What to Wear?

Outfit recommendation using computer vision and machine learning

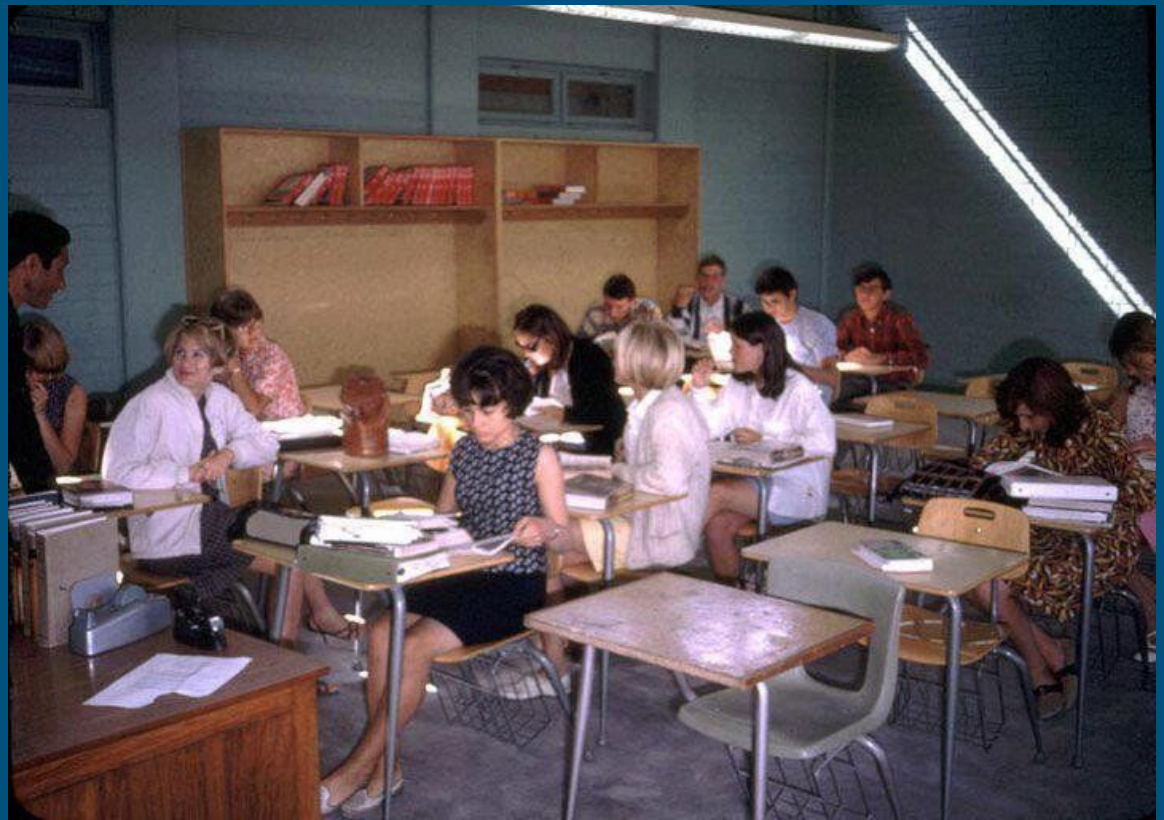
Motivation

Motivation

- You may spend a lot of time deciding what to wear.
- It can affect your self esteem.
- It impacts the way others see you.
- There are multiple etiquettes.
- Depends on where you are.
- Changes on time.

Different places, different “rules”

Different times, different “rules”



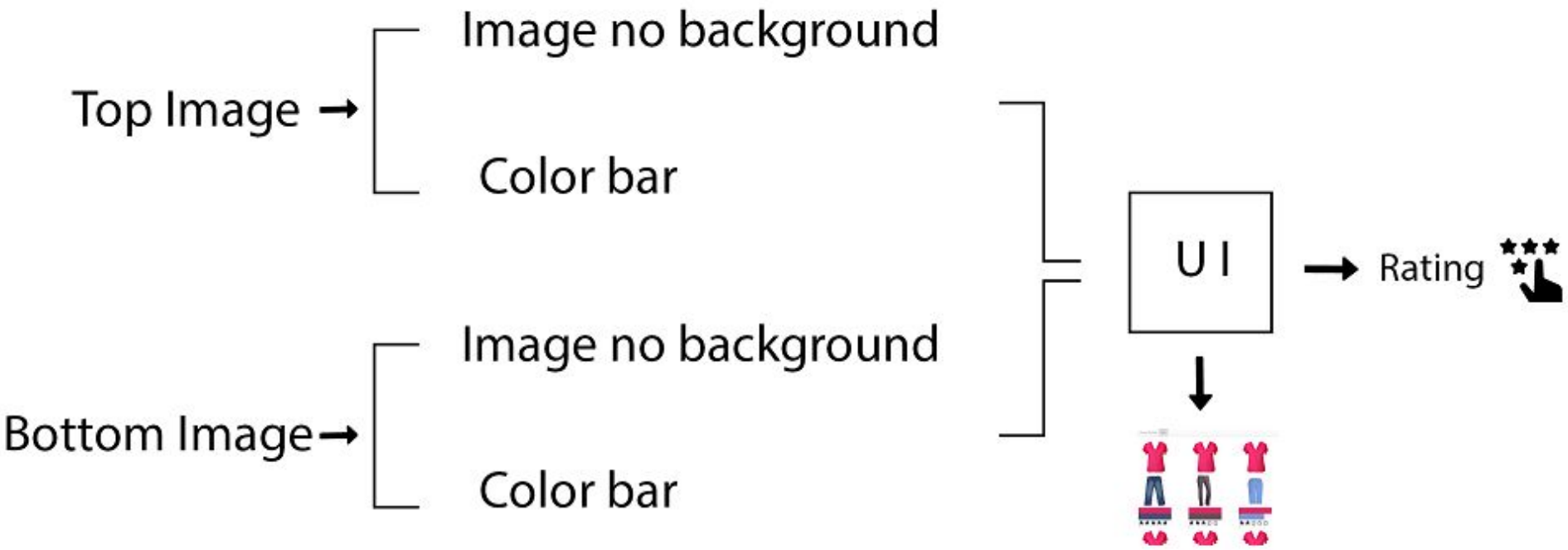
Who am I?

- Computer and Systems Engineer, UTP.
- Student of Master in CS, AI Branch, UTP.
- Software Developer for 6 years.
- Married to a Fashion Designer.
- My master thesis was basically this.

Challenges

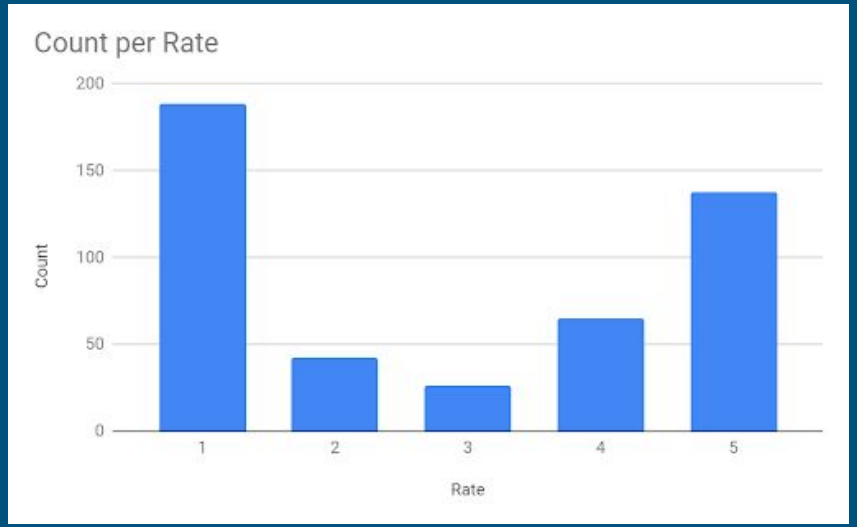
1. Build a DB for this very specific purpose.
2. Segmentate the clothes.
3. Extract features that are relevant.
4. Assemble those features.
5. Train a model to predict rating on 2 new images.
6. Put it all together in a usable way.

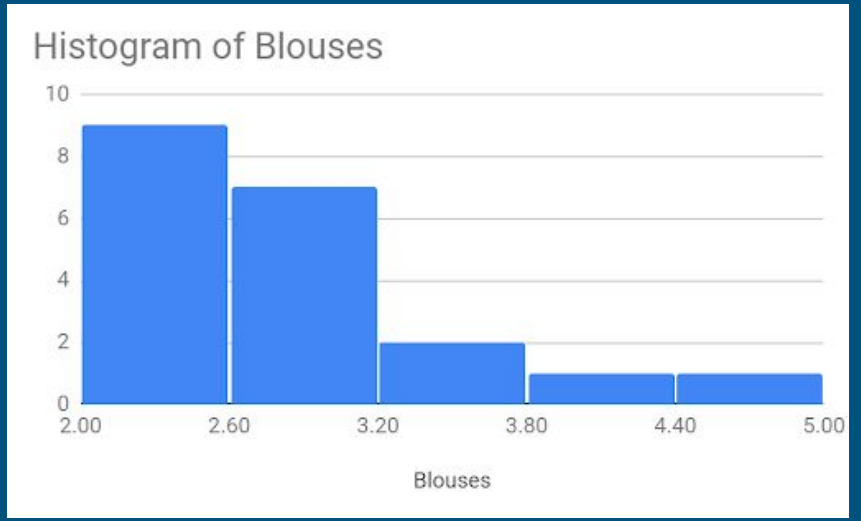
Building a DB



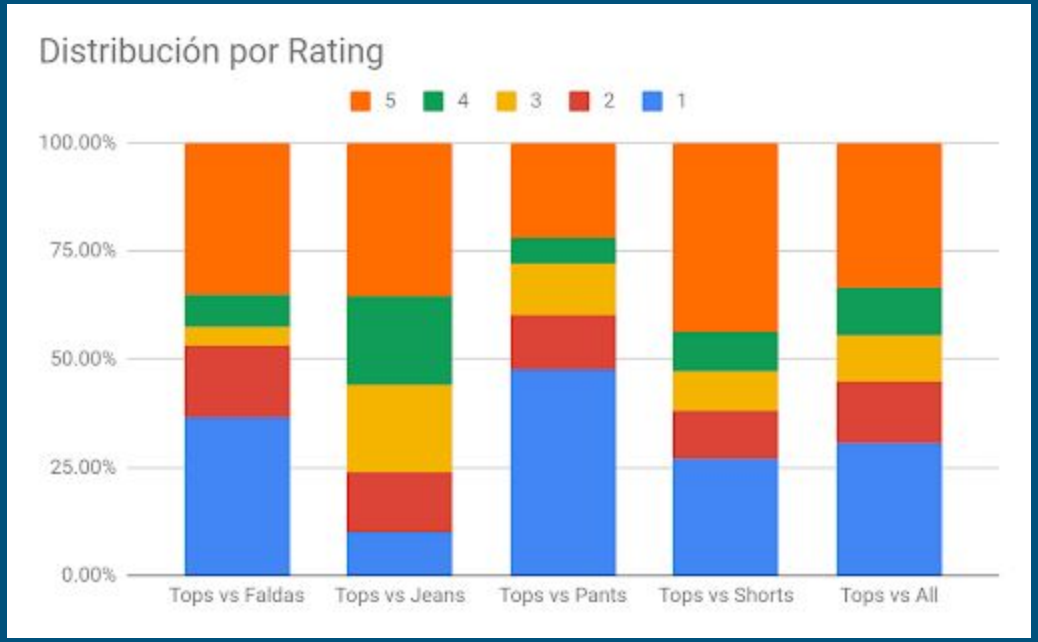
Building a DB: Capturing rating

Building a DB, First attempt: terrible clothes





Building a DB, Second attempt: cool clothes.



Segmentation: GrabCut

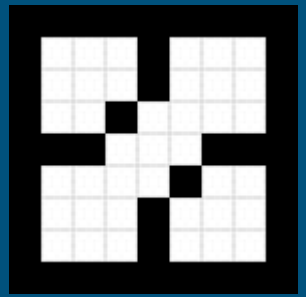
1. User selects a bounding box.
2. Estimates color distribution of target object and background with Gaussian Mixture Model.
3. Builds a Markov Random Field con energy function.
4. Runs a GraphCut based optimization.
5. Repeats until convergence.
6. Can be corrected by user by pointing misclassified regions and re running.

Segmentation: GrabCut

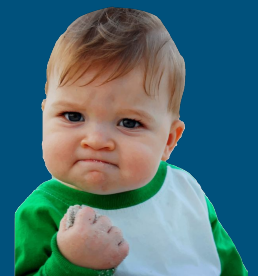
Segmentation: Floodfill

1. If target-color is equal to replacement-color, return.
2. ElseIf the color of node is not equal to target-color, return.
3. Else Set the color of node to replacement-color.
4. Perform Flood-fill to each direction.
5. Done.

Our replacement color in RBGA will have alfa=0



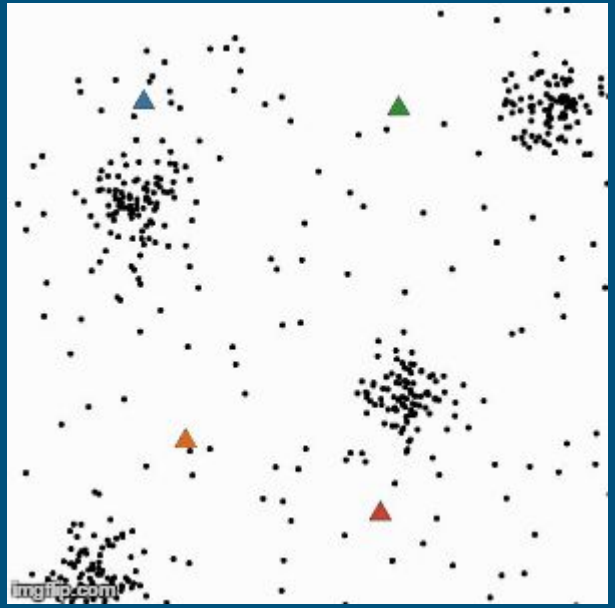
Segmentation: Floodfill



Segmentation: Comparison

- 83% of cases FloodFill is as good or better than GrabCut.
- GrabCut was as good 38% of the time and better 13%.
- For this case, Floodfill is clearly better! (plus, it’s a 100 times faster).
- GrabCut could be more precise though, especially when backgrounds are not plain.

Main Colors: K-Means

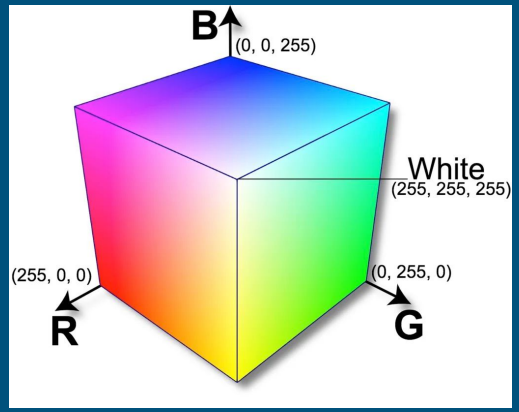


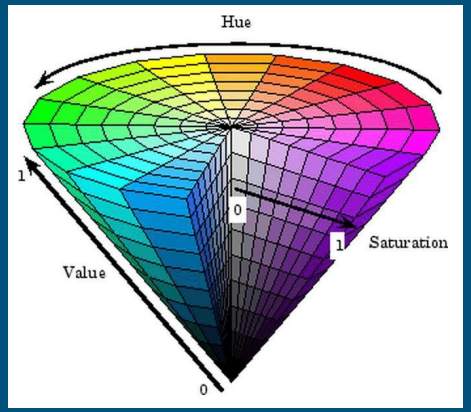
- Find clusters
- Minimize variance inside the group
- Maximize variance between groups
- You need to know K
- My approach for K: heuristic.
  - Start with 3 colors.
  - If 2 of the found colors are too similar (less than 15% difference), run with lower K.
  - If 2 of the found colors are similar (less than 20% difference) and one of them represents less than 5% of the pixels, run with lower K.
  - If K equals 1, you may want to stop now.

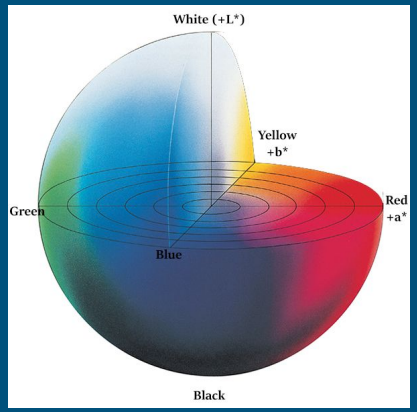
Main Colors: K-Means

Features: Color Histograms

- Color spaces:
  - RGB: red, green, blue.
  - HSV: hue, saturation, value.
  - Lab: lightness, green to red, blue to yellow.
- Bins: 8, for each color component

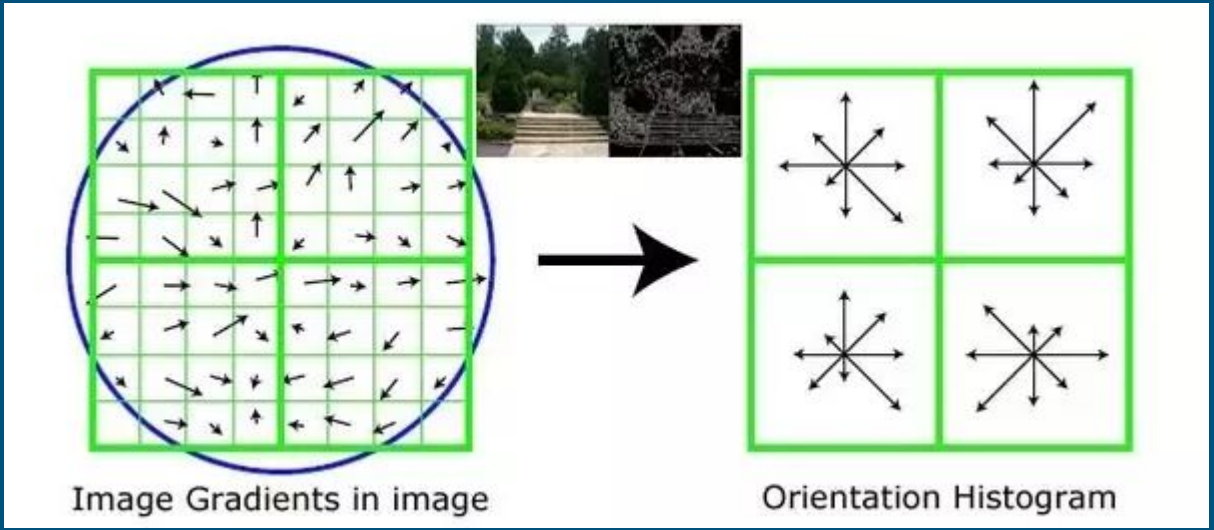






Features: HOG, Histogram of Oriented Gradients

Counts occurrences of gradient orientation in localized portions of an image.

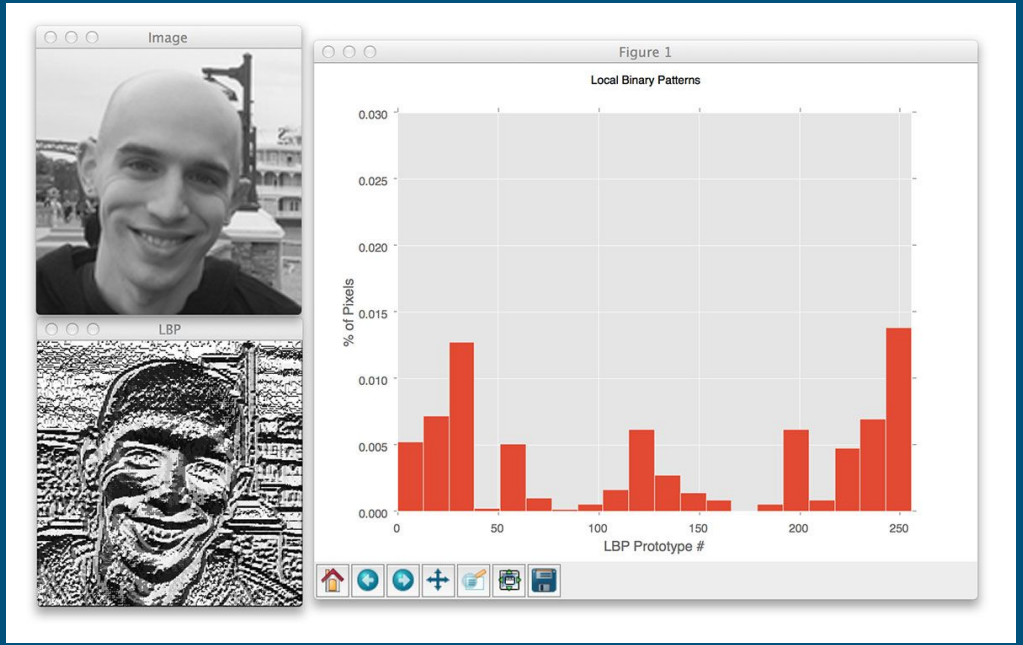


Features: LBP, Local Binary Patterns

Texture descriptor which builds local representations by comparing each pixel with its surrounding neighborhood of pixels.

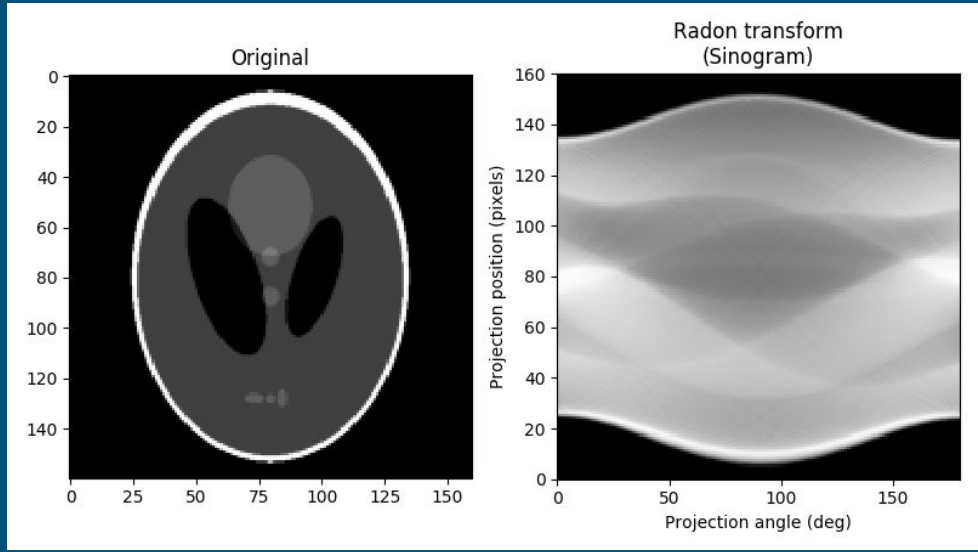
Check https://www.pyimagesearch.com/!

You may also want to check: https://www.learnopencv.com/

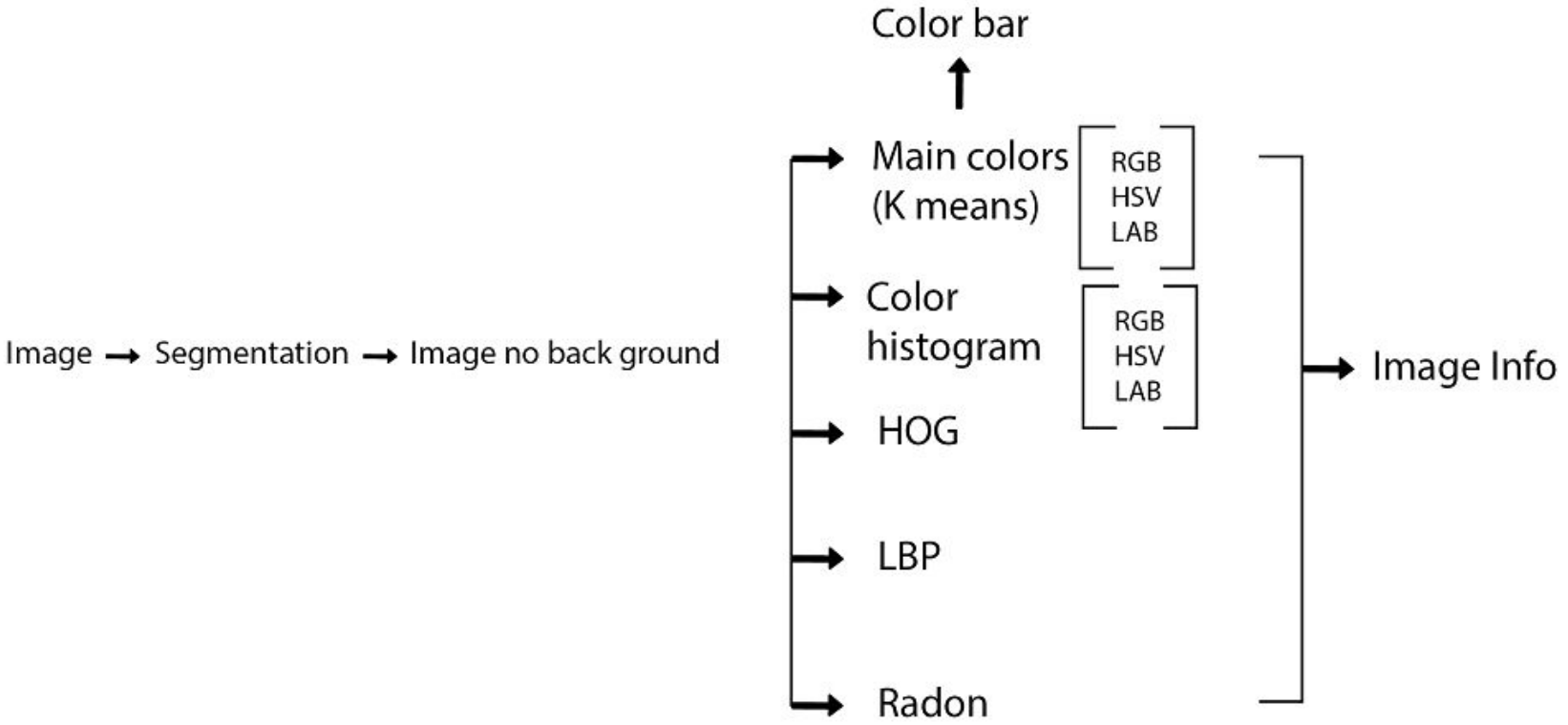


Features: Radon Signature

The Radon transform is the integral transform which takes a function f defined on the plane to a function Rf defined on the (two-dimensional) space of lines in the plane, whose value at a particular line is equal to the line integral of the function over that line



Features: Overview



Assembly: Feature options

Several choices:

- Use main colors extended to 1, 2, 3 or 5 colors, or use color histogram [5].
- Use RGB, HSV, LAB, just H or all of them [5].
- Use LBP with radius 2, 5, 10 or simply not use it [4].
- Use HOG or not [2].
- Use Radon transform or not [2].
- Use dimension reduction: PCA, ICA or none [3].

1200 combinations so far.

Training: Models

Neural Networks: Set of artificial neurons connected with weights and an activation function.

Bayes Regression: Statistical analysis using Bayesian Inference.

SVM with RBF kernel: Take points to a new space where they can be separated by a Hyperplane. RFB is radial basis function.

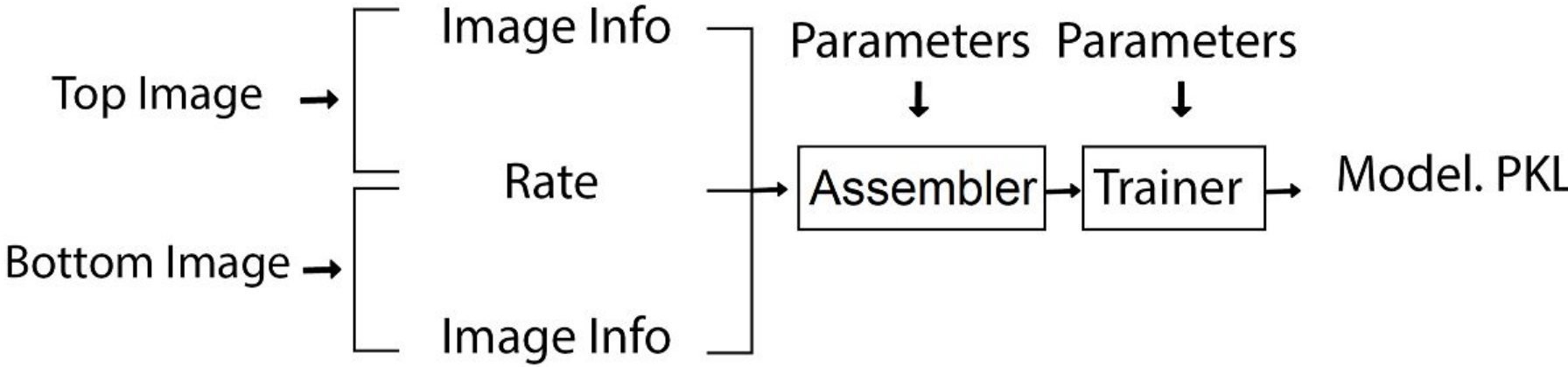
Random Forest: Train a set of decision trees with different samples, then use the mode or the average to predict new cases.

Training: Model options

- SVM: C can be 10^1, 10^2, 10^3, 10^4. Gamma 10^-1, 10^-2. [8]
- Neural Networks. Vary the number of hidden layers. [4]
- Random Forest. Num of trees: 50, 100, 200. Max depth: 5, 6, 7, 8. [12]
- Bayes Regression. [1]

25 unique configurations.

Training: Overview



Training: All options

I tested each independent category (tops vs pants, tops vs …) and all together. [5]

25 model options * 1200 assembly options * 5 sets = 150k.

If you do cross validation, WHICH YOU SHOULD, you get (with 5 folds):

750k unique trainings

Training: GridSearchCV

- Performs exhaustive search.
- Receives a param grid.
- Has flexible scoring.
- Allows cv with Stratified KFold or custom splitting.
- Finds the best parameters.
- Gives you a dictionary with cv_results.
- You'll need more than that.

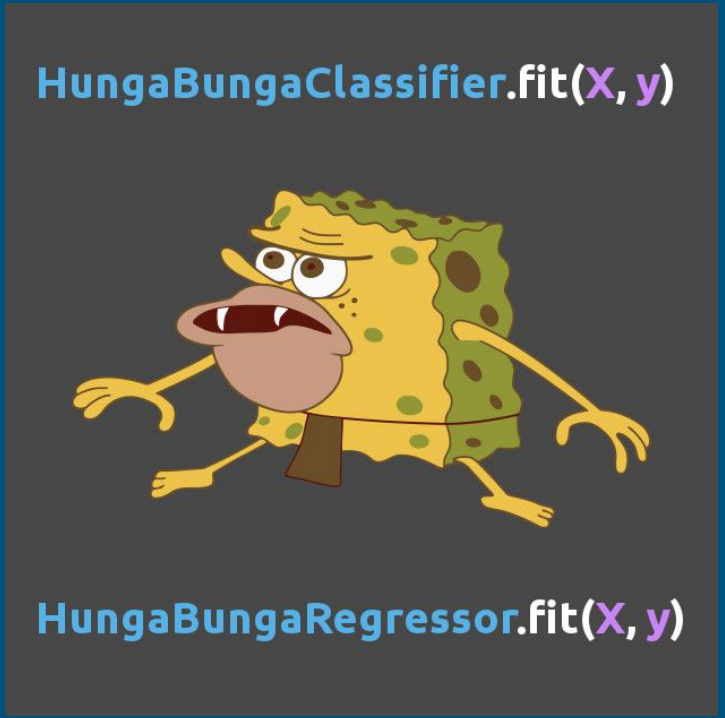
```python
def train_and_validate(
    tops, bottoms, num_folds, names_and_model_functions,
    color_spaces_options, n_jobs, colors_per_image_options,
    save_best_model=False, run_for_each=True, run_for_all=True,
    **kwargs,
):
    if run_for_each:
     1  for top in tops:
        2  for bottom in bottoms:
              3 for color_spaces in color_spaces_options:
                  4 for colors_per_image in colors_per_image_options:
                        X, y, dimensions = get_x_and_y_from_options([top], [bottom], color_spaces, colors_per_image, **kwargs)
                        kwargs['dimensions'] = dimensions

                        5 for model_name, model_fun, model_params in names_and_model_functions:
                            log_current(top, bottom, color_spaces, model_name, colors_per_image, **kwargs)
                            grid = train_configuration(model_name, model_fun, model_params, X, y, num_folds, n_jobs)
                            full_set_score = validate_model(grid.best_estimator_, X, y)
                            results = get_results_from_grid(
                                grid=grid,
                                top=top,
                                bottom=bottom,
                                model_name=model_name,
                                color_spaces=color_spaces,
                                num_folds=num_folds,
                                colors_per_image=colors_per_image,
                                **kwargs,
                            )
                            results['full_set_score'] = full_set_score
                            yield results

                            if save_best_model:
                                filename = get_model_filename(top, bottom, model_name, colors_per_image)
                                joblib.dump(grid.best_estimator_, filename)
```

Training: GridSearchCV not enough?

Try HungaBunga!

“Brute Force all scikit-learn models and all scikit-learn parameters with fit predict.”

https://github.com/ypeleg/HungaBunga



Training: Results

Bottom	Color Spaces	Colors per Img	Model	LBP	HOG	Radon	Red.	Mean Test Score	Std Test Score	Best Params
Shorts	RGB	5	Forest	No	Yes	No	PCA	0.881	0.028	Max Depth: 7, N Estimators: 50
Shorts	HSV	2	Forest	No	Yes	No	PCA	0.849	0.092	Max Depth: 8, N Estimators: 50
Jeans	RGBHSV	2	Neural Net.	Yes (R 5)	Yes	Yes	None	0.813	0.056	Hidden Layers: (50, 50, 50), Max Iter: 5000
Jeans	*	2	Neural Net.	No	Yes	No	None	0.812	0.027	Hidden Layers: (50, 50, 50), Max Iter: 5000
Skirts	RGB	5	Forest	No	No	Yes	PCA	0.801	0.017	Max Depth: 7, N Estimators: 100
Skirts	H	Histogram	Forest	No	Yes	No	PCA	0.778	0.026	Max Depth: 8, N Estimators: 50
Pants	*	3	SVM(Rbf)	No	No	Yes	ICA	0.784	0.023	C: 1000, Gamma: 0.1
Pants	RGB	Histogram	SVM(Rbf)	Yes (R 10)	Yes	Yes	None	0.784	0.022	C: 100, Gamma: 0.01
All	LAB	2	Forest	No	No	No	None	0.765	0.042	Max Depth: 8, N Estimators: 200
All	*	Histogram	Forest	Yes (R 10)	Yes	Yes	PCA	0.763	0.026	Max Depth: 8, N Estimators: 100

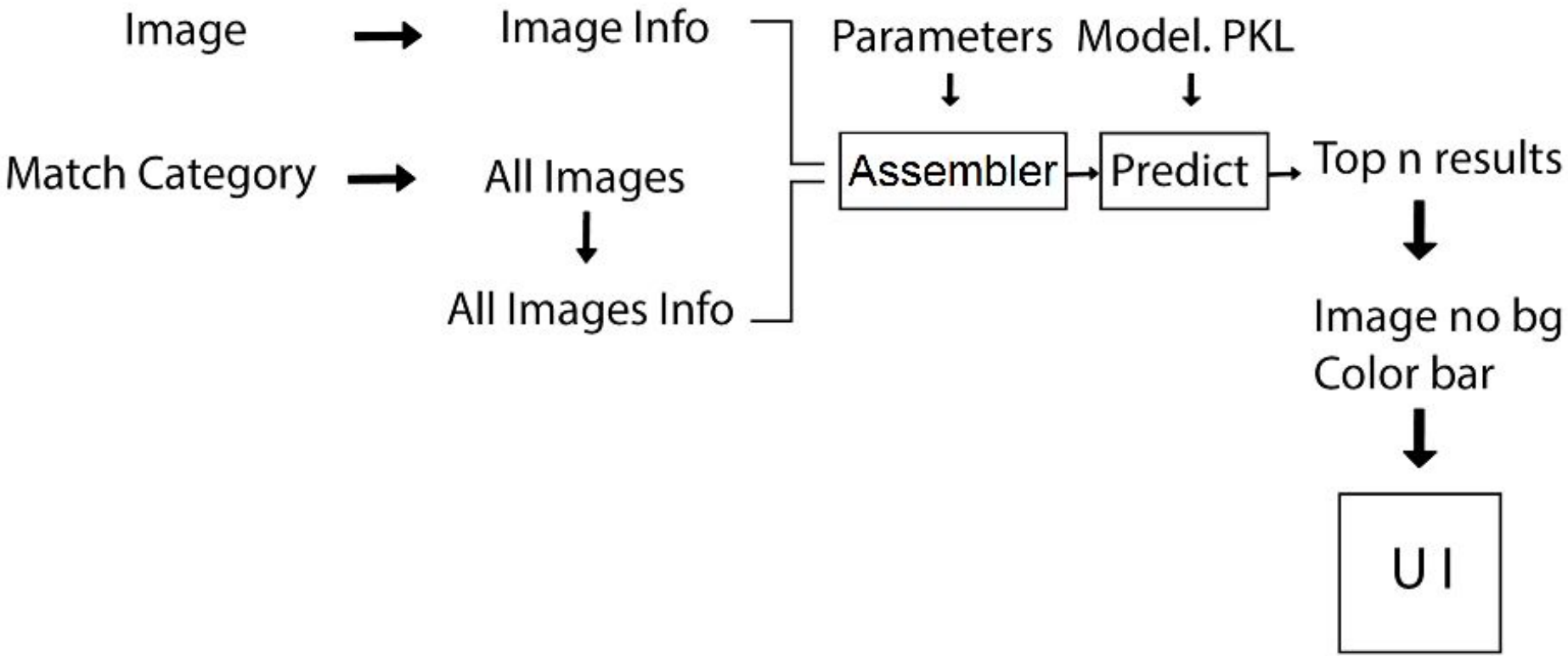
Training: Results using Random Forest Only

For random forest the best parameters in general were:

- Use 2 colors per image or histogram (cheaper).
- Color space: LAB
- No LBP, no HOG, no Radon.
- No dimension reduction.

In average we get 74.1% ± 5% accuracy among the 5 sets.

Putting it all together: Overview



Putting it all together

Tools used from sklearn

- **GridSearchCV**: sklearn.model_selection.GridSearchCV
- **KMeans**: sklearn.cluster.KMeans
- **SVM**: sklearn.svm
- **Random Forest**: sklearn.ensemble.RandomForestRegressor
- **Bayes Regression**: sklearn.linear_model.BayesianRidge
- **Neural Networks**: sklearn.neural_network.MLPRegressor
- **PCA**: sklearn.decomposition.PCA
- **ICA**: sklearn.decomposition.FastICA

Tools used from skimage and opencv

- **GrabCut**: cv2.grabCut
- **FloodFill**: cv2.floodFill
- **LBP**: skimage.feature.local_binary_pattern
- **Radon**: skimage.transform.radon
- **HOG**: cv2.HOGDescriptor
- **Convert color spaces**: cv2.cvtColor

Learnings

- Analyse your data before training. It might be biased.
- Use tools as GridSearchCV.
- Define clearly your validation function, and test it.
- Use different models.
- Don’t try to do every combination, do some screening first.
- Have some basic understanding of what you’re using.

Ideas for Future Work

1. Handle more than 2 clothes, for instance include also shoes.
2. Use GrabCut with user feedback.
3. Try other models: Gaussian Mixture, Deep Learning…
4. Try other features: ???

Questions?


شكراً جزيلاً
ngiyabonga
рахмат
danke
謝謝
teşekkür ederim
Баярлалаа
mersi
kia ora
barka
welalin
spas
tack
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
спасибі
blagodaram
dank je
misaotra
matondo
paldies
grazzi
thank you
gracias
kiitos
dankie
nanni
nandri
dhanyavad
mauruuru
köszönöm
хвала
asante
manana
obrigada
murakoze
tenki
bedankt
bayarlalaa
gràcie
hvala
dankon
aciū
akun
djiere dieuf
tau
mochchakkeram
enkosi
dziękuję
дякую
mamnun
chnorakaloutioun
gratias ago
gràcies
sulpáy
go raibh maith agat
chokrane
sobodi
dĕkuji
obrigado
sukriya
kop khun krap
taiku
sagolun
arigatô
takk
dakujem
trugarez
mèsi
didi madloba
ありがとう
najis tuke
tanemirt
grazie
terima kasih
rahmet
rahmat
kam sah hamnida
ευχαριστώ
diolch
dhanyavadagalu
shukriya
merce
мерси
তোমাকে ধন্যবাদ
감사합니다
xièxie
merci