Workforce Attire Project

Authors: Kacper Grzymkowski, Illia Tesliuk, Adam Narożniak, Jakub Fołtyn

1. Project description

Detection and classification of people based on their clothing (requires an acquisition of a dataset)
- type and color of the head cover (or its lack)
- type of topwear (shirt, t-shirt, vest)
- type of bottomwear (pants, skirt, jeans)
- type and color of shoes

- the presence of a badge (or some identifier)
- presence of accessories (scarf, headset)
The dataset should include people in various positions, groups of people
The second phase of the project would call for the classification of a role of the person based on the presence/absence of clothing components (based on the labeled images of people dressed certain way)

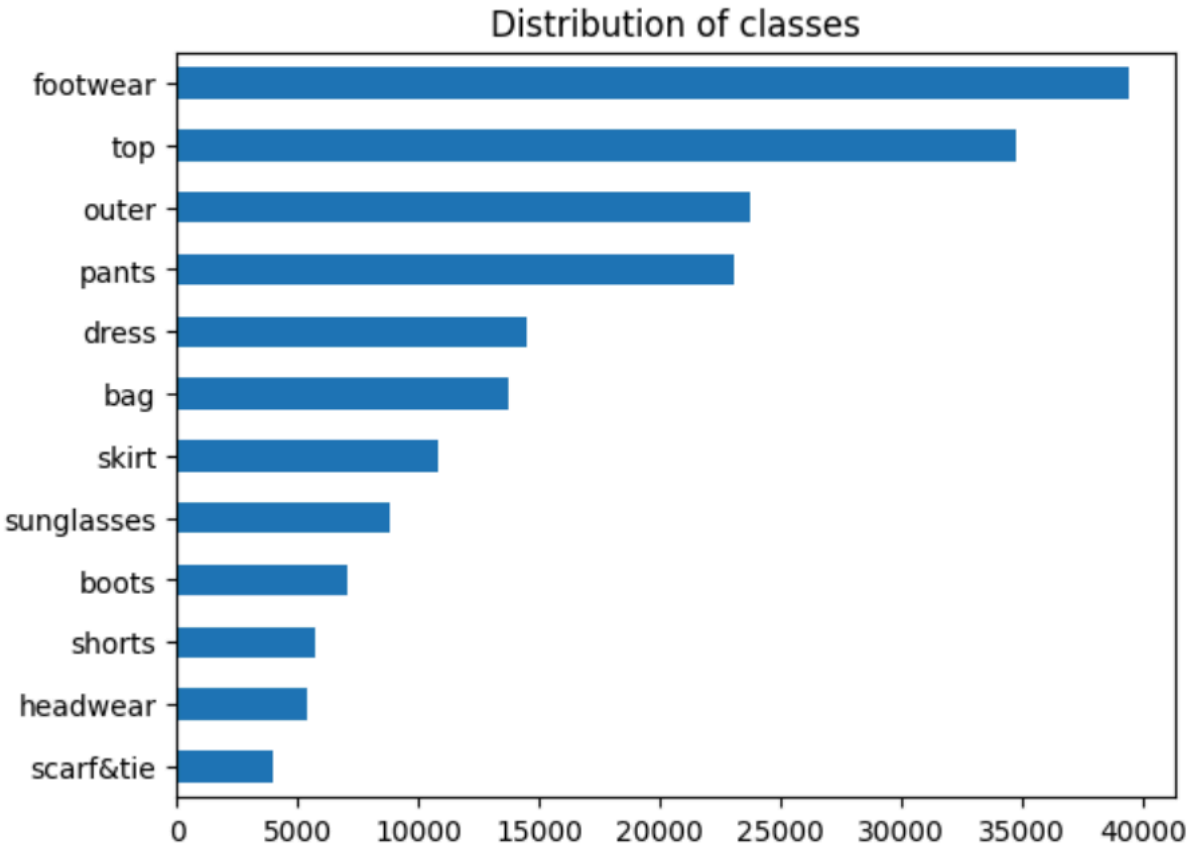
2. Datasets review

ModaNet

Modanet is a dataset of 55,176 street fashion images in *.jpg* format and 265,705 annotations prepared by the eBay CoreAI research team in 2018. The dataset consists of a subset of images taken from the PaperDoll image set - a collection of fashion images that were web-crawled from Chictopia website in 2012. The vast majority of images have the resolution of (400, 600), with only 22 images sized at (350, 400). The annotated images were chosen to ensure a diverse range of human poses, thereby enhancing the robustness of the models.

Modanet provides three types of annotations, namely pixel-level segmentation masks, polygonal annotations and bounding box annotations. Polygonal annotations contain coordinates of polygons enclosing individual fashion items. They record the shape information of the object boundary and serve as an alternative way to segment objects. Rectangular bounding box annotations are inferred from polygons to enable fashion item detection belonging to one of the 13 meta categories:

1. **bag**
2. **belt**
3. **boots**
4. **footwear**
5. **outer** (coat/jacket/suit/blazers/cardigan/sweater/Jumpsuits/Rompers/vest)
6. **dress**
7. **sunglasses**
8. **pants** (pants/jeans/leggings)
9. **top** (top/blouse/t-shirt/shirt)
10. **shorts**
11. **skirt**
12. **headwear**
13. **scarf & tie**



Example annotation:

{'segmentation': [
[140, 274, 126, 309, 119, 354, 121, 406, 12 444, 196, 435, 203, 444, 224, 442, 227, 419, 219, 401, 217, 347, 209, 304, 205, 289, 199, 284, 192, 286, 195, 305, 200, 325, 203, 347, 206, 366, 197, 373, 187, 366, 185, 354, 189, 330, 171, 293, 163, 274]],
'area': 18468,
'iscrowd': 0,
'image_id': 1025382,
'bbox': [119, 274, 108, 171],
'category_id': 11, 'id': 231371}
{'segmentation': [
[140, 525, 137, 531, 152, 538, 155, 543, 149, 544, 144, 537, 133, 532, 129, 540, 130, 562, 131, 575, 138, 588, 173, 588, 179, 586, 180, 581, 174, 567, 166, 552, 162, 542, 159, 536],
[161, 520, 158, 527, 163, 542, 166, 553, 171, 554, 174, 556, 179, 556, 181, 551, 171, 536, 173, 528, 172, 522]],
'area': 828, 'iscrowd': 0, 'image_id': 1025382, 'bbox': [158, 520, 23, 36], 'category_id': 4, 'id': 231373}
{'segmentation': [
[192, 141, 200, 142, 206, 142, 209, 145, 210, 150, 213, 150, 216, 149, 219, 149, 223, 153, 223, 160, 219, 163, 213, 161, 212, 151, 208, 152, 205, 156, 200, 156, 197, 152, 198, 146]], 'area': 682,
'iscrowd': 0,
'image_id': 1025382,
'bbox': [192, 141, 31, 22],

'category_id': 7, 'id': 231374}
Example image:

DeepFashion2

DeepFashion dataset contains real-life (customer) and professional (commercial) images of people with annotations of 13 different clothing.
The dataset is divided into 3 splits: train, validation and test. The total size of the dataset is nearly 500 000 images and about 800 000 annotations.
The annotations include:

- short sleeve top,
- long sleeve top,
- short sleeve outwear,
- long sleeve outwear,
- vest,
- sling,
- shorts,
- trousers,
- skirt,
- short sleeve dress,
- long sleeve dress,
- vest dress,
- sling dress.

Figure 2: Definitions of landmarks and skeletons.

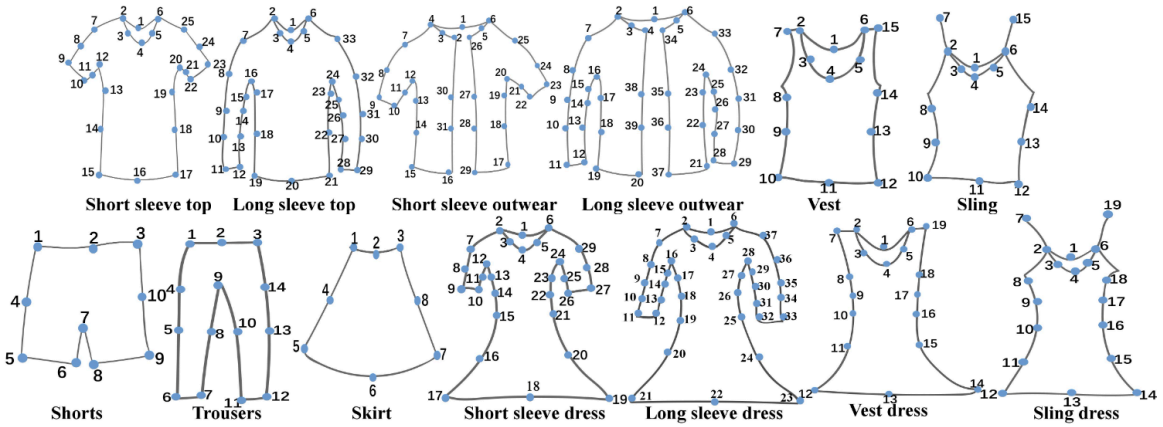


Figure 2: Representation of the images available in DeepFashion2 dataset.
The annotations include a few interesting information beyond the bounding boxes, notably, the information if the occlusion occurs, and the viewpoint (front, back, and side).

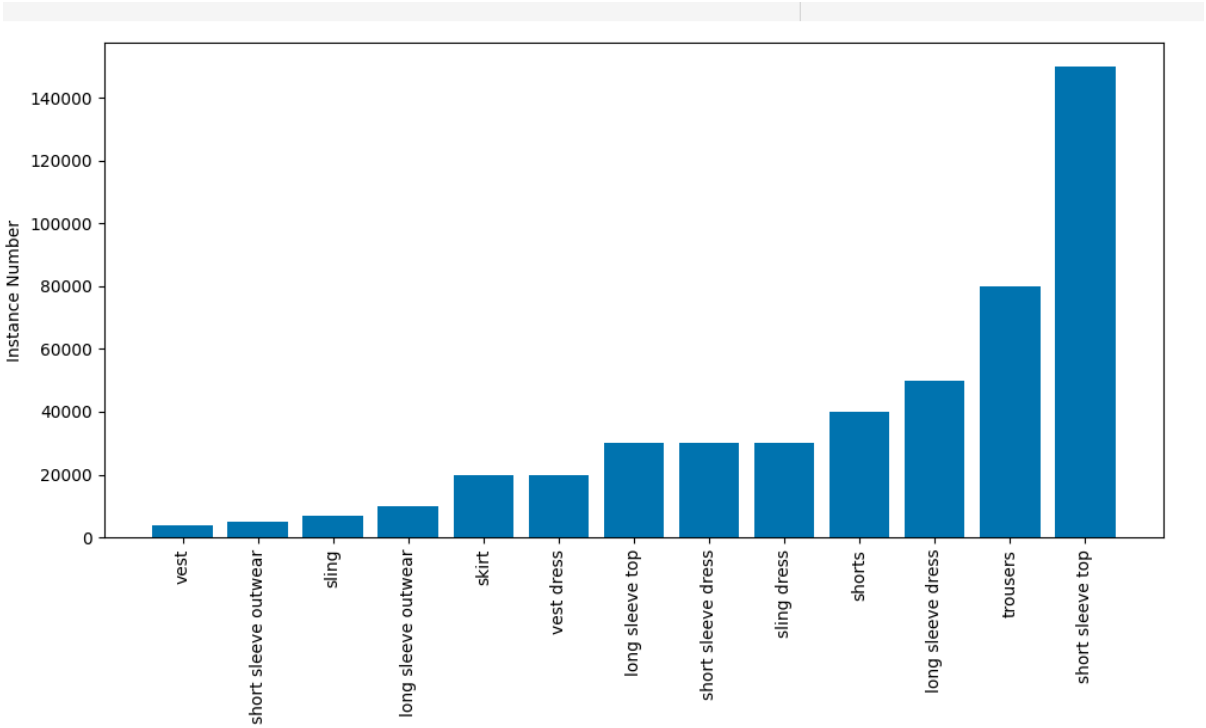
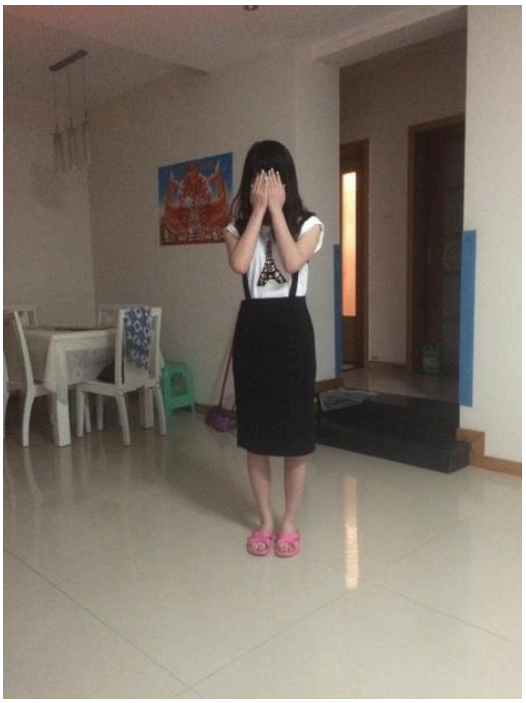


Figure: Count of the different annotations in DeepFashion2.
Example annotation:

```
{
 "item2": {
   "segmentation": [
     [...]
   ],
   "scale": 1,
   "viewpoint": 2,
   "zoom_in": 1,
   "landmarks": [...],
   "style": 2,
   "bounding_box": [
     204,
```

```
      189,
      293,
      414
    ],
    "category_id": 13,
    "occlusion": 2,
    "category_name": "sling dress"
  },
  "source": "user",
  "pair_id": 1,
  "item1": {
    "segmentation": [
      [...]
    ],
    "scale": 1,
    "viewpoint": 2,
    "zoom_in": 1,
    "landmarks":[...],
    "style": 1,
    "bounding_box": [
      199,
      190,
      287,
      269
    ],
    "category_id": 5,
    "occlusion": 2,
    "category_name": "vest"
  }
}
```


Example image:



Clothing detection dataset

This is a relatively small dataset with only 3,332 images. Those are real-life images, of different sizes and in *.jpg* format. Each image depicts a different person. People on the images are in different poses and configurations, there may be more than 1 person on an image.
There are also bounding boxes saved in *.json* format. They contain the box's width, height, x and y coordinates as well as two properties: *genre* and *class*. All labels are in spanish. The **genres** are as follows:

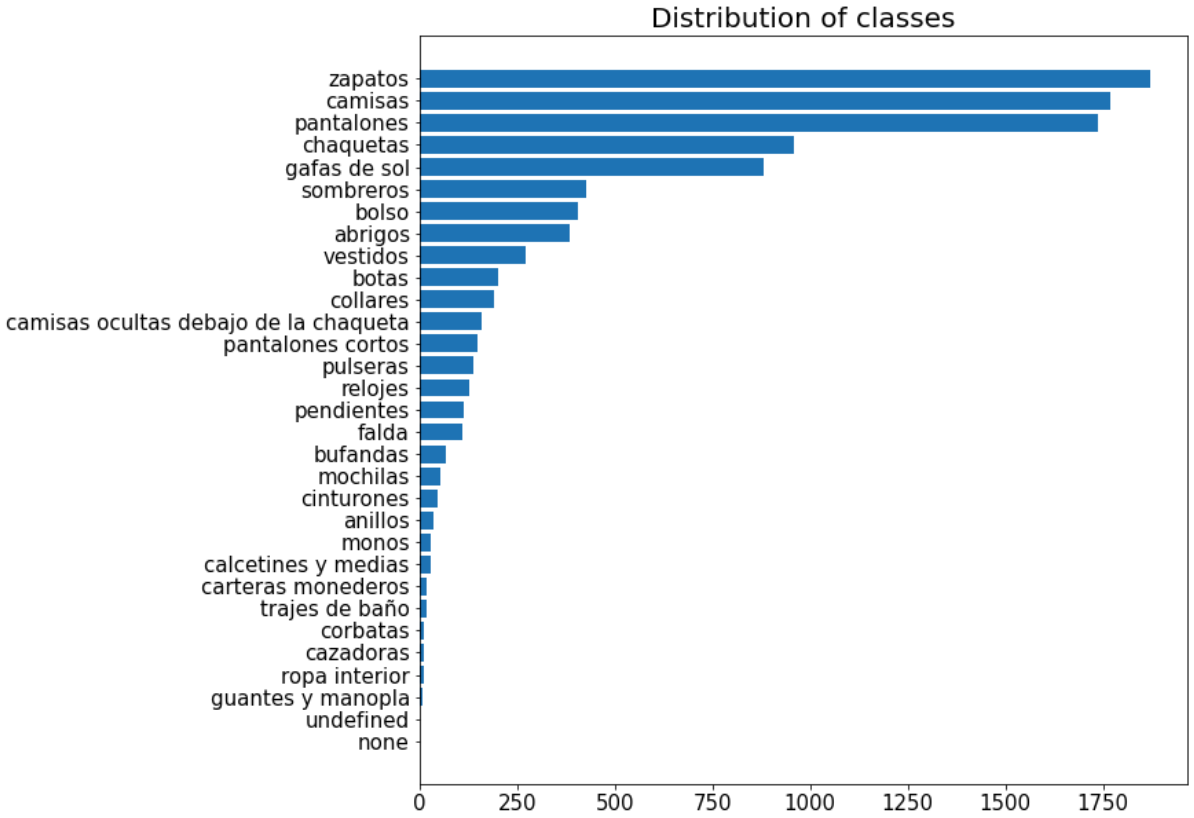
- **bebe** – baby
- **chica** – girl
- **chico** – boy
- **hombre** – man
- **mujer** – woman

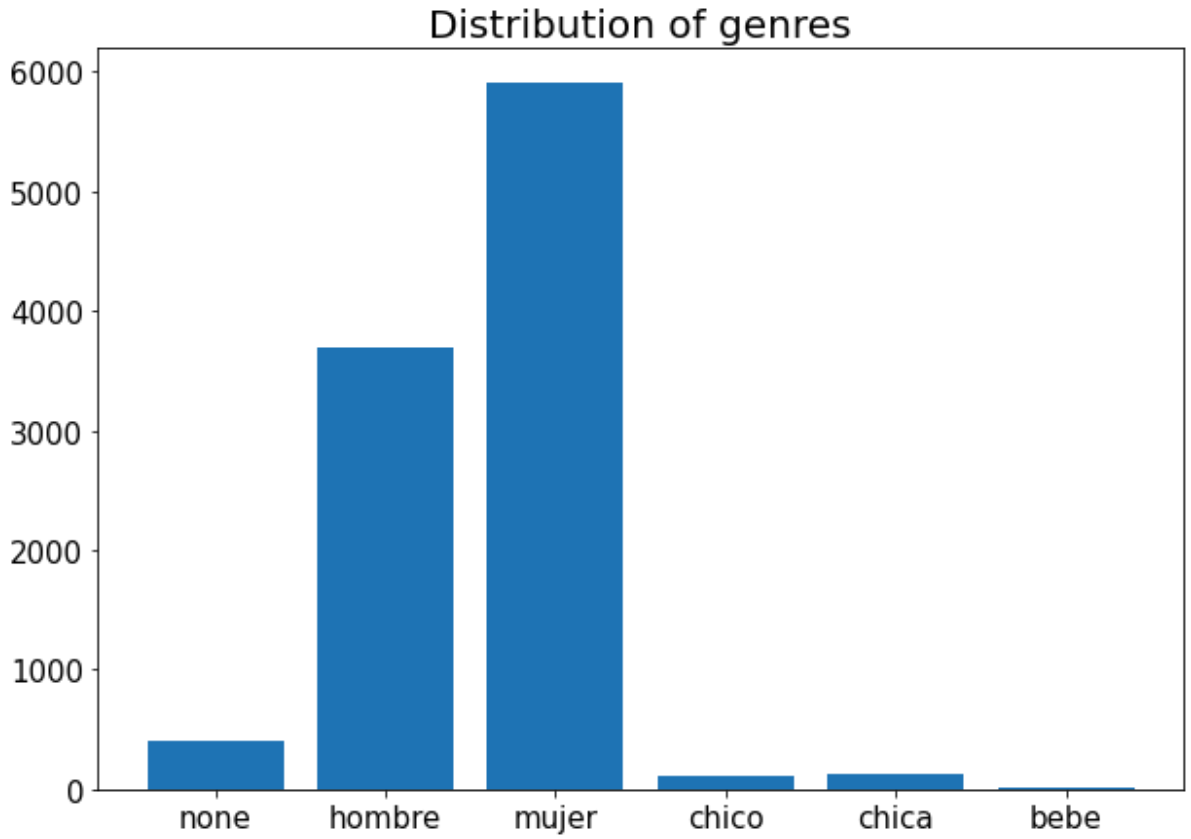
The **classes** are:

- **trajes de baño** –swimsuits
- **bolso** – bag
- **relojes** – watches
- **cinturones** – belts
- **collares** – necklaces
- **guantes y manopla** – gloves and mitten
- **bufandas** – scarves
- **pantalones cortos** – shorts
- **pulseras** – bracelets
- **pantalones** – pants
- **undefined** – undefined
- **abrigos** – coats
- **cazadoras** – coats/flight jackets (merged with jacket in our datasets)
- **monos** – overalls
- **zapatos** – shoes
- **chaquetas** – jackets
- **mochilas** – backpacks
- **corbatas** – ties
- **vestidos** – dresses

- **carteras monederos** – wallets purses
- **falda** – skirt
- **anillos** – rings
- **pendientes** – earrings
- **sombreros** – hats
- **camisas** – shirts
- **botas** – boots
- **camisas ocultas debajo de la chaqueta** – hidden shirts under jacket (merged with shirts in our dataset)
- **calcetines y medias** – socks and stockings
- **ropa interior** – underwear
- **gafas de sol** – sunglasses

Genres and classes distributions:





Example annotation:

{'arr_boxes': [{'x': 832.4105644226074,
'y': 916.6999161243439,
'width': 44.229798316955566,

'height': 51.71341896057129,
'genre': 'hombre',
'class': 'zapatos'},
{'x': 719.7499322891235,
'y': 525.2045542001724,
'width': 185.90759754180908,
'height': 164.91280496120453,
'genre': 'hombre',
'class': 'chaquetas'},
{'x': 249.637833237648,
'y': 635.629503428936,
'width': 129.3572062253952,
'height': 256.1057522892952,
'genre': 'mujer',
'class': 'pantalones'},
{'x': 721.2860655784607,
'y': 681.715714931488,
'width': 156.61130905151367,
'height': 271.1201548576355,
'genre': 'hombre',
'class': 'pantalones'},
{'x': 480.6297969818115,
'y': 498.0154439806938,
'width': 214.21112537384033,
'height': 212.03711181879044,
'genre': 'hombre',
'class': 'chaquetas'},
{'x': 443.4945487976074,
'y': 692.9111480712891,
'width': 212.44125366210938,
'height': 309.17860865592957,
'genre': 'hombre',
'class': 'pantalones'}],
'file_name':
'zzzsf6wbldcoxcl9urth0x3nh7nkd0uupqa67eq09g152bp89z56gry6qqbs49p2.jpg'}
Example image:

Transforming the “Clothing dataset” (in Spanish) to HF dataset

In order to finetune the fashionpedia YOLO, we make the Clothing Dataset accessible in HF Hub. The dataset is properly attributed the license rights and is available publicly to be used https://huggingface.co/datasets/adam-narozniak/clothing..This process includes several steps:

1. Translation of the categories
2. Matching of the categories + extending the dataset by new categories

3. Matching the format of the annotation
    a. transformation to the boxes
    b. adjustment of the categories' numerical values
    c. extraction of images' width and height
4. Creation of the dataset and dataset card
    a. Creation of features
    b. Description
5. Hosting the dataset of HF Hub

The first two steps resulted in 9 new categories. However, 2 of them were merged with the existing ones due to being too specific. So, finally, we support 7 new categories, notably extending accessories by backpacks, earrings, bracelets, necklaces, and rings.

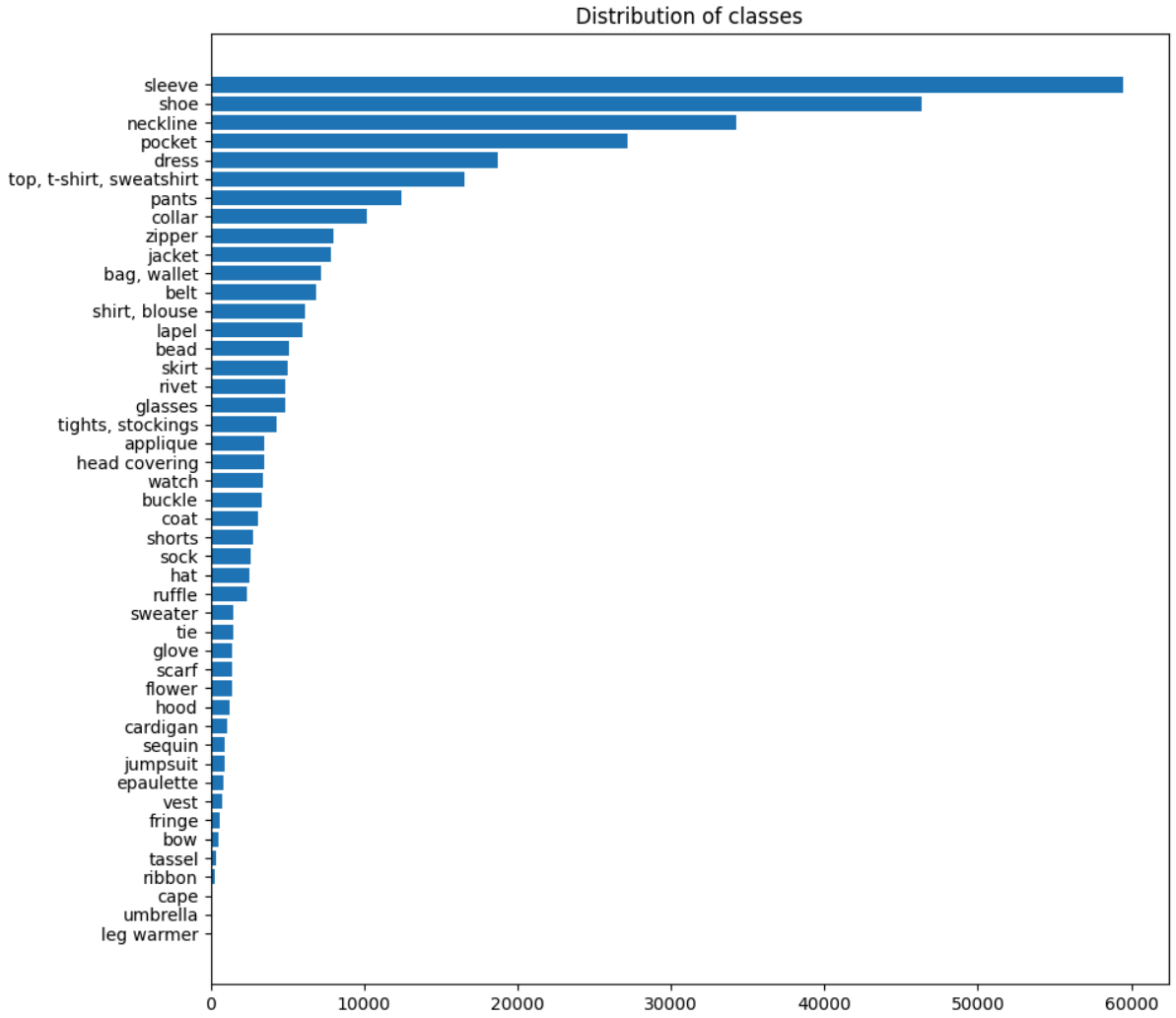
Fashionpedia

A dataset with a total of 48,825 clothing images in daily-life, street-style, celebrity events, runway, and online shopping annotated (342,182 annotations) both by crowd workers for segmentation masks and fashion experts for localized attributes, with the goal of developing and benchmarking computer vision models for comprehensive understanding of fashion. The dataset offers segmentation masks and the corresponding bounding boxes not only for the 27 main garments (e.g., jacket, coat, dress) and the accessory categories (e.g., bag, shoe), but also for 19 smaller garment objects (e.g., collar, sleeve, pocket, zipper, embroidery. These small objects are usually omitted in other fashion datasets, but could be valuable for real-world applications (e.g., searching for a specific collar shape during online-shopping).
The dataset's annotations are available under Creative Commons Attribution 4.0 license and therefore can be used for commercial purposes.

Categories:

- **shirt, blouse**
- **top, t-shirt, sweatshirt**
- **sweater**
- **cardigan**
- **jacket**
- **vest**
- **pants**
- **shorts**
- **skirt**
- **coat**
- **dress**
- **jumpsuit**
- **cape**
- **glasses**
- **hat**
- **headband, head covering, hair accessory**
- **tie**
- **glove**

- **watch**
- **belt**
- **leg warmer**
- **tights, stockings**
- **sock**
- **shoe**
- **bag, wallet**
- **scarf**
- **umbrella**
- **hood**
- **collar**
- **lapel**
- **epaulette**
- **sleeve**
- **pocket**
- **neckline**
- **buckle**
- **zipper**
- **applique**
- **bead**
- **bow**
- **flower**
- **fringe**
- **ribbon**
- **rivet**
- **ruffle**
- **sequin**
- **tasse**



Example annotation:

{

"image_id": 17370,

"category_id": 31,

"attribute_ids": [160,205],

"segmentation": [

[129,537,129,541,120,545,119,542],

[128,293,128,431,129,503,130,525,115,533,114,524,92,523]

],

"bbox": [75.0,281.0,55.0,264.0],

"area": 7695,

"iscrowd": 0,

"id": 10

},

Example image:

Fashionpedia remapped

To account for the huge number of classes in the fashionpedia dataset, we decided to merge some of these classes together. Thus, we have created the *fashionpedia-remapped* dataset. This dataset comprises of 18 classes:

- **top, t-shirt, sweatshirt** – comprises of classes: *top, t-shirt, sweatshirt* and *shirt, blouse*.
- **outerwear** – consists of classes: *sweater, cardigan, jacket, vest, coat, cape,*
- **pants** – consists of classes: *leg warmer, tights, stockings, shorts, pants.*
- **dress** – consists of classes: *jumpsuit, dress, skirt.*

- **glasses** – consists of classes: *glasses*.
- **hat** – consists of classes: *headband, head covering, hair accessory* and *hat*.
- **scarf, tie** – consists of classes *scarf* and *tie*.
- **glove** – consists of classes: *glove*.
- **watch** – consists of classes: *watch*.
- **belt** – consists of classes: *belt*.
- **shoe** – consists of classes: *shoe* and *sock*.
- **accessories** – consists of classes: *bag, wallet, umbrella, buckle, zipper, bead, bow, flower, fringe, ribbon, rivet, sequin, tassel, appalique.*
- **hood** – consists of classes: *hood*.
- **neckline** – consists of classes: *lapel, collar, neckline*
- **pocket** – consists of classes: *pocket*.
- **epaulette** – consists of classes: *epaulette*.
- **applique** – consists of classes: *applique*.
- **sleeve** – consists of classes: *ruffle, sleeve.*

The images as well as train-val split is the same as in the original fashionpedia dataset. The dataset is available on HuggingFace (link).

3. Clothing detection models review

ModaNet competition

The ModaNet dataset was released alongside a competition to build best models on this dataset. However, the link to the competition on the ModaNet GitHub repository seems to suffer from the so-called “link-rot”. There exists only one submission, with no contact information, no code, no weights of the model, and just some scores. As such, there is no model we can use here.

Clothing-Detection

The Clothing-Detection model looked promising and indeed code used to create the model is available, but the link to the model “weights” seems to only contain the configuration information of the model.

Clothing Detection YOLO

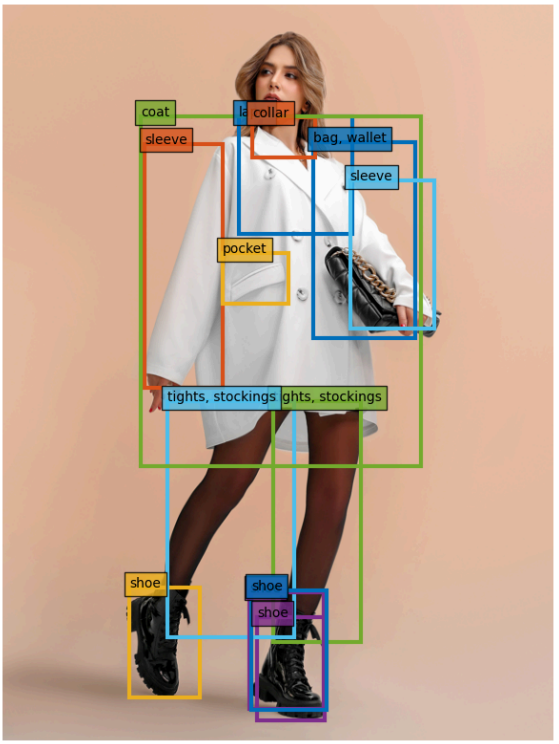
Due to the above failures to find a suitable model we decided to search for a new solution, eventually finding the Clothing_Detection_YOLO. The software took some effort to set-up, however eventually we were able to run inference on the model (Examples below). The model however only supports fairly general classifications, like outerwear or pants, without going into detail of what kind of clothing it is. We were not yet able to set-up fine-tuning for the model.


outer conf: 0.997
pants conf: 0.989
boots conf: 0.518

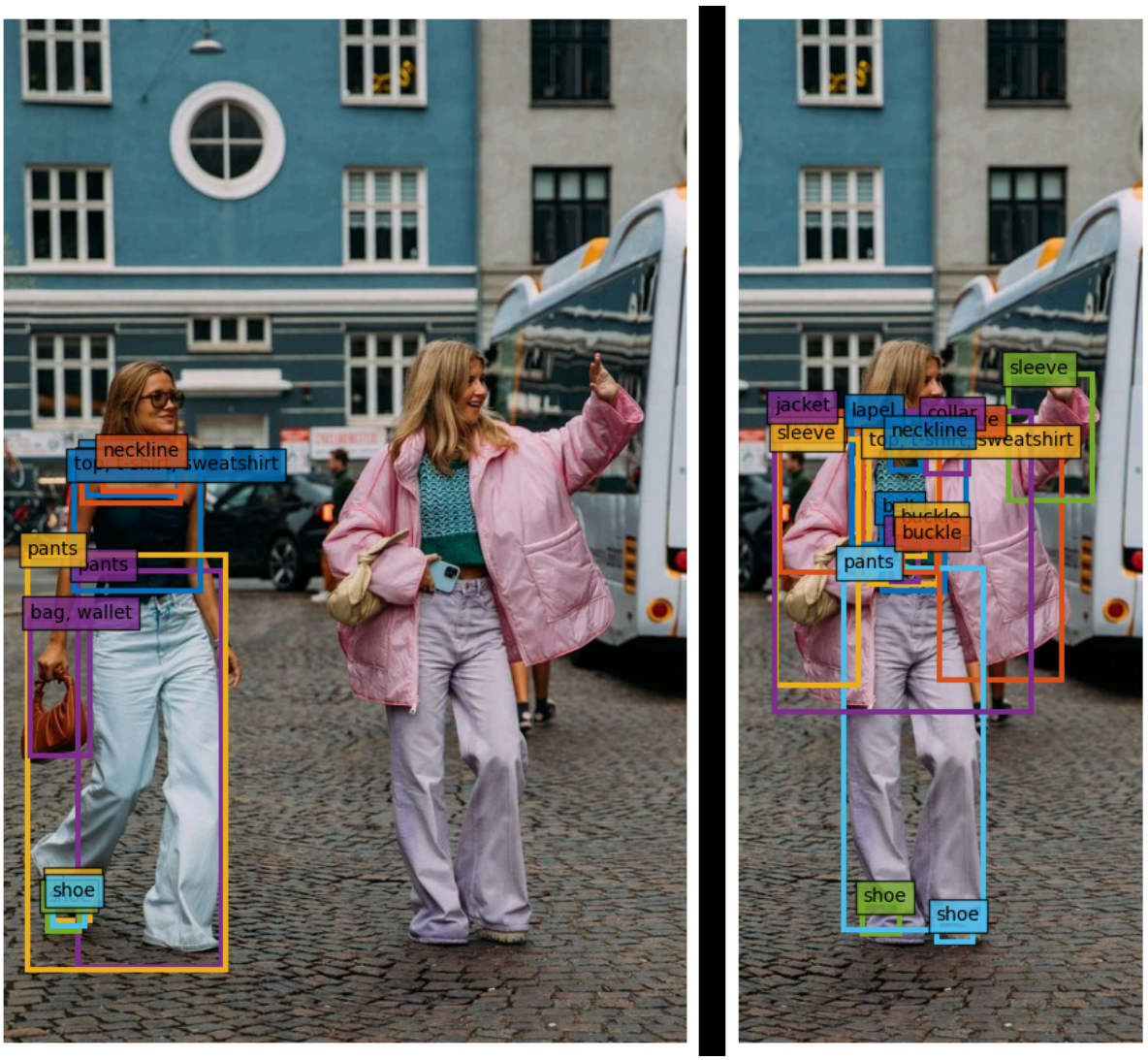

outer conf: 0.952

YOLOS-Small (ft. Fashionpedia)

The model was obtained by fine-tuning the YOLOS-Small model, that was originally trained on the COCO object detection dataset, for 1 epoch on the Fashionpedia training dataset. The model's weights are available at the HuggingFace platform (https://huggingface.co/valentinafeve/yolos-fashionpedia) and are easily accessible for inference (https://github.com/valentinafeve/fine_tunning_YOLOS_for_fashion). The model produces reasonable results for images of a single person and detects small objects such as buttons, sleeves or pockets.



However, the model appears to only detect clothes of a single person even when provided an image of a pair or a group of people. The next image presents results of model inference on an image with 2 people (left), and on a cropped image with only the rightmost person who wasn't detected in the previous scenario (right). In both cases, the model provides bounding boxes for the main clothes (jacket, pants, sweatshirt) and pays attention to smaller objects (bag, sleeve, buckle).

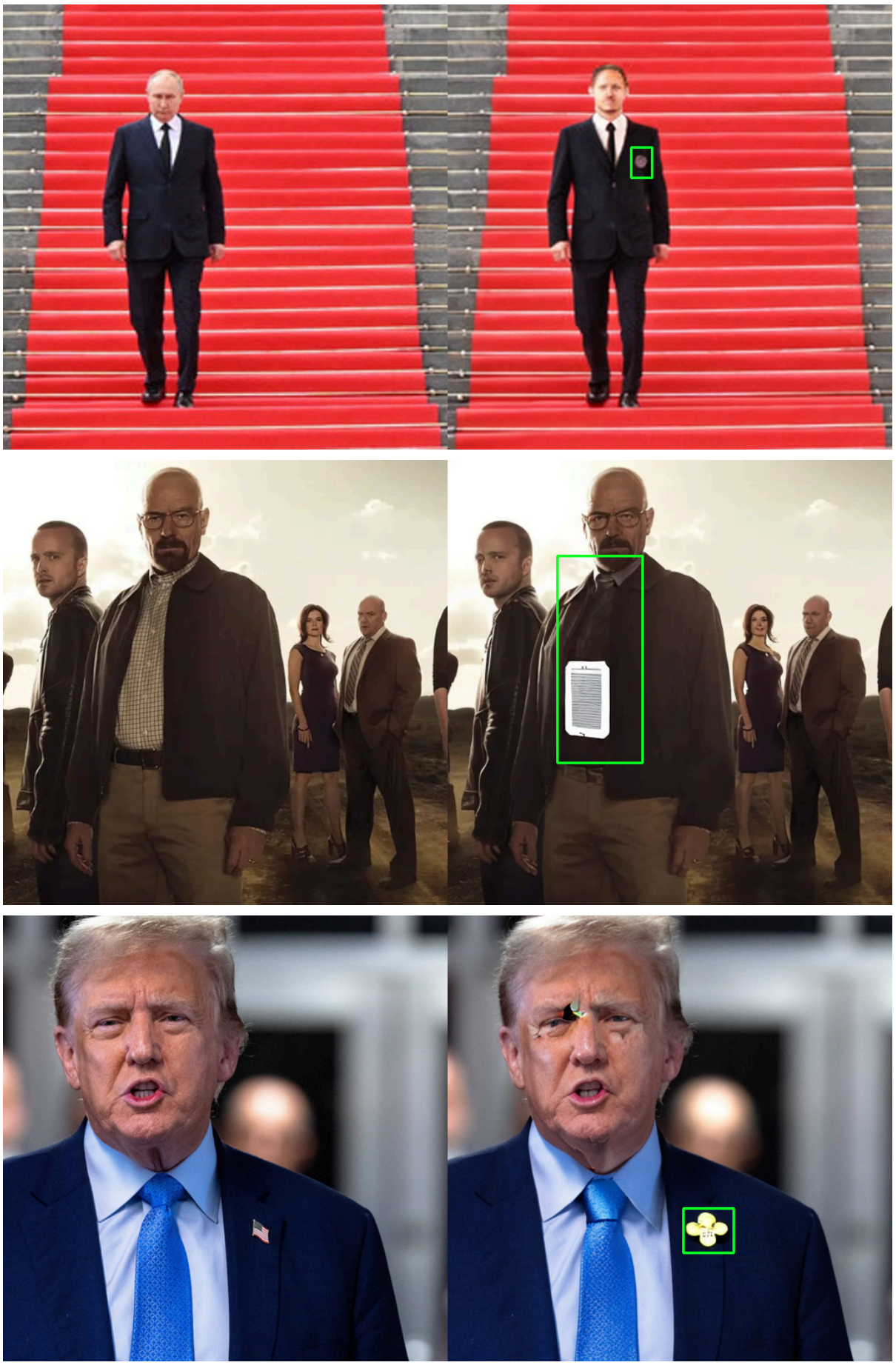


4. Other models

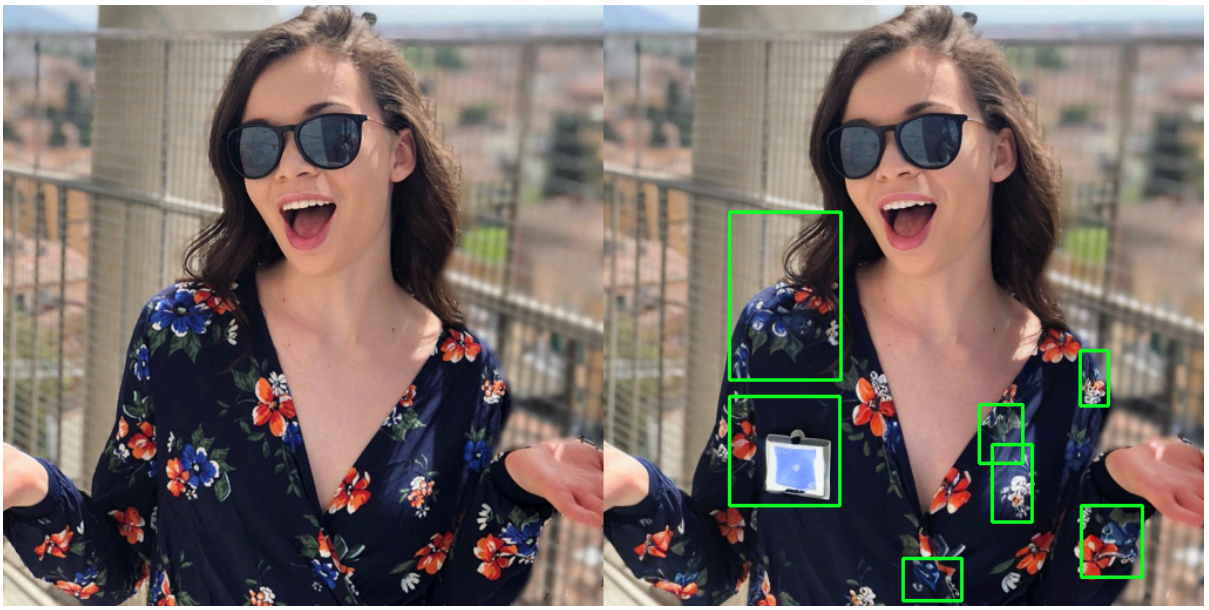
Generative AI dataset augmentation

We tried using the LEDITS++ model for automatic ID badges insertion into photos from our datasets. The model alters the given image in a way specified via a text command (e.g. “ID badge on chest”).
We tested the model on multiple images, some belonging to our datasets, some taken from the internet. The results may be considered as “mixed”. In most cases, the model succeeds in adding an ID badge (or some object vaguely resembling an ID badge or a police badge), however, it also alters the original image itself (especially in the face area of the people depicted). In some cases, however, the model fails to add a badge altogether.
Nevertheless, we then developed a method of automatically drawing a bounding box around the newly inserted batch. This method relies on comparing the images before and after model modifications and choosing only the modifications that were located

near the chest area of the person depicted. The chest area was detected using the HAAR cascade algorithm (Viola and Jones #)
Some examples of altered images with detected badges:



As we can see, even if the model and algorithm work in the intended way, the original image more often than not is being altered in a non-negligable way.
These changes often lead to many false positives, which are rather hard to detect:



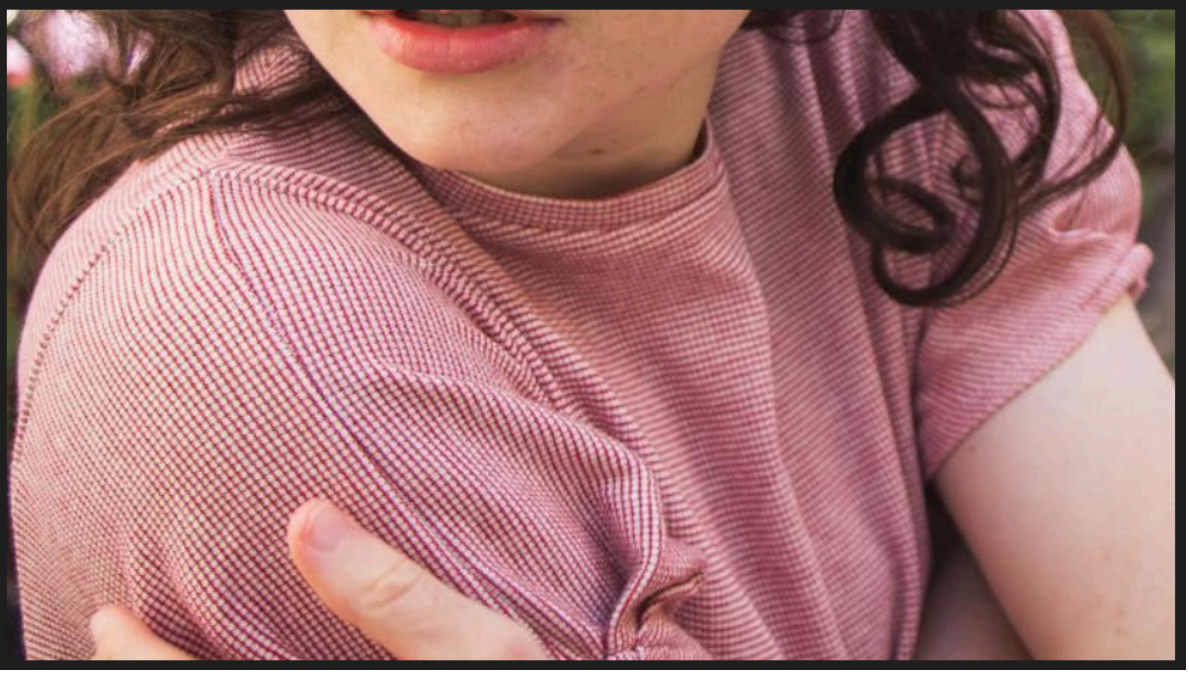
In the end, we have decided not to follow this idea further, as the generating model was deemed too unreliable. We believe, however, that we succeeded in showing the potential of such an approach, which, with perhaps a better model and meticulous supervision, could yield very promising results.

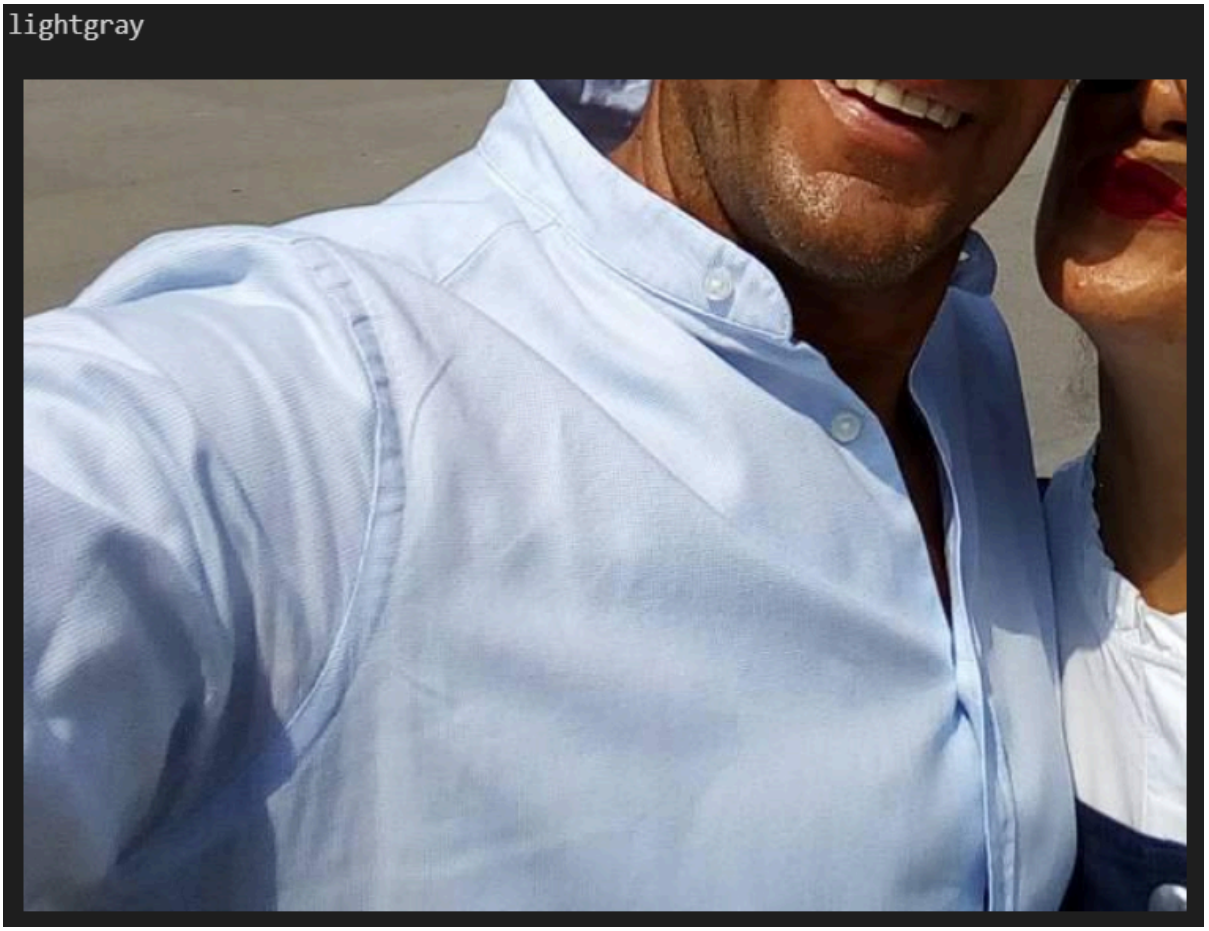
Color detection

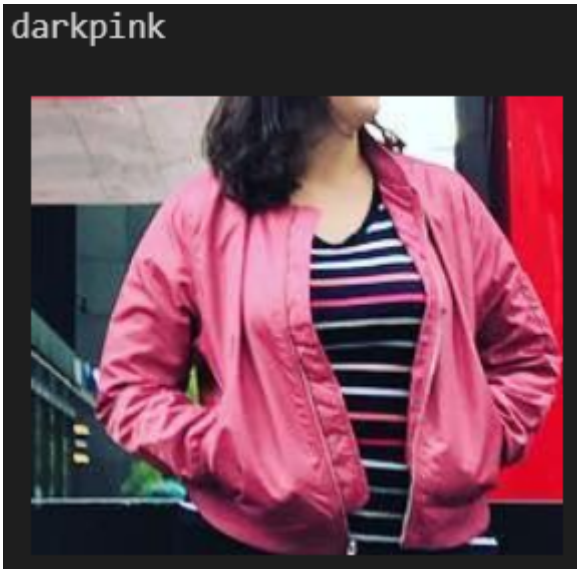
We created a separate Python module dedicated to the task of detecting a given piece of clothing's color. It has been implemented in the form of a Python function named *get_cloth_color*. It takes an image and a bounding box containing the detected item of clothing as an input. The bounding box should be of the form (x1, y1, x2, y2) (so it should contain the absolute values of top-left and bottom-right corners of the bounding box). The possible colors that can be detected are stored in a form of a pre-defined dictionary containing color names and their corresponding RGB values.
The input image is then cropped to the size of the bounding box. The module then uses the *rembg* Python package to remove the background from the cropped image (this is done only if the background covers less than a certain percentage of the cropped image, the percentage given as a parameter *BACKGROUND_PERCENTAGE_THRESHOLD*). We then use the *haishoku* Python package to find the dominant color (defined by its RGB value) in the resulting image. The function then returns the name of the color from the color dictionary that is the closest to the detected dominant color in terms of the RGB value.
Some examples of the module in action (the detected color names have been printed above the image):

black

darkpink







As we can see, the module is largely successful in detecting appropriate colors. The performance may be further enhanced by providing additional, more fine-grained color names to the COLOR_DICT, provided with module's implementation.

Per-person segmentation

To enhance the overall performance of the clothing detection system, we introduce a separate object detection model for extracting human bounding boxes. The additional

detection model enables a two-stage pipeline: the first stage involves person detection, and the second stage conducts clothing detection on the cropped person bounding boxes.
We selected the YOLOv7 architecture as the human detector, utilizing a COCO-pretrained checkpoint, configured to detect only the human category. Based on evaluation with test images, we manually adjusted the detector's post-processing parameters to a classification confidence threshold of 80% and a Non-Maximum Suppression (NMS) threshold of 45%. The post-processing algorithm eliminates predictions with a classification confidence below 80% and sorts the remaining bounding boxes in descending order of confidence. Starting from the highest confidence prediction, the algorithm calculates the Intersection over Union (IoU) with other boxes and removes those whose IoU exceeds 45%. This method effectively eliminates duplicate predictions corresponding to the same person. Finally, the model outputs a single bounding box per human in the input image.

5. Fine tuning YOLOS

YOLOS-Tiny

We performed two fine-tuning experiments on the *original* and *remapped* versions of Fashionpedia dataset that are split into 45,600 training and 1,160 validation images. Both runs use a specific YOLOS-Tiny checkpoint as the starting point. This model was initially pre-trained for 300 epochs on the ImageNet-1k dataset and subsequently fine-tuned for 300 epochs on the COCO2017 dataset for the object detection task.
The fine-tuning process employed Adam optimizer with the learning rate of 0.000025 and the weight decay of 0.0001. The final loss function was composed of the standard cross-entropy loss for class predictions and a linear combination of L1 loss and generalized IoU loss for bounding box predictions. We performed 5 complete fine-tuning epochs for each version of the 45,600-image training dataset. Each epoch required approximately 3 hours on an 8GB GPU.
The resulting fine-tuned checkpoints for both the original and remapped versions of Fashionpedia dataset were subsequently uploaded to HuggingFace.

6. Pipeline and packaging

The software is packaged as a Python script which performs a full inference pipeline from loading images to saving results, meaning all necessary pre-processing and post-processing steps are included. The pipeline works by chaining machine learning models and trimming operations. First, the pipeline finds all images in the provided input directory. For each image the pipeline detects persons in the image, then trims the image to the detected bounding boxes of the person, and runs a clothing detection model on the resulting trimmed image. Outputs of the clothing detection model are used to trim the image again, into individual clothing pieces, which are then analyzed for their color. This allows us to match the pieces of clothing with the person wearing them, as well as focus the color detection model to one piece of clothing.
Below is pseudocode which gives an overview of the pipeline:

```
for image in INPUT_DIR:
  persons = detect_persons(image)
  for person in persons:
    person_image = trim_image(image, person)
    clothes = detect_clothes(person_image)
    for clothing_item in clothes:
      clothing_item_image = trim_image(image, clothing_item)
      item_color = detect_color(clothing_item_image)
```

The script outputs a JSON-formatted output with the following hierarchical structure:

- Filename
- Person
- Clothing item

At the root of the JSON is a mapping from image filenames to a list of persons. Each person has two properties: “human_bbox” and “elements”. All *_bbox properties are bounding boxes of the object, represented as a list of 4 numbers: x, y coordinates of top left corner, and x, y coordinates of bottom right corner, in that order. The “elements” property of a person contains a list of clothing items. Each clothing item contains 5 properties: “object_name” - the detected class, “probability” - confidence in this prediction, “relative_bbox” - bounding box relative to the “human_bbox” property of the parent person, “absolute_bbox” - bounding box relative to the image, and “color” - the detected color of the clothing piece.
Below is an example result, which has been trimmed to one image, one person and one piece of clothing:

```
{
  "image.webp": [
    {
      "human_bbox": [291, 73, 767, 1086],
      "elements": [
        {
          "object_name": "shoe",
          "probability": 0.9565234184265137,
          "relative_bbox": [7.37, 879.02, 90.03, 997.93],
          "absolute_bbox": [298.37, 952.02, 381.03, 1070.93],
          "color": "black"
        }
      ]
    }
  ]
}
```

The script has two optional arguments: output-file and clothing-detection-model. If output-file is provided, then the script will save its results to the specified JSON file, otherwise it will print the “prettified” version to the standard output. The clothing-detection-model argument allows an override of the clothing detection model, which can be the Huggingface identifier

of any compatible model, for example our alternative models, described in the “Fine tuning YOLOS” section.

Dependencies of the script are split between requirements.in (direct dependencies, no set version numbers) and requirements.txt (all dependencies). The script has been tested with CPython 3.11.8 on Debian 12.5. The hardware used is a personal workstation with an 11th generation i5 Intel processor, a Nvidia 3060Ti GPU and 32GB or RAM.
The main file for script running is *run.py.* Example usage:

```
python run.py <folder with images> -o <output folder> -m <model name>
```

Arguments `<output file>` and `<model name>` are optional. When not passed, the script outputs results to the console. Model name is usually a HuggingFace path (e.g. *itesl/yolos-tiny-fashionpedia-remapped*)

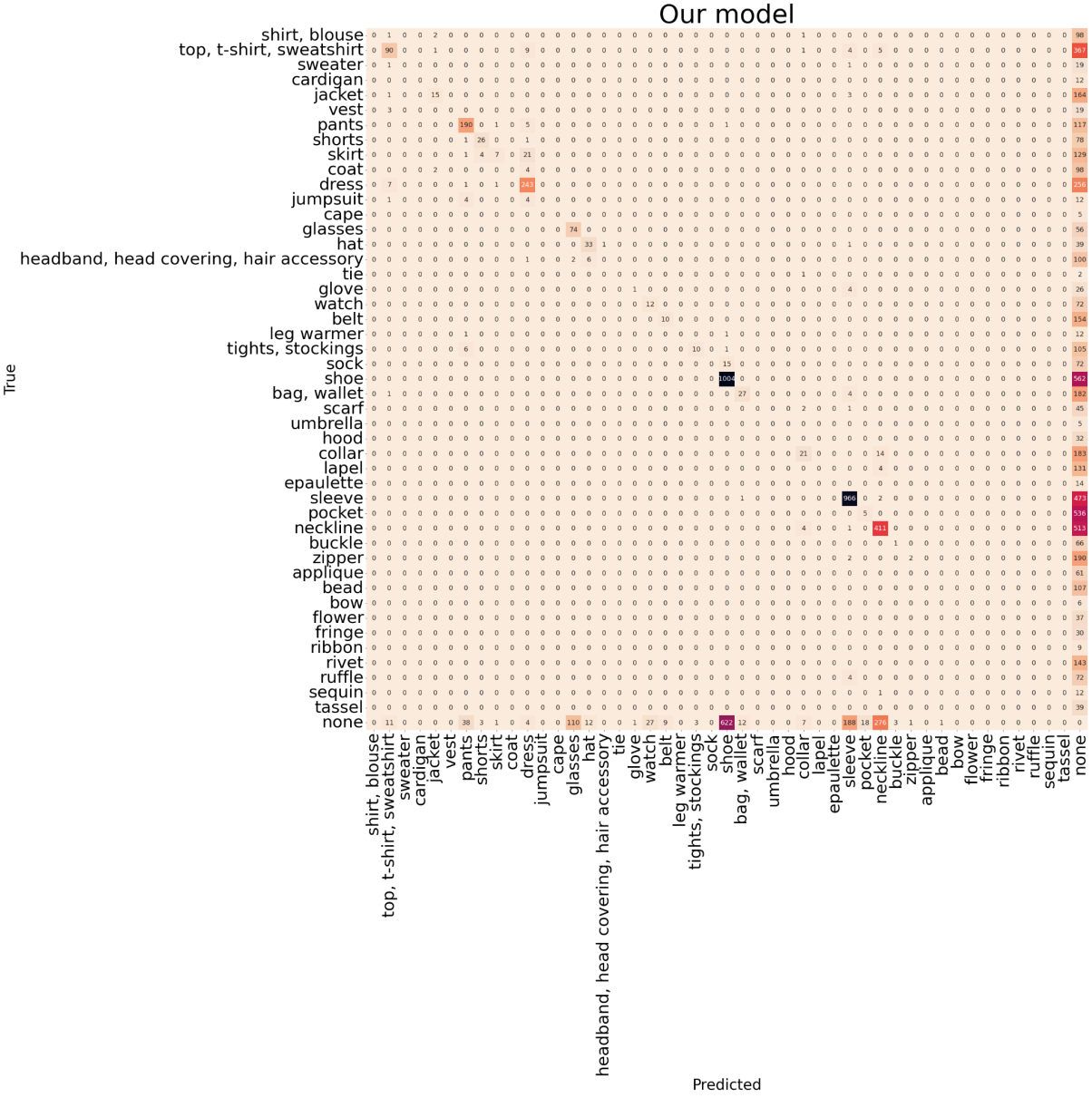
7. Results

In measuring our models’ performance, we focused on checking the precision (how accurate the model is in predicting a given item of clothing) and the recall (what fraction of the clothing’s instances the model is able to retrieve).
We focused on checking how many false positives (objects incorrectly predicted), false negatives (objects not predicted, but present), and true positives (objects correctly predicted) have been detected for each of the dataset’s classes. For testing, we chose the above-described dataset *Fashionpedia*, as it contained the highest number of high-quality data among all the datasets that we have collected for this project.
The statistics were calculated as follows. For each image, we have a set of ground truth bounding boxes (along with their labels), and a set of predicted bounding boxes. For each pair of bounding boxes (ground-truth prediction) we compute an *Intersection over Union* metric (*IOU*). This is essentially the area of intersection of the two bounding boxes divided by the area of their union. Having computed IOU scores for each pairing, we use the *Hungarian algorithm* to determine the optimal pairings. We then compare the labels of the paired boxes. Moreover, if the IOU score for a given pairing is below a certain threshold (0.4 in our case, as set by examining some of our models’ predictions), then we count such observation as a false positive, regardless of its label.
We will be comparing 3 models. The first one was found on the *HuggingFace* website and is the YOLOS-small vision transformer fine-tuned on the *fashionpedia* dataset. The second one is the YOLOS-tiny architecture, which we fine-tuned on the *fashionpedia* dataset. The third and final model is also the YOLOS-tiny architecture, which has been fine-tuned by us on the remapped version of the *fashionpedia* dataset.
After some initial setbacks (connected to the number of false positives detected), the final scores (averaged with weights corresponding to the number of elements in each class) for each model are:

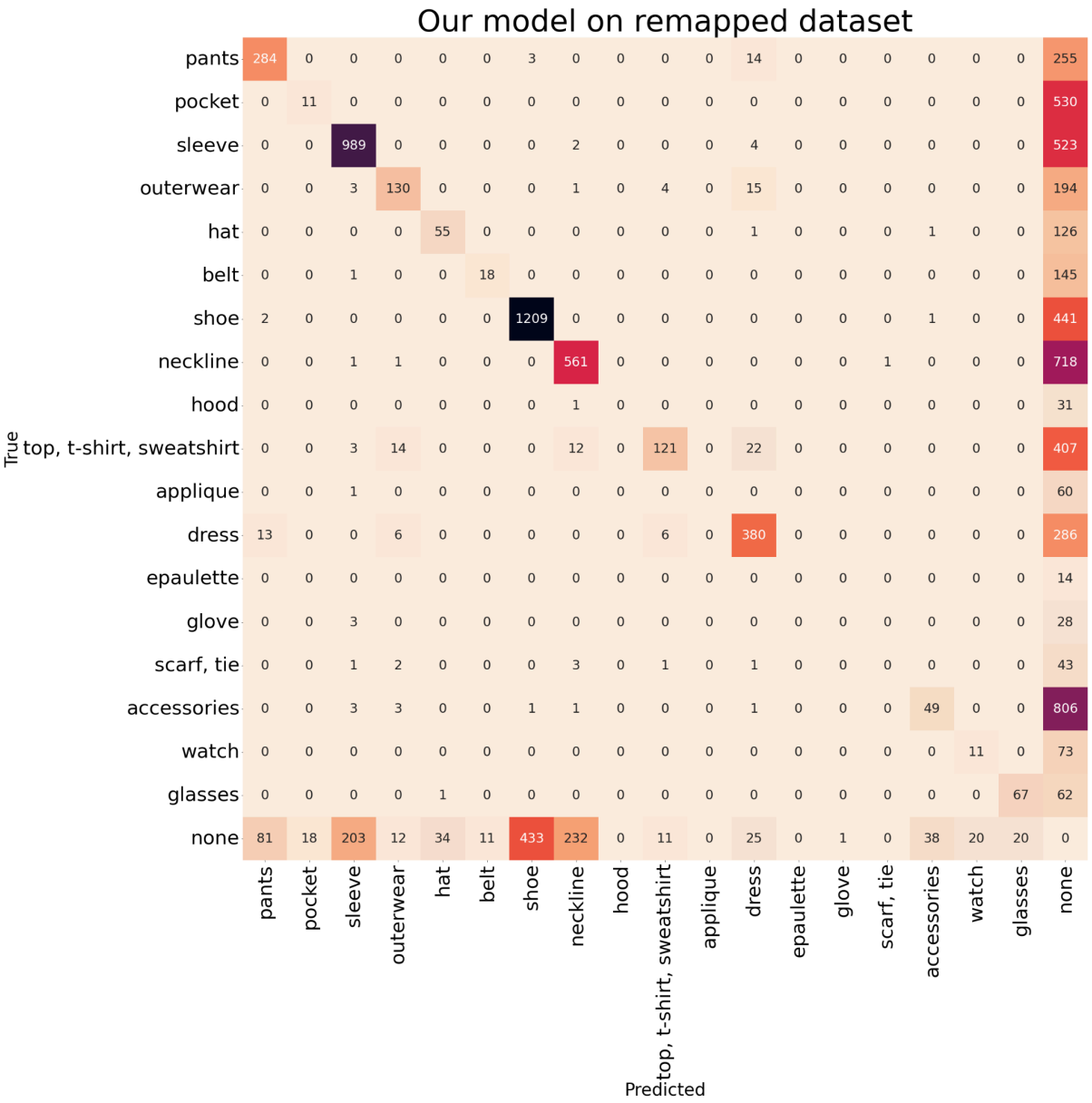
RESULTS ON “RAW” MODELS

	Our model (full fashionpedia)	Baseline (full fashionpedia)	Our model (fashionpedia remapped)
Average precision	0.56	0.66	0.7
Average recall	0.36	0.41	0.44

Those are the scores on “raw” models, meaning that we have not used our pipeline. As we can see, the scores are quite promising when we take into account the number of classes in the fashionpedia dataset (46) and the fact that both our model and the “baseline” model are relatively small in size (6.49M and 30.7M params, respectively) when compared to some other state-of-the-art models (65M for the “standard” YOLO model). These discrepancies in size also are the reason that our model (YOLOS-tiny architecture) perform worse than the baseline (YOLOS-small). This, however, comes as a trade-off, as our model is capable of much faster inference (3s on average when compared to 9s for “baseline”, tested on Intel Core i5 CPU).
It is also worth noting that there is a large imbalance between different classes (as described in section Fashionpedia), and as such, the difference in scores between some more “popular” classes and some less numerous ones can be substantial. As an example, class “dress” has 0.83 precision and 0.48 accuracy scores, while class “cardigan” has not been detected at all (and there are only 12 members of that class in the dataset). The situation can be further explored on the confusion matrix:



As we can see, most predictions fall on the diagonal of the matrix. There are, however, a lot of cases of false negatives (the last column) and false positives (the last row), which may be the result of the insufficient model size. Especially the false positives are usually the result of bounding boxes not aligning properly with ground truths. What is especially notable, however, is the low number of false positives between classes – which means that even our tiny architecture is rather accurate. The same conclusions can be drawn from the remapped dataset:

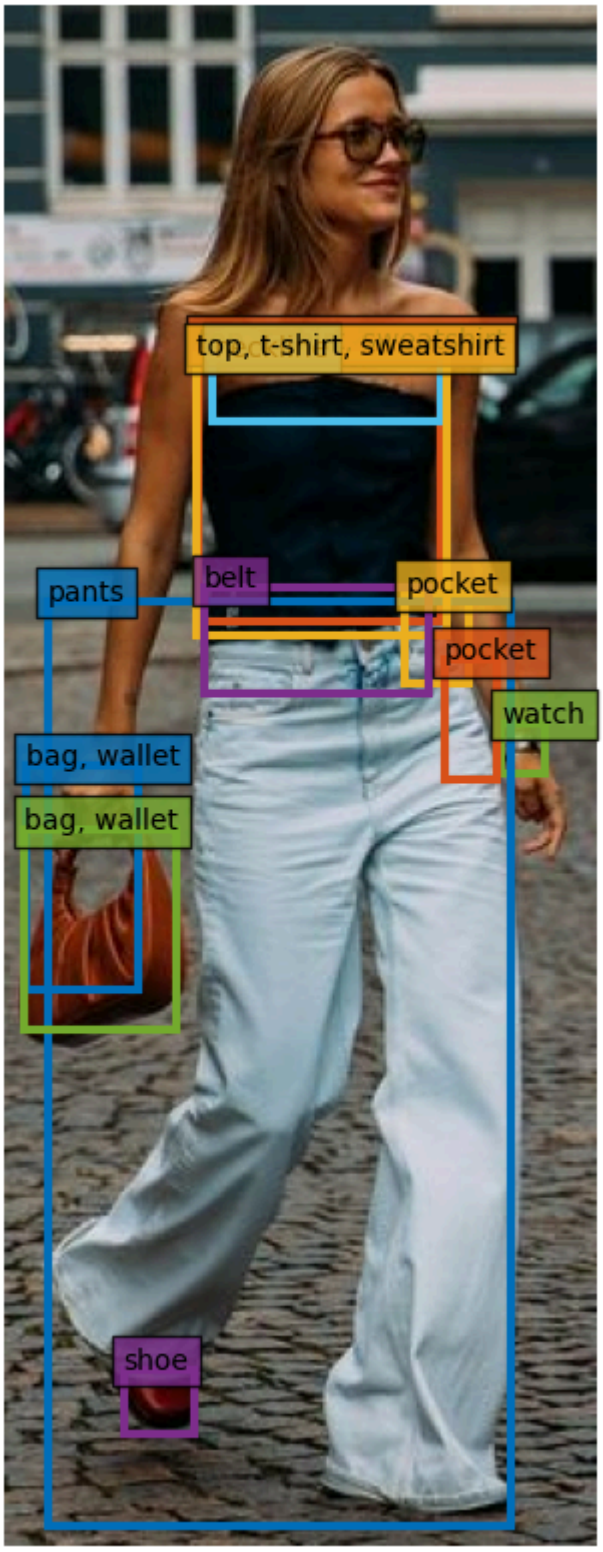


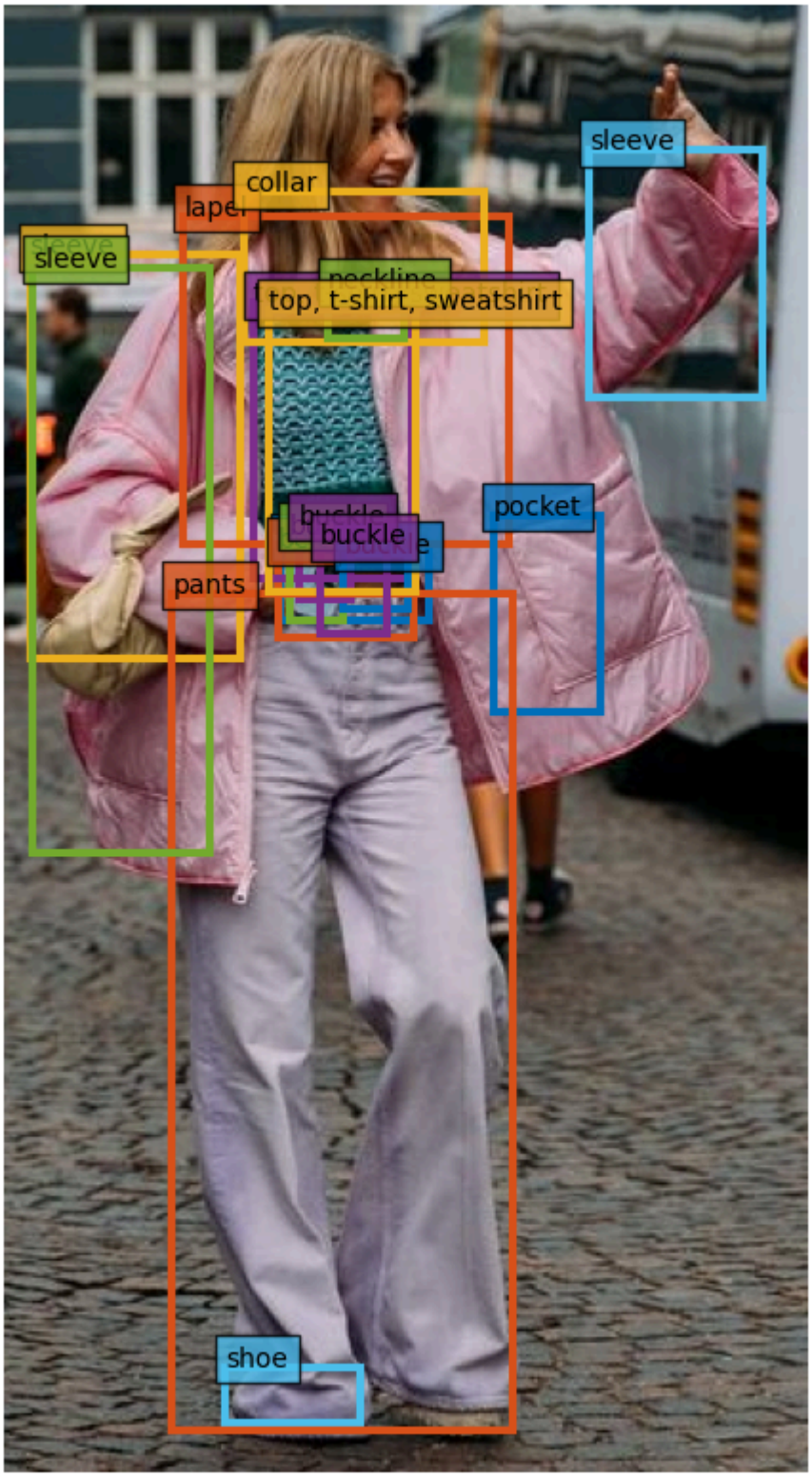
Here we can see even more clearly that most elements align on the diagonal. It is worth noting that the “accessories” class here consists of mostly not-detected instances (the last column). That is because this class encompasses items far less common with “typical” outfits, such as beads, bows or flowers.
We can also compare these results with the ones obtained by using our pipeline (that is, first detecting people, and then detecting items of clothing). Here are the results with our pipeline:

RESULTS WITH OUR PIPELINE			
	Our model (full fashionpedia)	**Baseline (full fashionpedia)**	**Our model (fashionpedia remapped)**
Average precision	0.55	0.66	0.67
Average recall	0.35	0.38	0.41

As we can see, the results are slightly worse when comparing to the “raw” models. This, however, we attribute to the dataset itself. Fashionpedia consists of images containing at most one person. This especially means that models trained on it are incapable of detecting clothes on more than one person at time. As a result, while models trained solely on fashionpedia may achieve better results without our pipeline, when applied to some new images (especially those containing more people), their performance quickly deteriorates. Here we present an example:

As we can see, when applied by itself, the model trained on fashionpedia only detects clothes on a single person. However, using our pipeline, we are able to obtain separate results:


top, t-shirt, sweatshirt
pants
belt
pocket
pocket
watch
bag, wallet
bag, wallet
shoe



This, we believe, proves the validity of our approach.

8. Conclusions

We performed a review of publicly available datasets and solutions for the problem of clothing detection. While there are many choices in datasets, the detail of labels leaves much to be desired. Additionally, we found several labeling mistakes in certain datasets. Clothing detection models are scarcely available, and many of them are hobby projects as opposed to production-ready software. We believe that many models might be kept behind closed-doors, due to their value for e-commerce applications. Therefore, we also trained our own models, which achieved reasonable results. The weights and code used to train these models is made available to make it easier to further fine-tune the models, or use a larger YOLOS architecture. We then combined select open source software and our own trained models into a complete pipeline for analyzing attire of people in images. The pipeline analyzes each person separately, detects their clothing pieces, and further analyses the color of the clothing pieces. Our experiments with generative AI for dataset augmentation were ultimately unsuccessful due to difficulty in running the software locally, as well as the difficulty with “focusing” the image editor. Despite this, we believe that the idea might still work for enhancing datasets with uncommon objects, but the results of the generative AI model need a human-in-the-loop to ensure that the model does not modify large chunks of the image. Despite the many issues we encountered, we hope the work we performed will be of use.

9. Code availability

Source code is available in this Github repository.