Benchmarking Facial Beauty Predictors

Motivation and Use Case

We set out to explore and benchmark the visual recognition problem of Facial Beauty Prediction (**FBP**), assessing facial attractiveness that is consistent with human perception. Such a FPB model can be used by online dating companies as a dating profile recommender. E.g: *When a brand new user Bob signs up on a dating website and uploads his personal photograph, that photograph can be fed to our model which outputs a number from 1 to 5 indicating how attractive his face looks. Based on his beauty score, the dating company could take strategic recommendation decisions. An example of such a decision could be that if the model rated him 2.2, Bob's profile would be shown to women rated in upper 4s because historically they noted that women with beauty scores in the upper 4s have gone out on dates with guys with beauty scores in the lower 2s.*

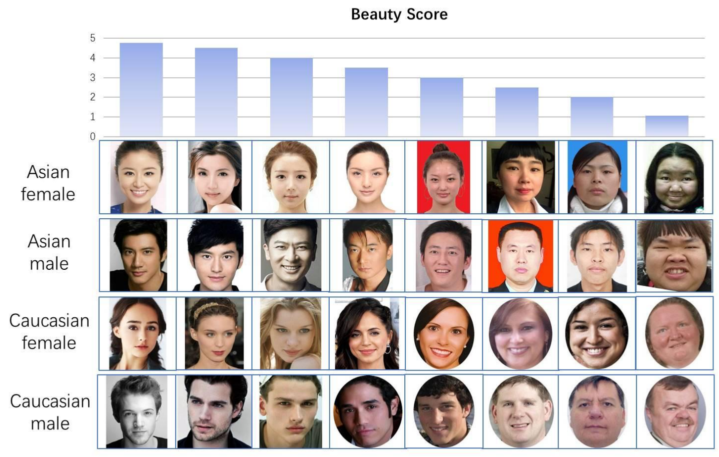
Introduction

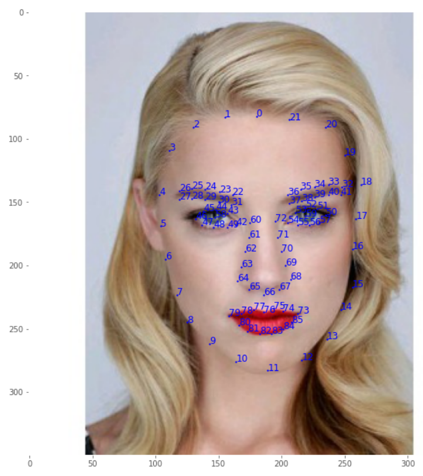
To tackle this problem, various computational models with different FBP paradigms were explored. We considered this problem a semantic topology problem so we focused our efforts on comparing the feature generation and modeling process of machine learning pipelines with various convolutional neural network (**CNN**) approaches. Specifically, we analogously engineered our machine learning pipelines to perform feature engineering similar to that of a CNN— extracting structure and topics from the topology of a headshot image. This allowed us to interpret the impact and efficiency of CNN layers as they relate to convolutions, pooling, weighting, and decision functions. And our goal was to build a model that beat the Root Mean Squared Error achieved by state-of-the-art CNNs such as AlexNet, ResNet-18 and ResNeXt-50 on the held-out *test* set.

Dataset

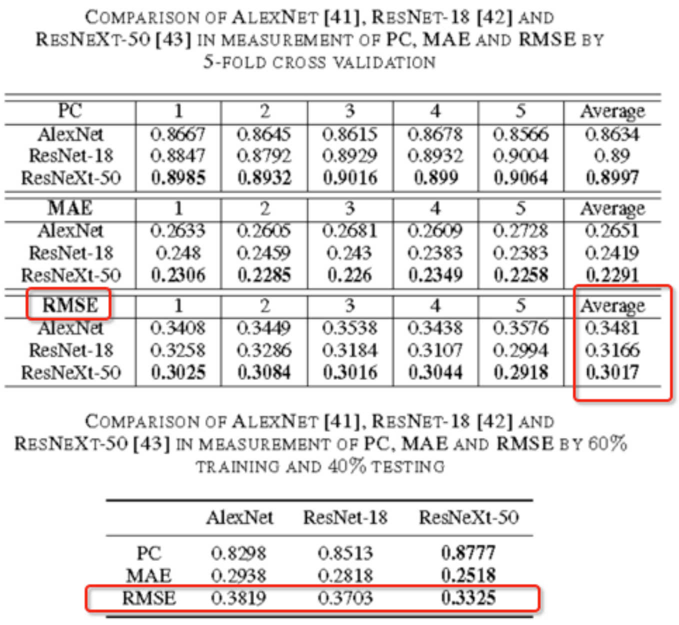
We used a multi-paradigm facial beauty prediction dataset named **SCUT-FBP5500** released by the *Human Computer Intelligent Interaction Lab* of *South China University of Technology.* The SCUT-FBP5500 dataset has 5500 frontal faces with diverse properties. The dataset can be divided into four subsets: ***2000 Asian females(AF), 2000 Asian males(AM), 750 Caucasian females(CF) and 750 Caucasian males(CM)***. It is also *pre-divided* into a 5 fold cross validation set as well as a 60%/40% train-test split. The dataset can be downloaded from https://github.com/HCIILAB/SCUT-FBP5500-Database-Release in the form of a zipped folder. The most important variable in the dataset was the target variable (label) namely the 'Beauty Score'. All the images are labeled with *beauty scores ranging from 1 to 5 by 60 volunteers* and as a measure of rater reliability, some raters were randomly required to rate some of the images multiple times. Additionally each image has *86 artificially labelled facial landmarks* attributed in a pts format (see image below for mapping).

The Human Computer Intelligent Interaction Lab used AlexNet, ResNet-18, and ResNeXt-50 as the benchmarks of the SCUT-FBP5500 dataset, and evaluated the benchmarks on various measurement metrics, including: Pearson correlation (PC), maximum absolute error (MAE), and root mean square error (RMSE). We took note of these metrics to compare our models' performance to these benchmarks. The benchmark *RMSE on the test set ranged from 0.3325 to 0.3819, equating to a MSE of 0.665* (for ResNEXt-50) to 0.7638 (for AlexNet)





(Left) Beauty Scale credit SCUT-FBP5500; (Right) Facial Landmarks



COMPARISON OF ALEXNET [41], RESNET-18 [42] AND RESNEXT-50 [43] IN MEASUREMENT OF PC, MAE AND RMSE BY 5-FOLD CROSS VALIDATION

PC	1	2	3	4	5	Average
AlexNet	0.8667	0.8645	0.8615	0.8678	0.8566	0.8634
ResNet-18	0.8847	0.8792	0.8929	0.8932	0.9004	0.89
ResNeXt-50	**0.8985**	**0.8932**	**0.9016**	**0.899**	**0.9064**	**0.8997**
MAE	**1**	**2**	**3**	**4**	**5**	**Average**
AlexNet	0.2633	0.2605	0.2681	0.2609	0.2728	0.2651
ResNet-18	0.248	0.2459	0.243	0.2383	0.2383	0.2419
ResNeXt-50	**0.2306**	**0.2285**	**0.226**	**0.2349**	**0.2258**	**0.2291**
RMSE	**1**	**2**	**3**	**4**	**5**	**Average**
AlexNet	0.3408	0.3449	0.3538	0.3438	0.3576	0.3481
ResNet-18	0.3258	0.3286	0.3184	0.3107	0.2994	0.3166
ResNeXt-50	**0.3025**	**0.3084**	**0.3016**	**0.3044**	**0.2918**	**0.3017**

COMPARISON OF ALEXNET [41], RESNET-18 [42] AND RESNEXT-50 [43] IN MEASUREMENT OF PC, MAE AND RMSE BY 60% TRAINING AND 40% TESTING

	AlexNet	ResNet-18	ResNeXt-50
PC	0.8298	0.8513	**0.8777**
MAE	0.2938	0.2818	**0.2518**
RMSE	0.3819	0.3703	**0.3325**

SCUT-FBP5500 published benchmarks

Methodology

As a conceptual overview to our approach, we leveraged various unsupervised learning methods for facial feature extraction, visual topic modeling and dimensionality reduction. We then carried over these transformed features to various supervised learning models for beauty score classification and regression.

Unsupervised Learning

For unsupervised learning, we used the *Bag of Visual Features "Words"* (**BoVW**) method, which is analogous to the "Bag of Words" (BoW) method used in NLP for document classification. BoVW method follows four simple steps: **(1)** Feature extraction from images; **(2)** Construction of visual vocabulary, “topics”, by clustering (K-means); **(3)** Image representation as a histogram of visual topic frequencies **(4)** Classification of images based on similarity to vocabulary generated — Obtain optimum class for query image.

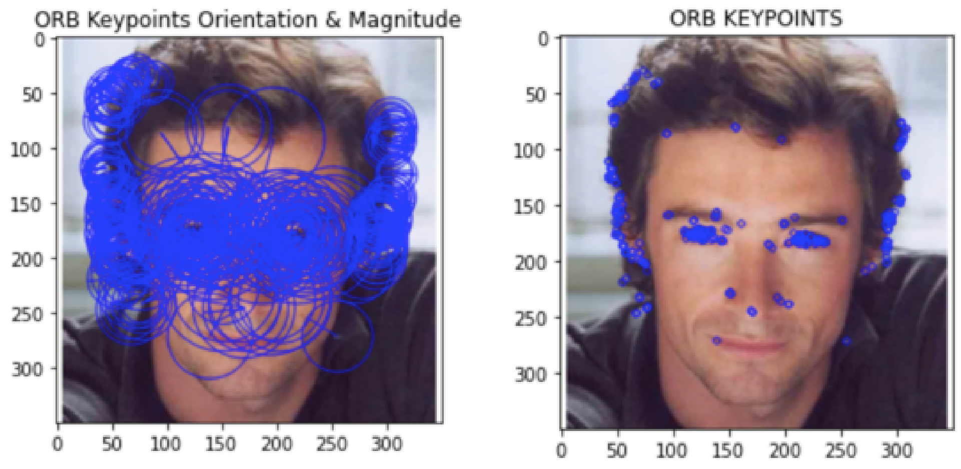
Feature Extraction

As stated one of our goals in unsupervised learning was to extract features manually such that the features have the same properties of the features extracted from a CNN layer, namely location invariance, scale invariance and orientation invariance—Scale Invariant Feature Transform (**SIFT**). This was an exercise that helped us understand the automatic features extraction, unsupervised learning, process of CNNs and set a baseline for our CNN models.

We extracted a maximum of **300 SIFT features** (also called keypoints) from each image in our training set using the ORB algorithm implementations within OpenCV. Each keypoint has a corresponding *descriptor*. *Keypoints* can be thought of as the most distinctive points of interest in an image that can be used to compare images and perform tasks such as image alignment, registration and object tracking. A *descriptor* is a vector of numbers that describes the visual appearance and properties of the key point. Descriptors can be used to compare and match key points across different images.

Each keypoint has the following properties:

- **x, y**: Keypoint coordinates in the image it was extracted from.
- **response**: “Strength” of a feature. A higher score means a higher local contrast.
- **angle**: Orientation angle (in radians)
- **scale**: Radius of the support region (in pixels)
- **level**: Scale level from which the feature was extracted. Starts at zero.

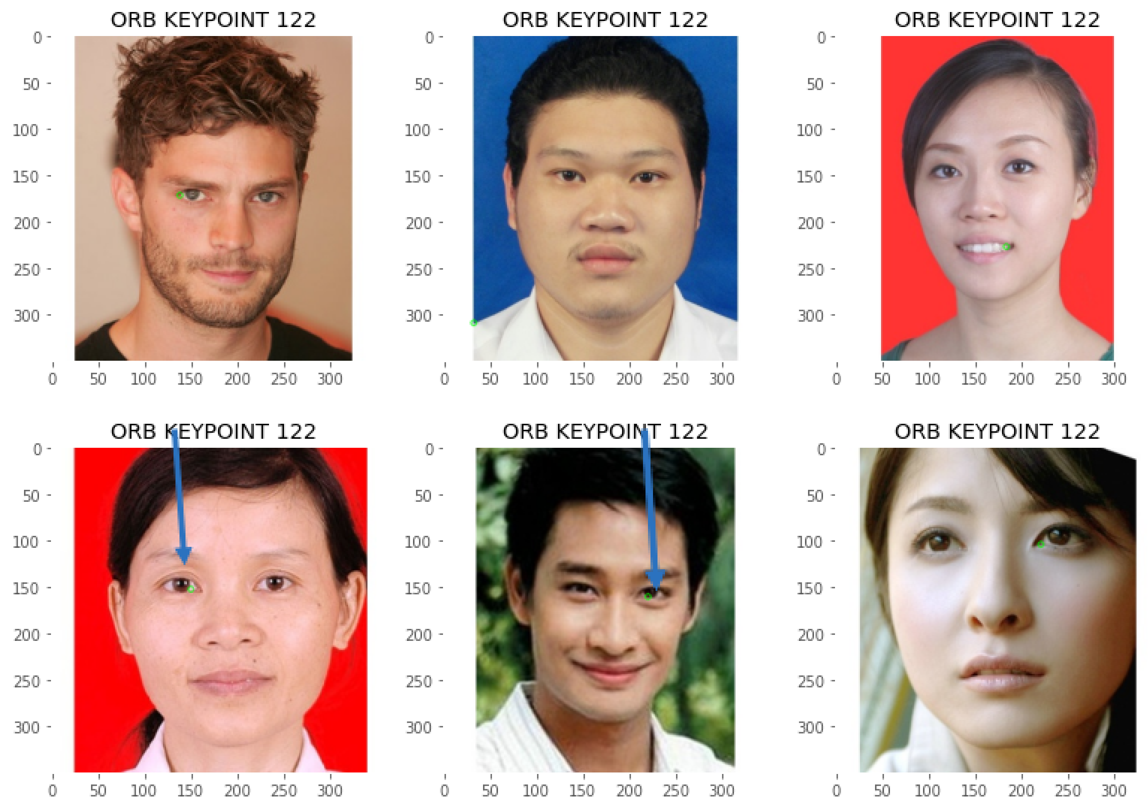


Construction of Visual Vocabulary

After feature extraction on the training set, we ended up with a total of **631,264 feature vectors**. We then performed K-Means Clustering in SciKit Learn *to cluster the feature vectors into 500 clusters*. The center of each cluster was computed and considered to be a **visual topic** that represents many ORB keypoints. This is analogous to topic modeling in natural language processing where a single topic is summarized by several words.

Histogram of visual topic frequencies

We then calculate the Euclidean distance of each keypoint in an image to the visual topic and assign the keypoint to that visual topic. This leaves us with a topic histogram matrix for each image that we then use in our supervised learning classification. In terms of **parameter tuning for K-Means**, we tuned to *10 initialized* runs since each initialization is random and we chose *K = 500* topics because the algorithm struggled with larger Ks — we tried 86 and 2200.

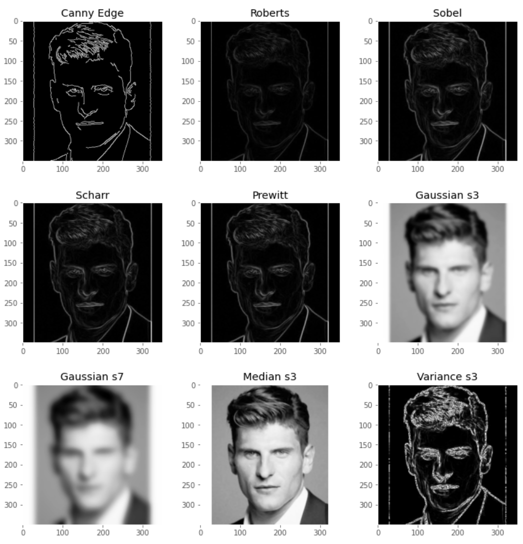


Representation of keypoints (green dots) mapped to cluster centroid #122 (Feature 122/500)

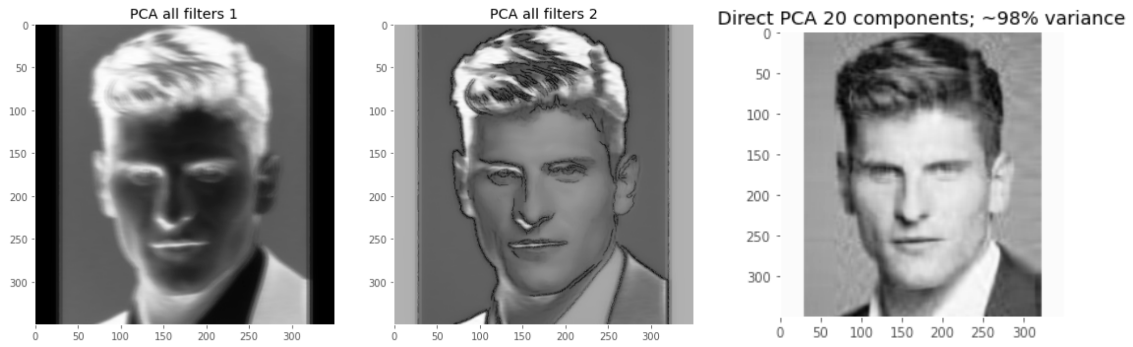
Convolution

In addition to calculating SIFTs, we also conduct image convolution in our machine learning pipeline, again analogous to our CNN. The challenge here was to manually conduct meaningful convolution. Gabor filtering is a method for detecting features in an image in the frequency domain. In the frequency domain, each point in the image can be thought of as representing a particular frequency contained in the spatial domain. E.g edges of the nose and ears likely have high frequencies whereas the pixels corresponding to hair likely have low frequencies. Gabor filters are features whose values determine the presence or absence of such frequencies in a given image. We generated **32 such Gabor features**, each being an array of size 350 x 350 whose numerical values determine the presence or absence of a range of frequencies in a given image. We also extracted **9 additional features** using methods like Canny Edge, Roberts, Sobel, Scharr, Prewitt, Gaussian s3 and Gaussian s7). Each of these methods yields a feature of array size 350 x 350. All of these 9 methods detect edges of slightly different types and textures.

Since our machine learning pipeline was conceptually analogous to our CNN, we also conducted pooling via the unsupervised learning method of Principal Component Analysis (**PCA**). Because PCA transforms representation of an image we considered multiple ways of reduced representation. *Specifically we* *considered*: **(1)** Reducing from the raw image to understand dimensionality reduction without convolution; **(2)** Reducing from 32 Gabor filters by themselves. **(3)** Reducing from the 9 edge filters; **(4)** Reducing from all the convolution layers. Along with exploring different spaces to reduce from, we also considered how many components to use in the end. In the end, *we chose 2 components* as the variance captured in the first component alone varied greatly depending on the image (*65%-85%*). Whereas taking 2 components on average yielded >80%. We found that processing convolutions created a "sharper" reduced representation of an images representation than reducing from the raw image. We then transferred these features to our supervised learning models.



Edge filters



PCA Results

Supervised Learning

Dummy Regression Model

We created a dummy model which learns how to calculate the average beauty score of all the training examples it has seen. Intuitively this can be thought of as teaching the dummy model to predict that each new user that signs up on the dating website looks just as good as ALL other existing users. So if all the existing users are very good looking on average, then the dummy model will predict that a new user who just signed up on the website also looks very good! Maybe the dating website attracts a certain segment of users who all look very good! For the dummy model, we calculate the average beauty score of a face in the training dataset np.mean(y_train) = 2.99. This average represents a typical user in the dataset. So according to the dummy strategy, the model will predict each new user (i.e. test image) to have a beauty score of 3.0

Deep Learning using CNN (Convolution Neural Network)

We built a CNN using Tensorflow and Keras libraries. Our CNN architecture is shown below

Feature Extraction Layers

Layer 1 - 100 Convolution filters with kernel size 10 x 10 activated by ‘relu’ function
Layer 2 - MaxPooling Layer for dimensionality reduction of feature maps
Layer 3 - 50 Convolution filters with kernel size 10 x 10 activated by ‘relu’ function
Layer 4 - MaxPooling Layer for dimensionality reduction of feature maps

Regression Layers

Layer 5 - Flatten layer
Layer 6 - Dense layer with 30 neurons each activated by ‘relu’ function
Layer 7 - Dense layer with 10 neurons each activated by ‘relu’ function
Layer 8 - Dense layer with 1 neurons activated by ‘linear’ function

In our search for the ideal CNN architecture, we started off experimenting by building CNNs with the general intuition that deep neural networks with smaller kernels produce higher accuracy than shallower neural networks with larger kernels. Our first attempt involved building a CNN with 8 convolution layers and kernel size of 3 x 3. We observed that this CNN performed well on the training set but fared poorly on the test set, indicating overfitting. Hence, we reduced the depth of the neural network in successive attempts to 6 layers, 4 layers and eventually 2 layers. The CNN with 2 convolution layers and larger kernel size of 10 x 10 performed much better on the test set and generalized better compared to the other architectures we attempted. This finding was similar to AlexNet which had large kernel sizes. The larger kernels perhaps generalized better than smaller kernels by avoiding picking up ultra small features which may have been just noise. We intuit that if we had a dataset with images of the same size (350 x 350 pixels), but with pictures of showing a full person or with a person standing far away in a park, smaller kernel sizes like 3 x 3 might have been necessary to produce good generalization.

ANN Using Weights Transferred From VGG

VGG16 is a famous convolutional neural network architecture proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper “Very Deep Convolutional Networks for Large-Scale Image Recognition”. The model has been shown to be highly effective at extracting features from images. VGG16 has a total of 13 feature extracting convolution layers followed by 3 dense layers. With our transfer learning approach, we imported VGG’s 13 convolution layers and made them untrainable. i.e froze them. By doing so, we locked in the weights of 14,714,688 parameters in those layers. VGG had learnt those millions of weights after weeks of training on over 14 million images on the ImageNet dataset. Thereafter, we created new features by running our training images through this frozen stack. The final architecture is laid out below:

Feature Extraction Layers

Layers 1 to 13 - Transferred from VGG16

Regression Layers

Layer 14 - Flatten Layer
Layer 15 - Dense layer with 30 neurons each activated by ‘relu’ function
Layer 16 - Dense layer with 10 neurons each activated by ‘relu’ function

Layer 17 - Dense layer with 1 neurons activated by ‘linear’ function

Support Vector Regression

For our Support Vector Machine Regression (SVR) we used the following features: 500 BOVW topics, ethnicity variable, gender variable. We conducted a grid search over different kernels such as rbf, linear, and polynomial. As well as tested C regularization at 1.0 and 0.5. SVR with a RBF kernel and regularization of 1.0 was the best performing ML model with a RMSE of 0.56. This is especially notable when comparing the performance to our CNN approaches and the intraclass classification challenges that arose from our BOVW feature engineering approach. We mention our insights and evaluation of the features engineered for this model in unsupervised learning and the discussion section.

Random Forest Regression

For our Random Forest Regression (RFR) we used the following features: 500 BOVW topics, ethnicity variable, gender variable. We conducted a grid search over max tree depth, the number of estimators, and minimum sample splits. Our best performing RFR had a max_depth of 30, minimum sample split of 10 and leveraged 350 estimators and was not particularly sensitive to the tuning range we explored in the grid search, which made us concerned about our features. This model had an RMSE of .60. We mention our insights and evaluation of the features engineered for this model in unsupervised learning and the discussion section.

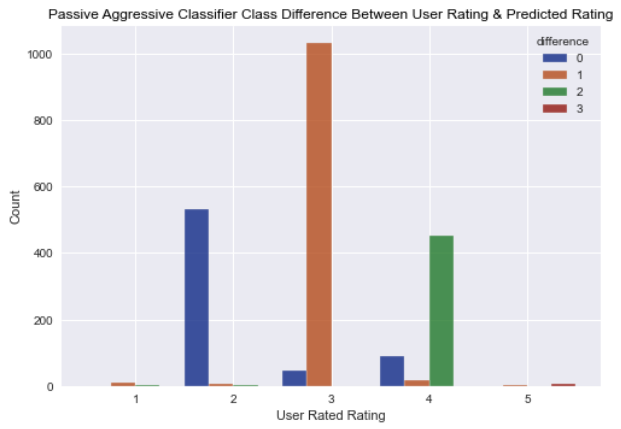
Passive Aggressive Regression & Classification

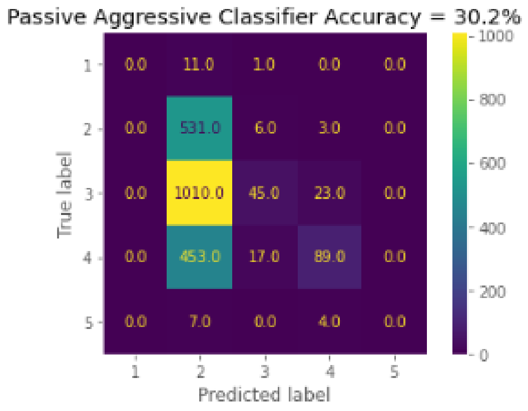
We explored Passive Aggressive models for two reasons: **(1)** to explore how training under streaming conditions affected our model performance; **(2)** training with 245k features required batch training i.e. partial fitting due to hardware constraints. We trained in batches of 250 images and the only parameter tuning we explored was warm starts. When warm starting, the existing fitted model attributes are used to initialise the new model in the subsequent call to fit. In warm starts parameters are expected to change and the data is (more-or-less) constant across calls to fit. Whereas with partial fit, the mini-batch of data changes and model parameters stay fixed. While these models had the worst performance, we did not realize the substantial shift in scoring that we expected when testing cold start versus warm start. This is notable because each batch did not explicitly include samples from each class, which is considered best practice when using warm start. This alone made us question the effectiveness of our PCA feature engineering. We mention our insights and evaluation of the features engineered for this model in unsupervised learning and the discussion section.

Model	Features	RMSE on test set
Dummy Regressor	Arithmetic Average	0.69
Shallow CNN	2 Convolutional layers	0.46
Deep ANN Using Transfer Learning	13 Convolutional layers transferred from VGG16	0.44
Support Vector Regressor	500 BOVW topics, gender variable, ethnicity variable	0.56
Random Forest Regressor	500 BOVW topics, gender variable, ethnicity variable	0.6
Passive Aggressive Regressor	500 BOVW topics, 2 PCA components composed of 41 filters, gender variable, ethnicity variable	1.33
Passive Aggressive Classifier	500 BOVW Topics, 2 PCA components composed of 41 filters, gender variable, ethnicity variable	Accuracy (.30)

Failure Analysis

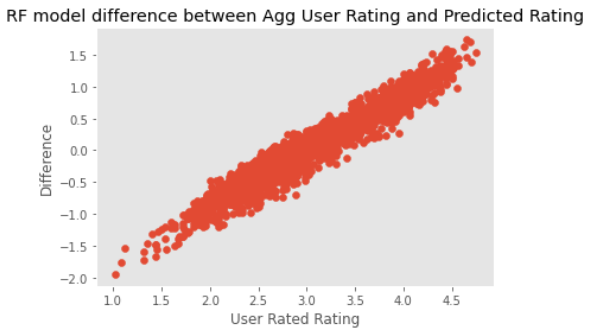
Reviewing our Passive Aggressive Classifier, which was trained on the BoVW, 2 PCA components, gender, and ethnicity feature, we found that it defaulted to predicting 2 as a label, which is surprising given that the mean label is 2.99. As the charts show below, this created 1010 false predictions of 2 for a user rating of 3 and means that our feature representation did not have a positive impact on the model.

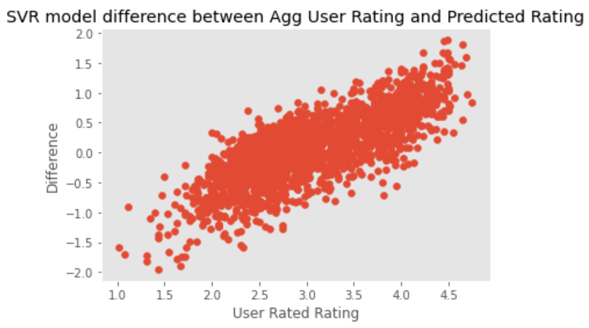




	1	2	3	4	5
1	0.0	11.0	1.0	0.0	0.0
2	0.0	531.0	6.0	3.0	0.0
3	0.0	1010.0	45.0	23.0	0.0
4	0.0	453.0	17.0	89.0	0.0
5	0.0	7.0	0.0	4.0	0.0

Our SVR and Random Forest models trained on the BoVW features, ethnicity, and gender showed a linear trend line that correlated with the user rating thus indicating that it defaulted to predicting the mean. This indicates that our BoVW features did not perform well at distinguishing intra-class differences.





Discussion

Overall

There was a significant computational and time advantage to the CNN approach. Conducting “manual” feature representation in the form of convolutions, reducing those convolutions into PCAs, and or creating BoVW took a significant amount of time and our models probably reduced the feature space during training such that some of the process had negligible impact on the model. Moreover, the machine learning pipeline humbly indicated that our manual feature representation did not have a strong positive impact on the task performance.

Unsupervised Learning

While the BoVW feature representation method seems to perform rather well at tracking and distinguishing between separate class categories we found that intraclass classification is a much harder problem to solve. When we mapped cluster keypoints on images we found that they routinely weren’t referencing the same feature of a person. Instead they seemingly were matching patterns at various pixel scales and as a result they were jumping between various image features. This undermined the intent of these variables — distinguishing facial features. This is remarkably humbling when one considers that a CNN automatically generates similar features and navigates between the pixel scales to provide a closer classification.

Our intuition tells us that the BoVW feature representation probably follows Power’s law. We wonder if the most frequent clusters, such as the seemingly unremarkable corners of the eye, are negatively impacting our clustering. We only analyzed and modeled a single class, human’s headshots, as such we didn’t model intraclass feature importance during our feature engineering. We relied on supervised classification to extract that weighting. Thus, our feature engineering could probably benefit from frequency-inverse image frequency weighting. Perhaps, an algorithmic way to approach this weighting biases would be to model each rater’s weighting and biases and then predict individual ratings. It is our impression that aggregating scores confuses the weighting and therefore yields a larger mean squared error that captures greater general rater variance.

Extending Unsupervised Learning

If we had more time we would explore modeling intra-class classes in our BoVW feature, although that might lead to ethical concerns in the subsetting. It is still our intuition that modeling the semantics of facial landmarks is the crux of this problem and improving feature representation in this manner would improve model performance. Additionally, we would explore what is the minimum ideal combination of convolution layers that refines a face to its “bare” geometric representation. For instance, we wonder if we could pool Prewitt, Scharr, Sobel and Roberts into a single principal component? This reduction of convolution is particularly important given that our shallow CNN, with less convolutional layers, performed better than the deeper CNN.

Ethical Considerations for Unsupervised Learning

Ethical issues could arise during unsupervised classification, the clustering phase, if clusters of features yielded biases formed based on a particular skin color or gender or ethnicity. In our approach, we made sure the descriptors (feature vectors) were clustered on the entire dataset as opposed to multiclass clustering (i.e. finding features from only Asians or only Caucasians or only males or only females). This ensured that there was no systemic racial or gender bias in terms of how a group was discriminated against. The 500 visual topics were global in nature.

Supervised Learning

Surprisingly our shallow CNN generalized better than *deep* ANN built using transfer learning. Transfer learnt ANN *suffered from overfitting*. In that, its RMSE training was 2x better than on the test set. We reason that could be due to the huge difference in the number of trainable parameters/weights between the two models. The feature extraction layers of VGG16 that we transferred to our ANN amounted to over **14 million convolution weights** compared to our shallow CNN which had only two feature extraction layers amounting to just **15,000 convolution weights** to learn. Also, the final convolution layer of VGG produced 51,200 weights/features whereas the final convolution layer of our shallow CNN produced just 5,000 weights.

Extending Supervised Learning

If we had more time and resources, we could have tried harder to beat the benchmark RMSE of 0.33 published by the authors of the dataset. Our best RMSE was 0.46. One way to do that would be to improve the regularization of transfer learnt ANN by adding multiple drop out layers. We would also like to do a root cause analysis on why the Passive Aggressive Model did not perform to our expectations.

Ethical Considerations for Supervised Learning

We foresee that making the results of facial beauty prediction models available to the users could create ethical issues e.g. If a user came to know they got a low score of 1.5, they may try to game their profile images or it might cause psychological trauma to them. Secondly, our model learns to generalize based on the perceptions of 60 volunteers and then would filter the users recommendations accordingly. This essentially would create an information bubble that does not represent any particular user's subjective interests. It would be more ethical to deploy a model that could learn a user's preferences and filter the information provided to them accordingly. Furthermore, if the raters that trained the model were all biased against a subset of the population and gave them low beauty scores then our model would become biased as well during the training. This can be analyzed using Google's What-If-Tool.

Statement of Work

Aditya Patel, led the tensor flow and neural network coding portion of the project. Chris Westendorf, led the machine learning coding of the project. We collectively made decisions about methodology, feature representation, and architecture through project status meetings. After the coding of the project we critically reviewed our approaches and the results. We then shared in communicating our methodology and results in this final report.

We collaborated through github: **https://github.com/adityahpatel/milestone2_dating_and_beauty**