Progress Report

By Wayne, Young Joo, and Kelson

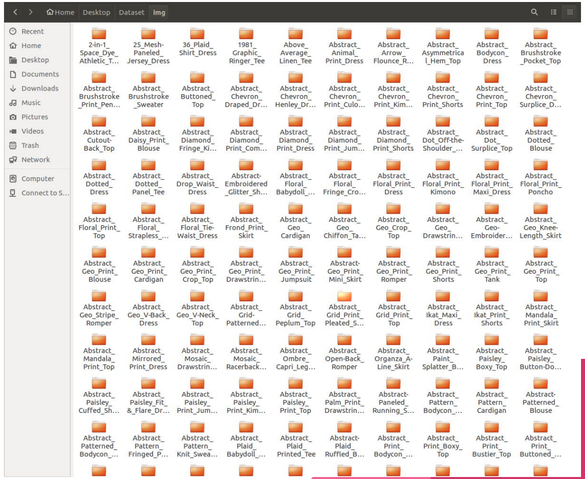
Outline

- Introduction to Deep Fashion
  - Introduction
  - Goal
- Object Detection API
  - Introduction
  - Young Joo
  - Kelson
  - Wayne
  - Problems & Conclusion
- Bad Data
  - Data cleaning
- Future Plan

Deep Fashion - Introduction

- A large scale dataset
  - Chinese University of Hong Kong(CHUK)
  - 289,222 clothes
  - 50 broad & 5612 specific categories
  - Bounding box & clothing type/label annotations

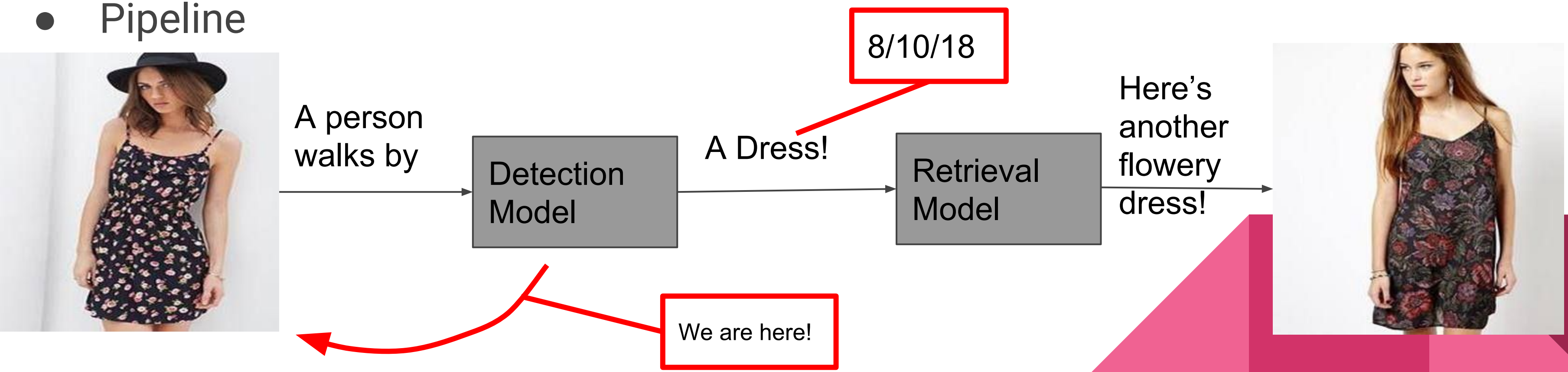
Flower & Dot Cami Dress



2-in-1_Space_Dye_Athletic_T…
25_Mesh-Paneled_Jersey_Dress
36_Plaid_Shirt_Dress
1981_Graphic_Ringer_Tee
Above_Average_Linen_Tee
Abstract_Animal_Print_Dress
Abstract_Arrow_Flounce_R…
Abstract_Asymmetrical_Hem_Top
Abstract_Bodycon_Dress
Abstract_Brushstroke_Pocket_Top
Abstract_Brushstroke_Print_Pen…
Abstract_Brushstroke_Sweater
Abstract_Buttoned_Top
Abstract_Chevron_Draped_Dr…
Abstract_Chevron_Henley_Dr…
Abstract_Chevron_Print_Culo…
Abstract_Chevron_Print_Kim…
Abstract_Chevron_Print_Shorts
Abstract_Chevron_Print_Top
Abstract_Chevron_Surplice_D…
Abstract_Cutout-Back_Top
Abstract_Daisy_Print_Blouse
Abstract_Diamond_Fringe_Ki…
Abstract_Diamond_Print_Com…
Abstract_Diamond_Print_Dress
Abstract_Diamond_Print_Jum…
Abstract_Diamond_Print_Shorts
Abstract_Dot_Off-the-Shoulder_…
Abstract_Dot_Surplice_Top
Abstract_Dotted_Blouse
Abstract_Dotted_Dress
Abstract_Dotted_Panel_Tee
Abstract_Drop_Waist_Dress
Abstract-Embroidered_Glitter_Sh…
Abstract_Floral_Babydoll_…
Abstract_Floral_Fringe_Cro…
Abstract_Floral_Print_Dress
Abstract_Floral_Print_Kimono
Abstract_Floral_Print_Maxi_Dress
Abstract_Floral_Print_Poncho
Abstract_Floral_Print_Top
Abstract_Floral_Strapless_…
Abstract_Floral_Tie-Waist_Dress
Abstract_Frond_Print_Skirt
Abstract_Geo_Cardigan
Abstract_Geo_Chiffon_Ta…
Abstract_Geo_Crop_Top
Abstract_Geo_Drawstrin…
Abstract_Geo-Embroider…
Abstract_Geo_Knee-Length_Skirt
Abstract_Geo_Print_Blouse
Abstract_Geo_Print_Cardigan
Abstract_Geo_Print_Crop_Top
Abstract_Geo_Print_Drawstrin…
Abstract_Geo_Print_Jumpsuit
Abstract-Geo_Print_Mini_Skirt
Abstract_Geo_Print_Romper
Abstract_Geo_Print_Shorts
Abstract_Geo_Print_Tank
Abstract_Geo_Print_Top
Abstract_Geo_Stripe_Romper
Abstract_Geo_V-Back_Dress
Abstract_Geo_V-Neck_Top
Abstract_Grid-Patterned…
Abstract_Grid_Peplum_Top
Abstract_Grid_Print_Pleated_S…
Abstract_Grid_Print_Top
Abstract_Ikat_Maxi_Dress
Abstract_Ikat_Print_Shorts
Abstract_Mandala_Print_Skirt
Abstract_Mandala_Print_Top
Abstract_Mirrored_Print_Dress
Abstract_Mosaic_Drawstrin…
Abstract_Mosaic_Racerback…
Abstract_Ombre_Capri_Leg…
Abstract_Open-Back_Romper
Abstract_Organza_A-Line_Skirt
Abstract_Paint_Splatter_B…
Abstract_Paisley_Boxy_Top
Abstract_Paisley_Button-Do…
Abstract_Paisley_Cuffed_Sh…
Abstract_Paisley_Fit_&_Flare_Dr…
Abstract_Paisley_Print_Jum…
Abstract_Paisley_Print_Kim…
Abstract_Paisley_Print_Top
Abstract_Palm_Print_Drawstrin…
Abstract-Paneled_Running_S…
Abstract_Pattern_Bodycon_…
Abstract_Pattern_Cardigan
Abstract-Patterned_Blouse
Abstract_Patterned_Bodycon_…
Abstract_Pattern_Fringed_P…
Abstract_Pattern_Knit_Swea…
Abstract_Plaid_Babydoll_…
Abstract_Plaid_Printed_Tee
Abstract-Plaid_Ruffled_B…
Abstract_Print_Bodycon_…
Abstract_Print_Boxy_Top
Abstract_Print_Bustier_Top
Abstract_Print_Buttoned_…

Deep Fashion - Goal

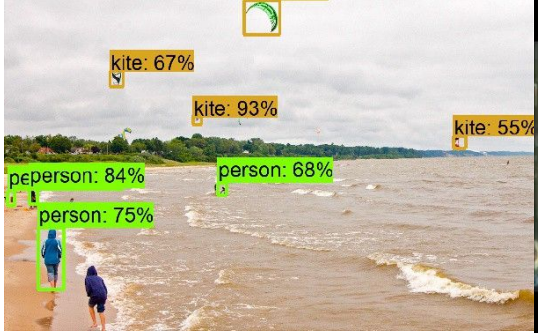
- In the far future…
  - Recommend clothing to customers that walk by
  - Based on the clothes in the data, recommend those of the similar styles
- Before August 10th
  - Detect the clothes the customer is wearing
- Pipeline

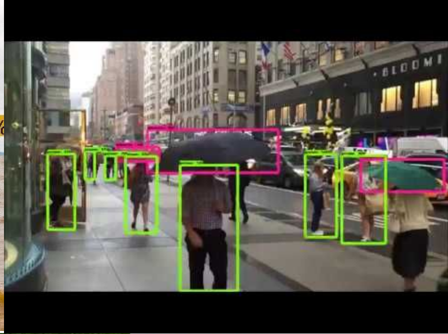


Object Detection API

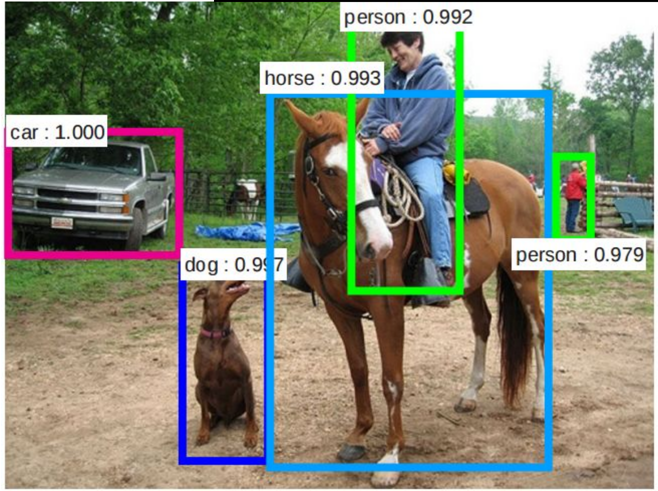
Our Models

Object Detection



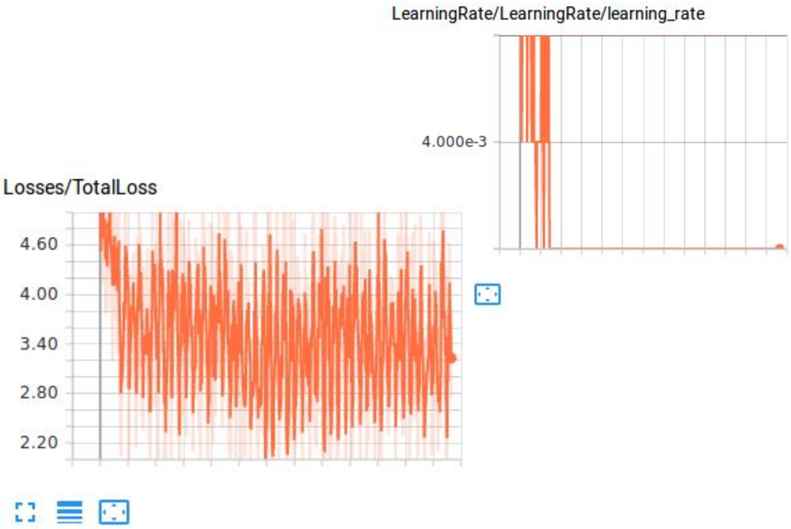


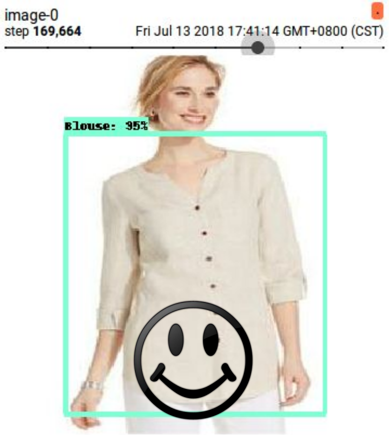
- An API provided by GOOGLE
  - Provides model zoo with pretrained ckpt
  - Pretrained on datasets including
    - COCO (objects)
    - Kitti (street view)
    - OpenImages (objects)
    - AVA (human actions e.g. listening, holding)
- How to use
  - Prepare data
  - Select proper model
  - Finetune

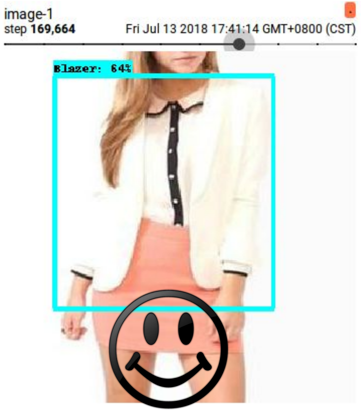


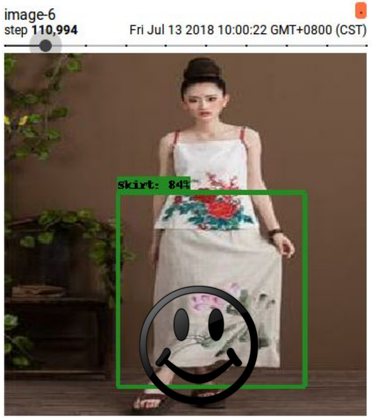
Young-Joo

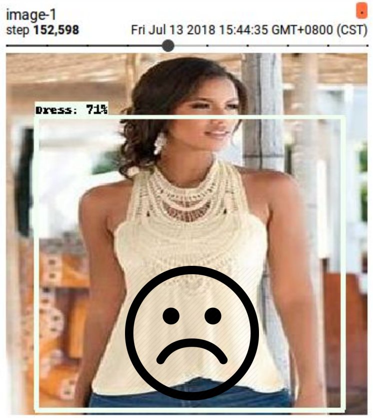
- Ssd_mobilenet_v2 (pre-trained on COCO)
- 50 categories
- Trained for 4 days
    - Instead of converging, loss fluctuated
    - Learning rate also fluctuated
- Performed ok, but far from well





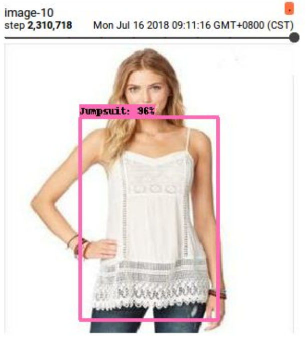


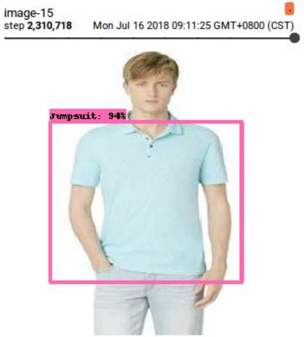


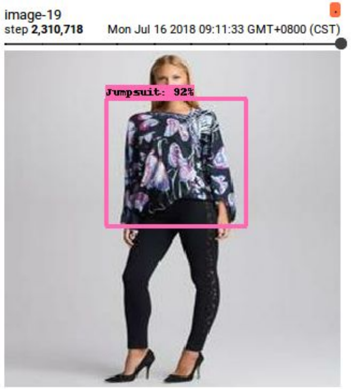


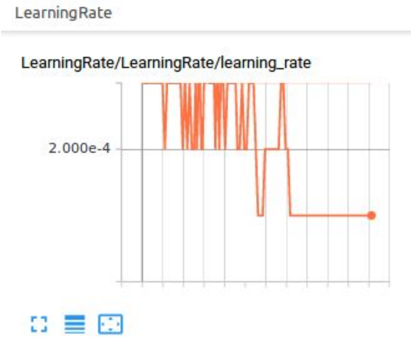
Kelson

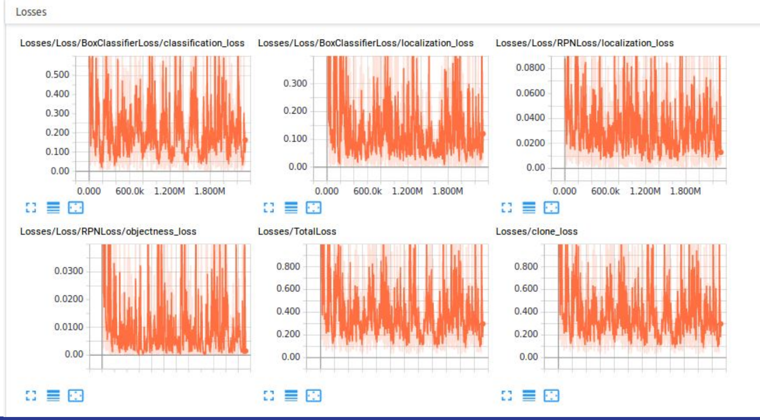
- Model: faster_rcnn_inception_v2
    - Pretrained on COCO
- Number of Categories: 50
- Training time:
    - 2d 23h 8m 0s
    - 2,318,309 steps
- Fluctuations
    - Learning Rate
    - Losses
- Poor performance
    - Everything is jumpsuit?

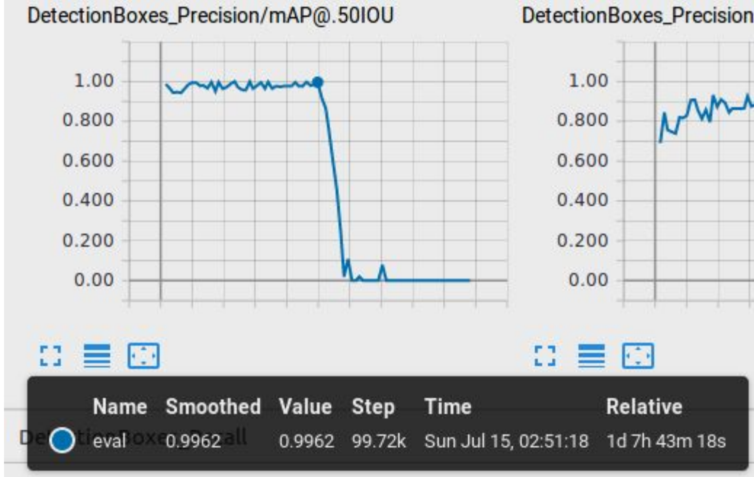












	Name	Smoothed	Value	Step	Time	Relative
●	eval	0.9962	0.9962	99.72k	Sun Jul 15, 02:51:18	1d 7h 43m 18s

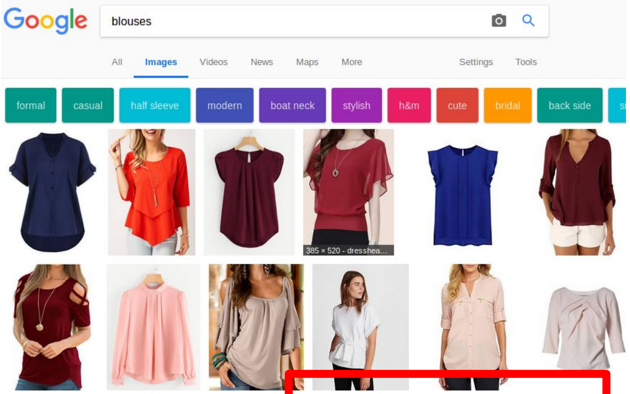
Wayne

- Model
  - Faster RCNN Inception Resnet v2 atrous
  - Pretrained on COCO
- Reasoning
  - This model has ckpts pretrained on both COCO and OpenImage
  - Two datasets are both more relevant to our dataset
  - Can see if it is the model structure or the pretraining dataset that matters
- 3 categories instead of 50
  - Trained for (196.5k steps) 2d15h39m11s
  - Model had great test mAP(0.9962) when at 99.72k steps
  - Works well with 3 categories if only there were no Overfitting
    - Too few categories compared with the complexity/capacity of the model
    - No Dropout layer
    - No Early stopping

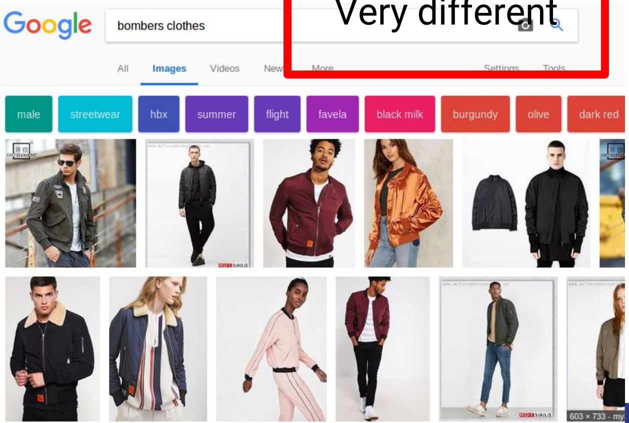
Problems We Faced

Problems, Causes, and Solutions

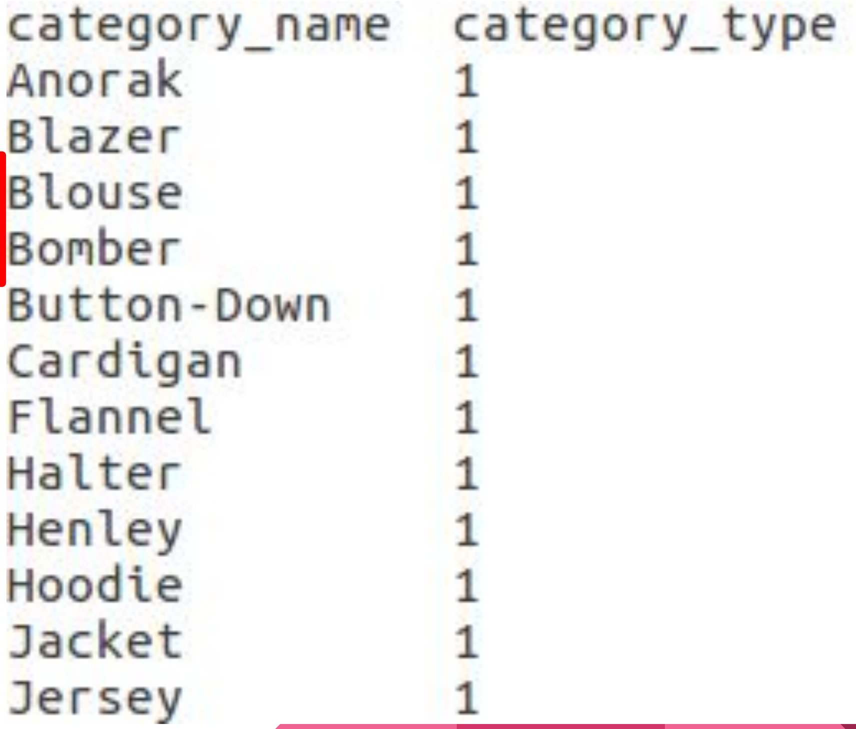
Problem 1: All blouses are bombers…?



Very different

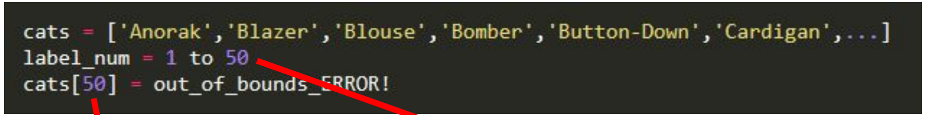


Right next to each other…?

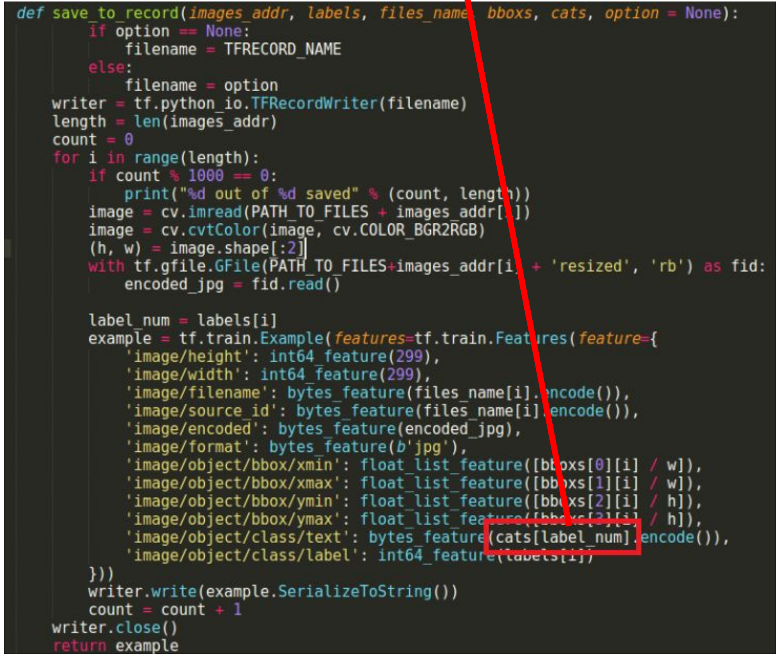


category_name	category_type
Anorak	1
Blazer	1
Blouse	1
Bomber	1
Button-Down	1
Cardigan	1
Flannel	1
Halter	1
Henley	1
Hoodie	1
Jacket	1
Jersey	1

Cause: Mismatch between labels and indices

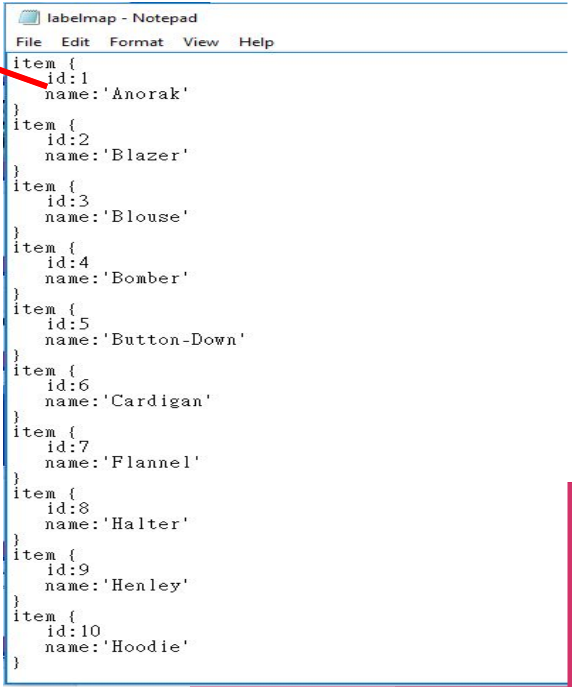


```
cats = ['Anorak','Blazer','Blouse','Bomber','Button-Down','Cardigan',...]
label_num = 1 to 50
cats[50] = out_of_bounds_ERROR!
```



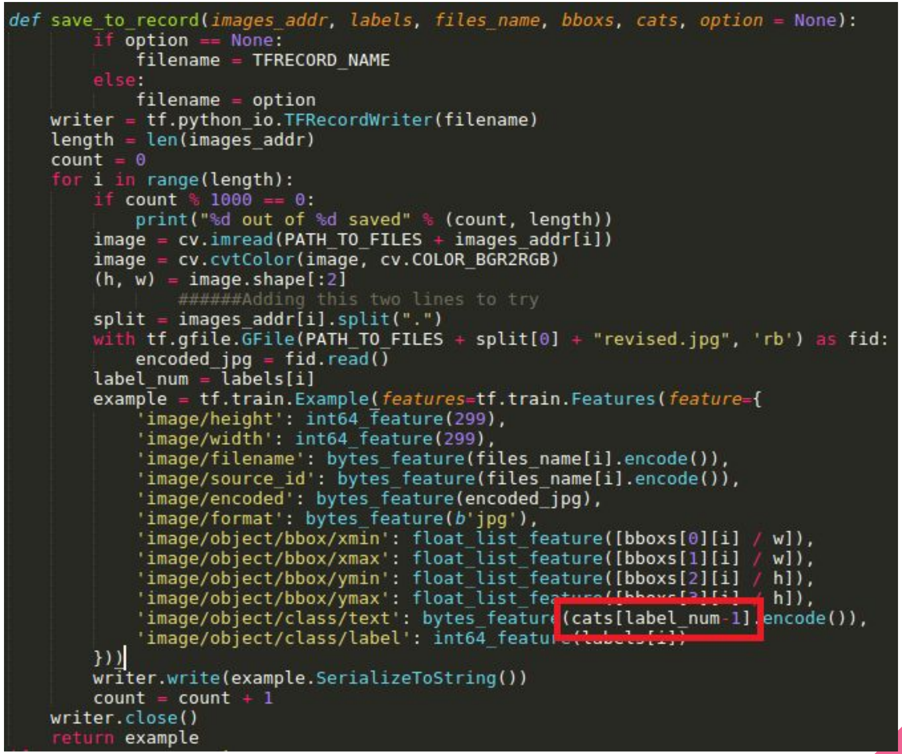
```
def save_to_record(images_addr, labels, files_name, bboxs, cats, option = None):
        if option == None:
            filename = TFRECORD_NAME
        else:
            filename = option
    writer = tf.python_io.TFRecordWriter(filename)
    length = len(images_addr)
    count = 0
    for i in range(length):
        if count % 1000 == 0:
            print("%d out of %d saved" % (count, length))
        image = cv.imread(PATH_TO_FILES + images_addr[i])
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
        (h, w) = image.shape[:2]
        with tf.gfile.GFile(PATH_TO_FILES+images_addr[i] + 'resized', 'rb') as fid:
            encoded_jpg = fid.read()

        label_num = labels[i]
        example = tf.train.Example(features=tf.train.Features(feature={
            'image/height': int64_feature(299),
            'image/width': int64_feature(299),
            'image/filename': bytes_feature(files_name[i].encode()),
            'image/source_id': bytes_feature(files_name[i].encode()),
            'image/encoded': bytes_feature(encoded_jpg),
            'image/format': bytes_feature(b'jpg'),
            'image/object/bbox/xmin': float_list_feature([bboxs[0][i] / w]),
            'image/object/bbox/xmax': float_list_feature([bboxs[1][i] / w]),
            'image/object/bbox/ymin': float_list_feature([bboxs[2][i] / h]),
            'image/object/bbox/ymax': float_list_feature([bboxs[3][i] / h]),
            'image/object/class/text': bytes_feature(cats[label_num].encode()),
            'image/object/class/label': int64_feature(labels[i])
        }))
        writer.write(example.SerializeToString())
        count = count + 1
    writer.close()
    return example
```



```
item {
   id:1
   name:'Anorak'
}
item {
   id:2
   name:'Blazer'
}
item {
   id:3
   name:'Blouse'
}
item {
   id:4
   name:'Bomber'
}
item {
   id:5
   name:'Button-Down'
}
item {
   id:6
   name:'Cardigan'
}
item {
   id:7
   name:'Flannel'
}
item {
   id:8
   name:'Halter'
}
item {
   id:9
   name:'Henley'
}
item {
   id:10
   name:'Hoodie'
}
```

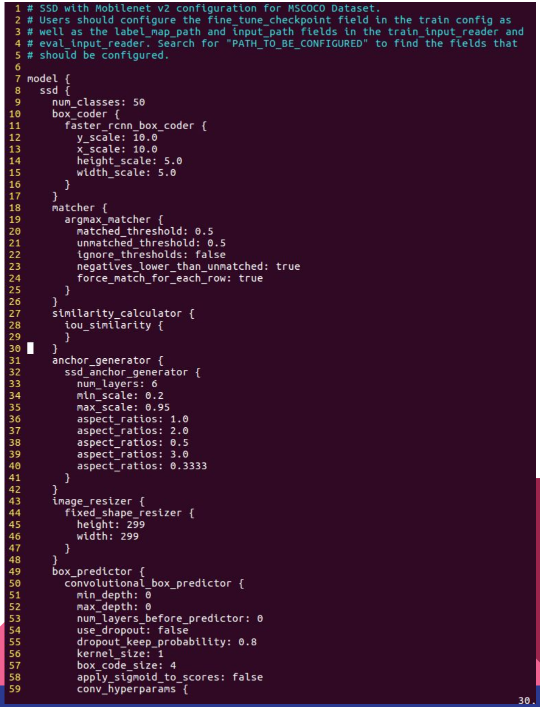
Solution: Shift by 1



```python
def save_to_record(images_addr, labels, files_name, bboxs, cats, option = None):
        if option == None:
            filename = TFRECORD_NAME
        else:
            filename = option
    writer = tf.python_io.TFRecordWriter(filename)
    length = len(images_addr)
    count = 0
    for i in range(length):
        if count % 1000 == 0:
            print("%d out of %d saved" % (count, length))
        image = cv.imread(PATH_TO_FILES + images_addr[i])
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
        (h, w) = image.shape[:2]
                ######Adding this two lines to try
        split = images_addr[i].split(".")
        with tf.gfile.GFile(PATH_TO_FILES + split[0] + "revised.jpg", 'rb') as fid:
            encoded_jpg = fid.read()
        label_num = labels[i]
        example = tf.train.Example(features=tf.train.Features(feature={
            'image/height': int64_feature(299),
            'image/width': int64_feature(299),
            'image/filename': bytes_feature(files_name[i].encode()),
            'image/source_id': bytes_feature(files_name[i].encode()),
            'image/encoded': bytes_feature(encoded_jpg),
            'image/format': bytes_feature(b'jpg'),
            'image/object/bbox/xmin': float_list_feature([bboxs[0][i] / w]),
            'image/object/bbox/xmax': float_list_feature([bboxs[1][i] / w]),
            'image/object/bbox/ymin': float_list_feature([bboxs[2][i] / h]),
            'image/object/bbox/ymax': float_list_feature([bboxs[3][i] / h]),
            'image/object/class/text': bytes_feature(cats[label_num-1].encode()),
            'image/object/class/label': int64_feature(labels[i])
        }))
        writer.write(example.SerializeToString())
        count = count + 1
    writer.close()
    return example
```

Problem 2: Results were Inaccurate

- Fine-tuning pre-trained models
    - can only change the configuration file
        - # of categories
        - the dimensions that the images will be converted to
        - # of maximum training steps
        - Batch size
        - Path to tfrecord files

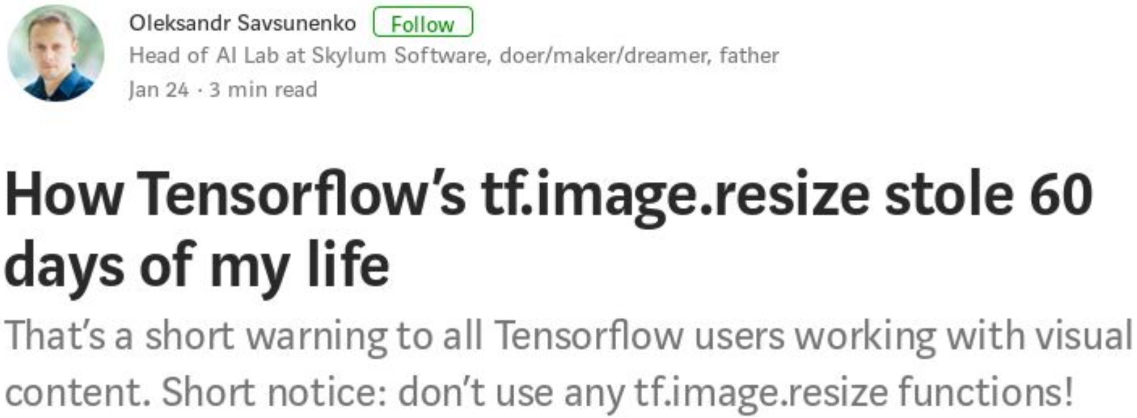


```
# SSD with Mobilenet v2 configuration for MSCOCO Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.

model {
  ssd {
    num_classes: 50
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
    anchor_generator {
      ssd_anchor_generator {
        num_layers: 6
        min_scale: 0.2
        max_scale: 0.95
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        aspect_ratios: 3.0
        aspect_ratios: 0.3333
      }
    }
    image_resizer {
      fixed_shape_resizer {
        height: 299
        width: 299
      }
    }
    box_predictor {
      convolutional_box_predictor {
        min_depth: 0
        max_depth: 0
        num_layers_before_predictor: 0
        use_dropout: false
        dropout_keep_probability: 0.8
        kernel_size: 1
        box_code_size: 4
        apply_sigmoid_to_scores: false
        conv_hyperparams {
                                                                         30.
```

Cause: Images of Different Size

- A bug in the tfrecord image size conversion function
  - ○ Need to resize it ourselves



Oleksandr Savsunenko

Head of AI Lab at Skylum Software, doer/maker/dreamer, father

Jan 24 · 3 min read

How Tensorflow's tf.image.resize stole 60 days of my life

That's a short warning to all Tensorflow users working with visual content. Short notice: don't use any tf.image.resize functions!

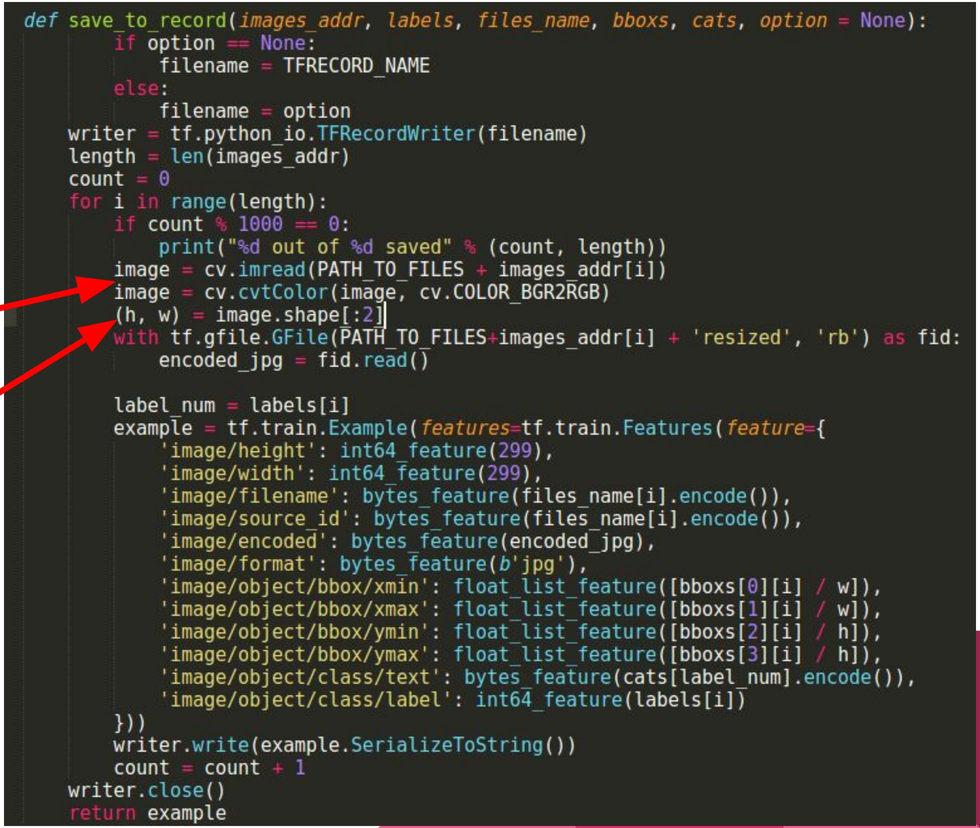
- Resizing images doesn't affect the quality of images
  - ○ The category of each article of clothing - invariant to the size of the image

Solution: Resize them, but how?

- Inside TFRecord
    - Complicated due to bounding boxes
    - Too time-consuming and inefficient

If you resize it here, you need to read it again

Has to be the original height and width to normalize bounding boxes



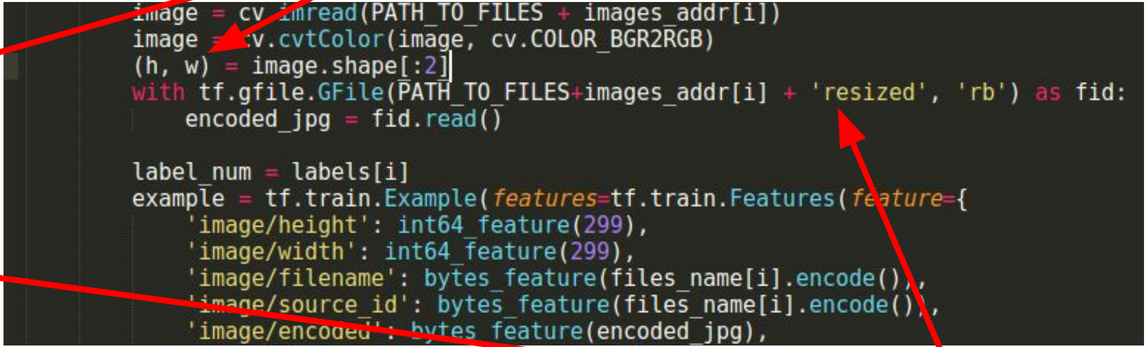
```
def save_to_record(images_addr, labels, files_name, bboxs, cats, option = None):
        if option == None:
            filename = TFRECORD_NAME
        else:
            filename = option
    writer = tf.python_io.TFRecordWriter(filename)
    length = len(images_addr)
    count = 0
    for i in range(length):
        if count % 1000 == 0:
            print("%d out of %d saved" % (count, length))
        image = cv.imread(PATH_TO_FILES + images_addr[i])
        image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
        (h, w) = image.shape[:2]
        with tf.gfile.GFile(PATH_TO_FILES+images_addr[i] + 'resized', 'rb') as fid:
            encoded_jpg = fid.read()

        label_num = labels[i]
        example = tf.train.Example(features=tf.train.Features(feature={
            'image/height': int64_feature(299),
            'image/width': int64_feature(299),
            'image/filename': bytes_feature(files_name[i].encode()),
            'image/source_id': bytes_feature(files_name[i].encode()),
            'image/encoded': bytes_feature(encoded_jpg),
            'image/format': bytes_feature(b'jpg'),
            'image/object/bbox/xmin': float_list_feature([bboxs[0][i] / w]),
            'image/object/bbox/xmax': float_list_feature([bboxs[1][i] / w]),
            'image/object/bbox/ymin': float_list_feature([bboxs[2][i] / h]),
            'image/object/bbox/ymax': float_list_feature([bboxs[3][i] / h]),
            'image/object/class/text': bytes_feature(cats[label_num].encode()),
            'image/object/class/label': int64_feature(labels[i])
        }))
        writer.write(example.SerializeToString())
        count = count + 1
    writer.close()
    return example
```

Solution: Resize them, but how?

- Preprocess the Images
    - Still Complicated due to bounding boxes
    - Much more efficient

Record height and width of the original image



```
image = cv.imread(PATH_TO_FILES + images_addr[i])
image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
(h, w) = image.shape[:2]
with tf.gfile.GFile(PATH_TO_FILES+images_addr[i] + 'resized', 'rb') as fid:
    encoded_jpg = fid.read()

label_num = labels[i]
example = tf.train.Example(features=tf.train.Features(feature={
    'image/height': int64_feature(299),
    'image/width': int64_feature(299),
    'image/filename': bytes_feature(files_name[i].encode()),
    'image/source_id': bytes_feature(files_name[i].encode()),
    'image/encoded': bytes_feature(encoded_jpg),
```

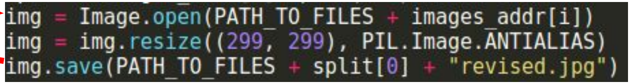
img_00000001.jpg

img_ 00000001revised. jpg

img_00000003.jpg

img_ 00000003revised. jpg

convert_size.py



```
img = Image.open(PATH_TO_FILES + images_addr[i])
img = img.resize((299, 299), PIL.Image.ANTIALIAS)
img.save(PATH_TO_FILES + split[0] + "revised.jpg")
```

But save the resized ones

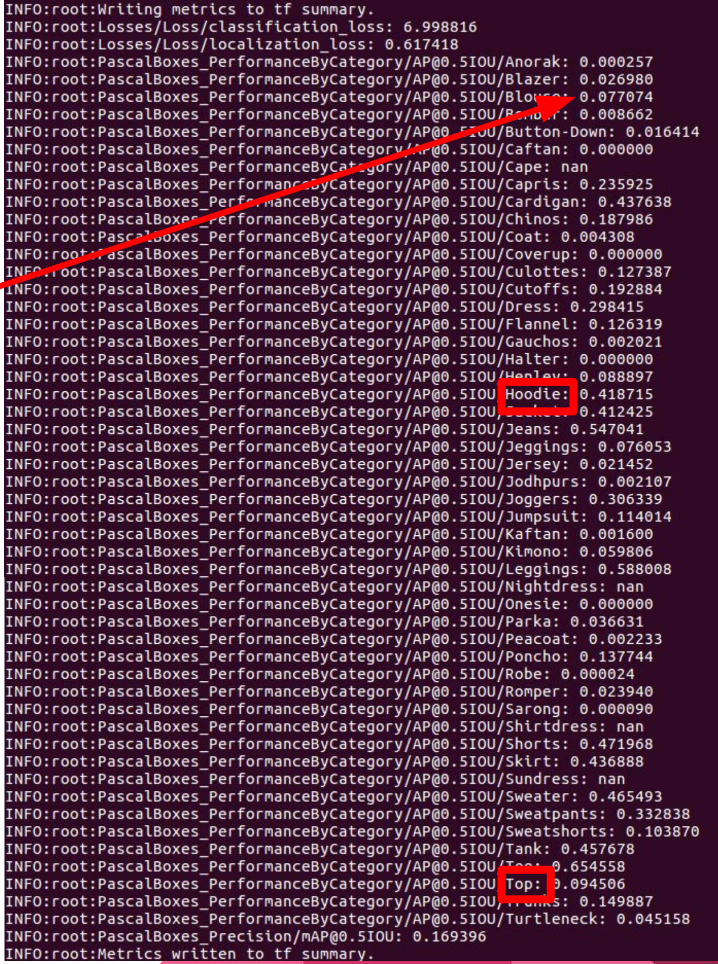
Input

Output

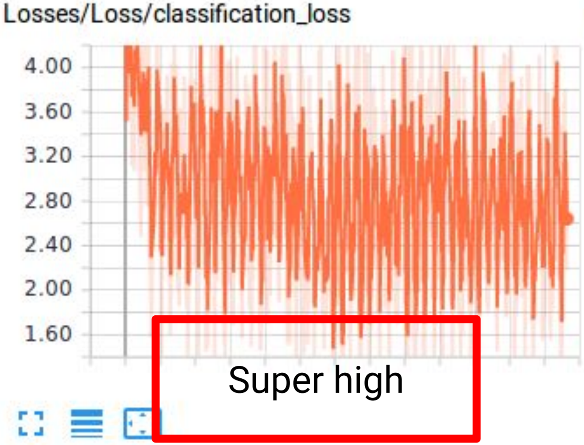
Problem 3: Poor Performance

- Even after four days of training...

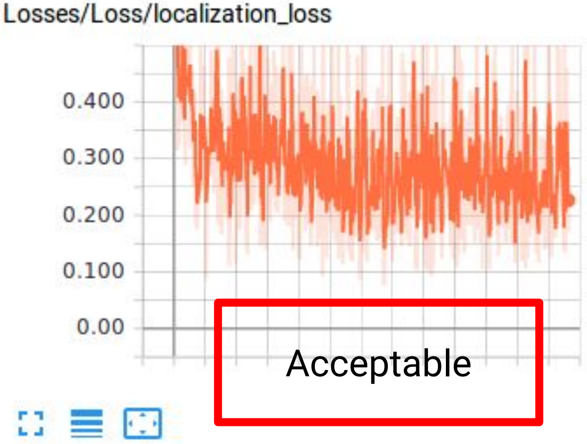
Terrible accuracy for most classes



```
INFO:root:Writing metrics to tf summary.
INFO:root:Losses/Loss/classification_loss: 6.998816
INFO:root:Losses/Loss/localization_loss: 0.617418
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Anorak: 0.000257
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Blazer: 0.026980
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Blouse: 0.077074
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Bomber: 0.008662
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Button-Down: 0.016414
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Caftan: 0.000000
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Cape: nan
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Capris: 0.235925
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Cardigan: 0.437638
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Chinos: 0.187986
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Coat: 0.004308
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Coverup: 0.000000
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Culottes: 0.127387
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Cutoffs: 0.192884
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Dress: 0.298415
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Flannel: 0.126319
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Gauchos: 0.002021
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Halter: 0.000000
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Henley: 0.088897
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Hoodie: 0.418715
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Jacket: 0.412425
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Jeans: 0.547041
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Jeggings: 0.076053
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Jersey: 0.021452
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Jodhpurs: 0.002107
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Joggers: 0.306339
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Jumpsuit: 0.114014
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Kaftan: 0.001600
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Kimono: 0.059806
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Leggings: 0.588008
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Nightdress: nan
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Onesie: 0.000000
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Parka: 0.036631
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Peacoat: 0.002233
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Poncho: 0.137744
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Robe: 0.000024
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Romper: 0.023940
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Sarong: 0.000090
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Shirtdress: nan
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Shorts: 0.471968
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Skirt: 0.436888
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Sundress: nan
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Sweater: 0.465493
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Sweatpants: 0.332838
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Sweatshorts: 0.103870
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Tank: 0.457678
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Tee: 0.654558
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Top: 0.094506
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Trunks: 0.149887
INFO:root:PascalBoxes_PerformanceByCategory/AP@0.5IOU/Turtleneck: 0.045158
INFO:root:PascalBoxes_Precision/mAP@0.5IOU: 0.169396
INFO:root:Metrics written to tf summary.
```



Super high



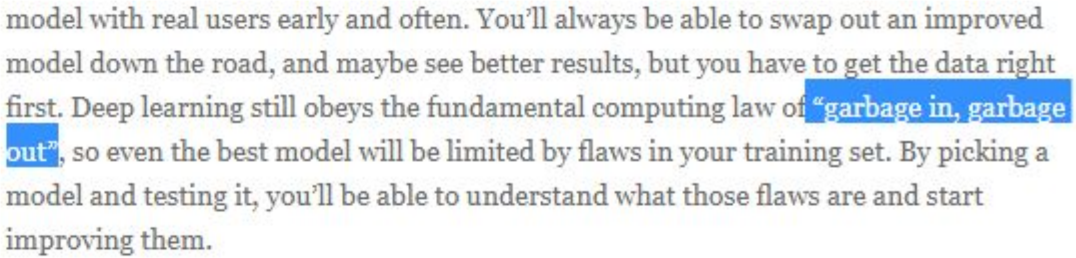
Acceptable

Cause: Terrible Dataset

PETE WARDEN'S BLOG

Ever tried. Ever failed. No matter. Try Again. Fail again. Fail better.

Why you need to improve your training data, and how to do it

- Data is important
    - According to https://petewarden.com/2018/05/28/why-you-need-to-improve-your-training-data-and-how-to-do-it/
    - 

      model with real users early and often. You'll always be able to swap out an improved model down the road, and maybe see better results, but you have to get the data right first. Deep learning still obeys the fundamental computing law of "garbage in, garbage out", so even the best model will be limited by flaws in your training set. By picking a model and testing it, you'll be able to understand what those flaws are and start improving them.
- Model learns garbage from bad data

Bad Data

What We are Working on Right Now

Bad Data

- Inaccurate Data
  - Sorting/Labeling
    - E.g. Suit inside Tees folder
- Poor Quality
  - Bad resolutions
  - Other items covering the object
- Images with:
  - Multiple items
    - (of same object)
  - No models
    - Images with only clothes

Note: These are actual images from our data set

Solution: Manually Clean the Dataset!

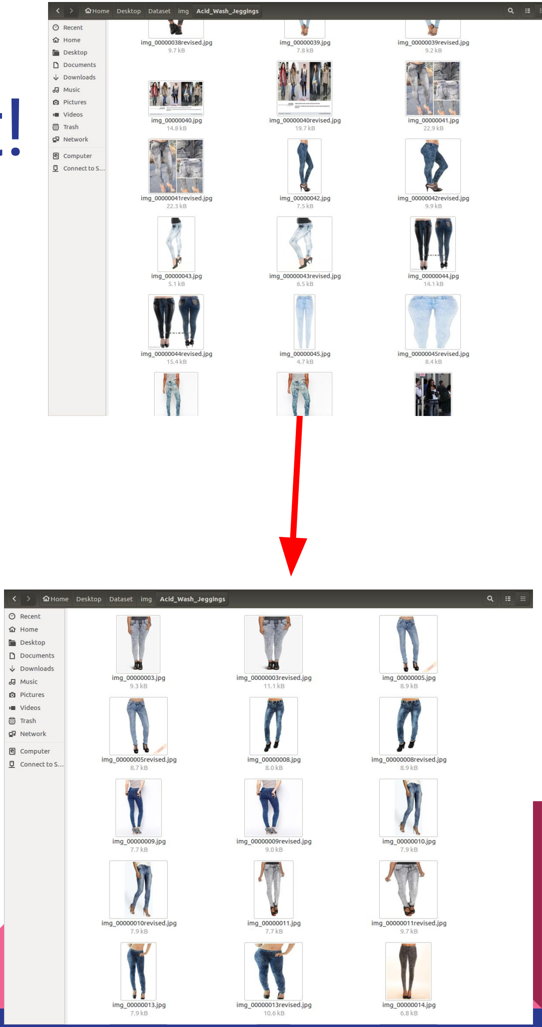
- Keep images only if there is one person
  - Remove those with mannequin
  - Remove those with multiple people
    - Multiple articles of clothing
  - Remove those with no person
- Merge or delete categories
  - Some types of clothing are hybrid
    - If it looks like a dress, then it is a dress, but not a romper
  - Anoraks and Bombers are basically Jackets

Anorak

Jacket

Bomber

Romper that looks like a dress



Future Plan

What We are Working on Right Now

Future Plan

- Still in progress
  - We each pick 3 categories from either TOP, BOTTOM, FULL-BODY
  - Collect 1k images for each categories
- If synchronously finished cleaning data:
  - Train with all the cleaned data we have
- Else:
  - Train our own cleaned data
- Then, keep on cleaning the data during the training
  - Incrementally clean and add more categories
- Come up with more ways to improve the performance of our model

Citations

https://cdn-images-1.medium.com/max/655/1*62JsN-0BZ2rl1QdhNLSN8A.png

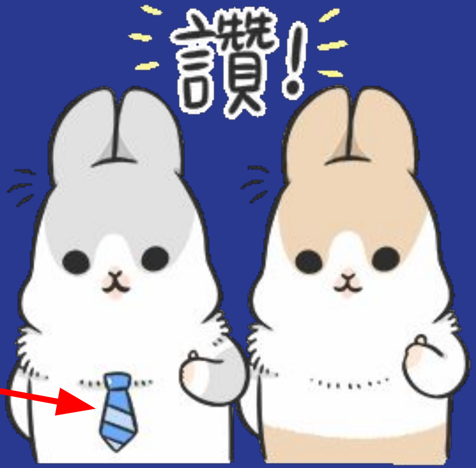
https://cdn-images-1.medium.com/max/603/1*N8H9Z3RDQz2LW_NHh9l3Fw.jpeg

https://i.ytimg.com/vi/_zZe27JYi8Y/hqdefault.jpg

http://mmlab.ie.cuhk.edu.hk/projects/DeepFashion.html

Questions? Advice?

你們是老大!!!!!!!!!!!!!!!!



Fashionable Machiko Rabbit