification Network without S.I.				17.6	86M
Trained only using BBC_GDS_TRAIN	0.8211	0.7568	0.7876		
Trained using $\mathcal{B}$	0.8468	0.8011	0.8233		
Trained using $\mathcal{B}$, after image classification pretraining	0.8417	0.8255	0.8335		
Trained using $\mathcal{B}$, after text detection pretraining	0.8768	0.8436	0.8599		
Trained using $\mathcal{B}$, after both pretrain tasks	0.8766	0.8507	0.8635		
Multi Box Classification Network with S.I.				17.6	86M
Trained only using BBC_GDS_TRAIN	0.8076	0.7741	0.7758		
Trained using $\mathcal{B}$	0.8708	0.7927	0.8299		
Trained using $\mathcal{B}$, after image classification pretraining	0.8541	0.8320	0.8429		
Trained using $\mathcal{B}$, after text detection pretraining	0.9010	0.8419	0.8704		
Trained using $\mathcal{B}$, after both pretrain tasks	0.8937	0.8516	0.8721		

Table 2: Comparision of the proposed architectures with the benchmark models. Note 1: Trained using $\mathcal{B}$ implies the model was first pretrained using BBC_AUTO_TRAIN and then finetuned using BBC_GDS_TRAIN

3. Pythia v0.3 + LoRRA [14] [15] is a VQA model which uses both image features and OCR tokens to answer a natural language question. To solve our usecase, we ask three questions corresponding to each attribute and limit the model to generate answer strictly based on OCR tokens. Pretrained LoRRA (on TextVQA) was finetuned using our dataset.

4. YOLOv3 [12] and YOLOv5 [6] are used to detect bounding-box around the text with dimensional value and classify them as an attribute. Lastly, using IOU between the YOLO-bounding-box and the OCR-bounding-box, we tag the attribute predicted by YOLO with the measurement value inside the intersecting OCR-bounding-box.

5. DeepLabV3+ [3] is a semantic segmentation model used to classify each pixel in the image as one of the attributes. Later we calculate the mean of softmax of predictions from each pixel inside the OCR bounding boxes and use it to predict the attribute associated with the box.

The way dimensional images are represented in the catalog is quite complex and diverse. Owing to which the algorithmic approach generates a moderate precision but lags to effectuate a high recall. Clearly indicating that the attribute classification task requires a deep dive approach to analyse and explore deep learning methodologies. In order to classify a bounding-box as an attribute, the image features in its vicinity is very influential and although LayoutLM considers the coordinates of the bounding-box, it lacks the context of the box's neighbourhood visual features. Hence we deduce that LayoutLM might be an ideal model for text-heavy input, but it fails to perform for usecases in which image features are crucial. LoRRA VQA is designed to answer natural language questions and is often robust to the way the question is asked, but it fails to outperform the proposed architecture in dimension extraction. YoloV3, YoloV5 and DeepLabV3+, although is efficient as it processes an image only once; but it transforms the simple classification task into a much more complex problem of text-localisation and classification simultaneously, hence resulting in lesser recall. On the other hand, both the proposed architectures, leverage the OCR's capability to solve the text-localisation problem and just focus on the attribute classification task.

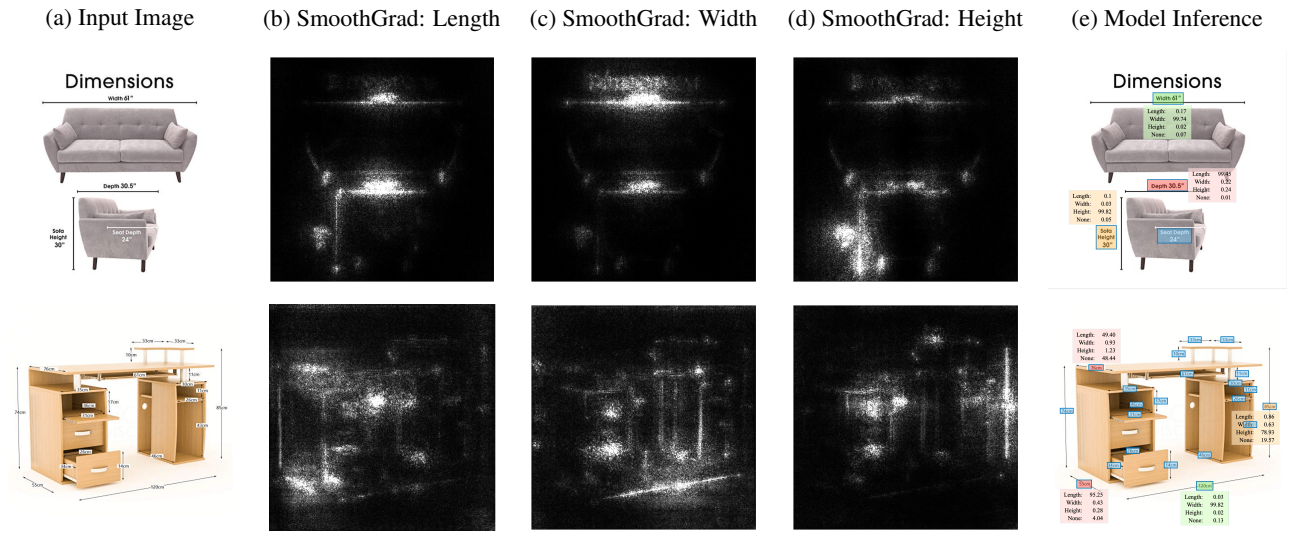


Figure 8: Visualisation of the image classifier's gradients and inferences. Column a displays few exemplary product images with dimensional information; Column b, c, d plots the saliency map of the input image using SmoothGrad [16] with respect to the three detected attributes respectively; Column e unveils the inferences of $\mathcal{D}$-Extract.

Table 2 shows that fusion of statistical inference with the architectures, enhances the model's performance, especially with the Single Box Classification network where we observe $\approx$ 3.78% improvement in F1-score. The proposed single box classification network with statistical inference has the highest F1-score of 0.9064 and is thus the best system demonstrated so far to extract dimensions from images. While in terms of efficiency, multi-box classification network is considered suitable for images that contains numerous dimension related text. It has the highest F1-score when compared to all benchmark models which processes the image only once like Yolo and DeepLabV3+.

5. Discussions

The single classifier model with S.I., hereafter referred as $\mathcal{D}$-Extract, achieves 95.65% precision at 85% recall for the *Height* attribute, whereas 91.01% and 91.53% precision at 85% recall for the *Length* and *Width* attribute respectively. The *Length* and *Width* confusion still exist, but it has >90% precision in length and width attributes which is a substantial improvement over the existing $\approx$ 30% error rate in text extraction methodologies. Further analysing the errors, we found lower F1-score for length and width attributes in product types which has ambiguous front and side view definition like safe, storage-box, etc. making these attributes indistinguishable, even for humans. Moreover, it is found that the dimension errors by interchanging length and width for these type of products result in fewer customer returns as they can be placed in any orientation.

Diving deeper into the image classifier's learnings, we plot image-specific attribute-wise saliency map using SmoothGrad [16]. Saliency map is a heatmap where white regions correspond to regions that have higher impact on the model's final decision. Figure 8 demonstrates that the image classifier proficiently learns the important boundary edges of the actual product, detects the thin dimension line corresponding to each attribute and makes reasonable inferences using them. Furthermore, it is notable to note that the image classifier understands the various orientation of the actual product and also considers the possibility that a single image might have multiple views of the product (first example of Figure 8). The second example (desk) in Figure 8 is a seemingly difficult task with 22 dimensional values mentioned inside the image, but the model is able to detect the most relevant dimensional lines and tag them accurately.

6. Conclusion and Future Work

In short, we propose a highly accurate but effective end-to-end pipeline enabling e-commerce websites to extract dimensional attributes from product images using an OCR engine, regex based parser and an image classifier coupled with product-category specific dimension priors. Our experimental results show that the model operates at a F1-score of 0.8889, 0.8958 and 0.9344 for length, width and height attributes respectively and outperforms other state of the art architectures by $\approx$ 4.76% in F1-score. Future goal is to extend the model to more product categories, attribute types like seat height, diameter etc. and include visual reasoning with complex layouts like tables and diagrams.

References

[1] Sandeep Singh Adhikari, Sukhneer Singh, Anoop Rajagopal, and Aruna Rajan. Progressive fashion attribute extraction. *arXiv preprint arXiv:1907.00157*, 2019.

[2] Andres Baloian, Nils Murrugarra-Llerena, and Jose M Saavedra. Scalable visual attribute extraction through hidden layers of a residual convnet. *arXiv preprint arXiv:2104.00161*, 2021.

[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Rayid Ghani, Katharina Probst, Yan Liu, Marko Krema, and Andrew Fano. Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1):41–48, 2006.

[6] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu, changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, Oct. 2020.

[7] Ido Kissos and Nachum Dershowitz. Ocr error correction using character correction and feature-based word classification. In *2016 12th IAPR Workshop on Document Analysis Systems (DAS)*, pages 198–203. IEEE, 2016.

[8] Rongmei Lin, Xiang He, Jie Feng, Nasser Zalmout, Yan Liang, Li Xiong, and Xin Luna Dong. Pam: Understanding product images in cross product category attribute extraction. *arXiv preprint arXiv:2106.04630*, 2021.

[9] Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*, 2016.

[10] Kartik Mehta, Ioana Oprea, and Nikhil Rasiwasia. Latex-numeric: Language-agnostic text attribute extraction for e-commerce numeric attributes. *arXiv preprint arXiv:2104.09576*, 2021.

[11] Viral Parekh, Karimulla Shaik, Soma Biswas, and Muthusamy Chelliah. Fine-grained visual attribute extraction from fashion wear. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3973–3977, 2021.

[12] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[13] Keiji Shinzato and Satoshi Sekine. Unsupervised extraction of attributes and their values from product description. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1339–1347, 2013.

[14] Amanpreet Singh, Vivek Natarajan, Yu Jiang, Xinlei Chen, Meet Shah, Marcus Rohrbach, Dhruv Batra, and Devi Parikh. Pythia-a platform for vision & language research. In *SysML Workshop, NeurIPS*, volume 2018, 2018.

[15] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[16] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[17] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[18] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1192–1200, 2020.

[19] Guineng Zheng, Subhabrata Mukherjee, Xin Luna Dong, and Feifei Li. Opentag: Open attribute value extraction from product profiles. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1049–1058, 2018.

[20] Wei Zhou, PY Mok, Yanghong Zhou, Yangping Zhou, Jialie Shen, Qiang Qu, and KP Chau. Fashion recommendations through cross-media information retrieval. *Journal of Visual Communication and Image Representation*, 61:112–120, 2019.

[21] Tiangang Zhu, Yue Wang, Haoran Li, Youzheng Wu, Xiaodong He, and Bowen Zhou. Multimodal joint attribute prediction and value extraction for e-commerce product. *arXiv preprint arXiv:2009.07162*, 2020.