Beauty Is Only as Deep as the Neural Network: Predicting the Beauty of Flickr Images

Shikhar Sunny Gaur
Department of
Management Sciences
University of Waterloo
Waterloo, Canada
Student Number: 20522438
Email: s2gaur@edu.uwaterloo.ca

Andrej Rosic
Department of
Nanotechnology Engineering
University of Waterloo
Waterloo, Canada
Student Number: 20517184
Email: arosic@edu.uwaterloo.ca

Rui-Yi Julia Zhang
Department of
Systems Design Engineering
University of Waterloo
Waterloo, Canada
Student Number: 20520196
Email: rjzhang@edu.uwaterloo.ca

Abstract—**Is beauty subjective or objective? An attempt to answer this age-old question is made by assessing whether a machine can not only learn about the human notion of beauty, but also predict it using machine intelligence. In this paper, a large dataset of images ranging in beauty and categories is used under the Flickr Creative Commons License to train and test a machine's understanding of beauty. Multiple feature extraction techniques are used to extract attributes of each image such as texture, shapes and patterns and then used to train machine learning models using different regression techniques, to understand and predict beauty. The best performing method uses support vector regression trained on deep features extracted from images by a pre-trained Convolutional Neural Network. This method outperforms previous attempts described in literature by a maximum of 94.1%, providing an argument that beauty can perhaps be objective.**

***Keywords*—beauty, computer vision, image processing, local binary patterns, deep feature extraction.**

I. Introduction & Background

The notion of beauty is a primordial theme that philosophers, artists and scientists have discussed for centuries. It is a primary theme among ancient Greek, Indian, Hellenistic cultures while also being a central topic of philosophical inquiry. One of the most recurring issues in the theory of beauty is whether beauty is subjective, or whether it is an objective feature.

Many philosophers, writers and historians believe in the subjective nature of beauty, that it lies in the eye of the beholder. While Margaret Wolfe Hungerford is found to be the earliest citation to write "beauty is in the eye of the beholder" in her book *Molly Bawn*, 1878 [1], the notions that beauty is subjective can be found throughout history. Shakespeare, one of the most influential English poets and writers, expressed this sentiment in *Love's Labours Lost*, 1588:

"Good Lord Boyet, my beauty, though but mean,
Needs not the painted flourish of your praise:
Beauty is bought by judgment of the eye,
Not utter'd by base sale of chapmen's tongues" [2]

Similarly, David Hume, a Scottish philosopher, displayed his subject belief on beauty in his essay *Moral and Political*, 1742: "Beauty is no quality in things themselves: it exists merely in the mind which contemplates them; and each mind perceives a different beauty" [3]. On the contrary, if beauty is deemed entirely objective and not connected to a subjective response, this would imply that in a world with no perceivers, things could still be beautiful themselves. The classical conception of beauty is that beauty consists of an arrangement of integral parts into a coherent whole, according to proportion, harmony, symmetry and similar notions [4]. Aristotle says in *Poetics* that "to be beautiful, a living creature, and every whole made up of parts, must ... present a certain order in its arrange met of parts", and in *Metaphyiscs* that "the chief forms of beauty are order and symmetry and definitiveness, which the mathematical sciences demonstrate in a special degree" [5]. Aristotle implies that there are some mathematical features that can be beautiful when combined and arranged properly. In the current digital age, there is now an abundance of data to further investigate Aristotle's beliefs. This notion of determining beauty by decomposing the object in question and understanding its features was tackled through the use of computer vision in this paper. Approximately 15,000 images were retrieved using Flickr, a third party image- and video-hosting website. The dataset contains images, category of the image, and beauty scores crowdsourced by individuals. The individual beauty scores capture the subjective nature of beauty, while the machine learning algorithms objectify notions of beauty by quantifying attributes of each image.

II. Related Work

There is existing research done in topics related to beauty, such as predicting popularity and internet virality, many of which can be relevant to classifying and predicting beauty or aesthetics, as one would expect that beautiful pictures would gain popularity. Research into predicting popularity often relies on external metadata such as the photographer's social information and popularity [6]. However, research done into classifying or defining beauty is often limited in scope, particularly on the topic of beautiful people, or scenic landscapes. For example, symmetry in faces is often a high indicator of beauty.

In terms of work done in the machine intelligence community, there is limited research on defining beauty of an image purely based on the raw image itself. One method that has previously been used is an approach that uses eigenfaces [7]. However, this method is only useful for facial beauty, and not for other types of images. Another method that is used in a paper that classifies facial beauty is a convolutional neural network that uses multiple downsampling/convolution layers to extract features. In this paper, a model with many layers is more successful than the single and two-layer models that were also attempted [7].

Another paper that attempts to predict facial beauty uses a variety of feature extraction methods with different regression methods. For feature extraction, the four different methods were active shape models (ASMs) to find facial landmarks, active appearance models (AAM), local binary patterns (LBP), and PCANet (a deep learning network for image classification). The methods that were used for regression were multivariate linear regression, and support vector regression (SVR). The combination that performed the best was PCANet, along with SVR [8]. Although this paper focuses specifically on the task of predicting facial beauty, these methods can be applied to feature extraction and regression of generic images as well.

There is one paper that has a similar goal of classifying the beauty of images, that predicts aesthetic image tags, such as "beautiful", "wonderful", "divine", "awful", "ugly", "terrible". Images were taken from Flickr, from the *most interesting* category, and that were tagged with specific adjectives. In this paper, feature extraction is done on only the image, with no other metadata included. Some features include hue, saturation, and value of the image, position of the main image, and colorfulness. The method used for classification in this paper is support vector machines (SVM), and was fairly successful with an accuracy of 88% when classifying whether the image had positive or negative connotations [9].

The dataset used for this project is from previous work done by Schifanella et al., and the dataset itself is described in more detail in the next section. One of the goals of Schifanella et al. was to find beautiful pictures, regardless of popularity. Thus, their model, CrowdBeauty, was purely based on the image itself, and all the features identified for the model was based on the image rather than including other metadata such as popularity. Their hypothesis was that there were images that would be considered beautiful, hidden among photos that did not receive a lot of attention. This hypothesis was verified after they had collected their data by gathering images with varying popularity and using crowdsourcing to assign beauty scores to each image. The goal of their model was predicting a beauty score, based on only the image itself, which has potential to be used to find beautiful images easily, without relying on social media popularity [10].

The features that Schifanella et al. used encompassed three broad categories: color features, spatial arrangement features, and texture features, with many of the features based on prior research on aesthetics of images. Colour features included contrast, hue, saturation, brightness, as well as using these features in order to generate emotional dimensions such as pleasure, arousal and dominance. Spatial arrangement features included symmetry and rule of thirds, which are well-known photography principles, and texture features included Haralick's texture features - entropy, energy, homogeneity and contrast [10].

The method that was used for learning was partial least squares regression, and was specific to the image category. The performance of the CrowdBeauty model was evaluated using the Spearman Correlation Coefficient between scores predicted by CrowdBeauty and the crowdmarked scores, which gauges the ability of the model to predict the ranking of the images in terms of the beauty score. The model was also compared against two other models backed by academic research, but are predicting different types of information, which is why the Spearman correlation was used as the evaluation method. The first was the MIT popularity predictor, which predicts image popularity scores in terms of views. The second model that CrowdBeauty was compared against was a model with the same features as the CrowdBeauty model, but trained on traditional aesthetic datasets with images from semi-professional photographers. The dataset is from AVA, which is a large scale database for Aesthetic Visual Analysis. Compared to the MIT popularity predictor, and the model trained with traditional aesthetic images, the CrowdBeauty model outperformed the other two in terms of the Spearman correlation. The results from the Schifanella et al. paper and other evaluated methods are shown in Table I. A random baseline is also included in the results for comparison purposes, and to demonstrate that the Spearman correlations have significance.

TABLE I: Model Results from Schifanella et al. [10]

	CrowdBeauty	**Popularity**	**Traditional Beauty**	**Random**
animals	0.54	0.37	0.251	0.001
urban	0.46	0.27	0.12	0.003
nature	0.34	0.29	0.11	-0.003
people	0.42	0.31	0.27	-0.008

Although the same dataset is used for this paper, the goal is not to reproduce the work of Schifanella et al., but rather to attempt different feature extraction methods and machine learning methods to characterize beauty in images.

III. Available Data

The "CrowdFlower" dataset provided by Schifanella et al. relates to 15,686 images obtained under the Flickr Creative Commons License. Each record in the dataset contains a Flickr photo ID, a category, and a set of beauty scores. The Flickr photo ID allows the image to be retrieved via Flickr's API. Four categories of images were included: people, animals, nature, and urban. An image's category was determined based on the Flickr machine tags that were associated with the image. Photos in each category were then separated into three classes based on how many times the photo was favourited: Tail ($<$ 5 favourites), Torso (5 $<$ favourites $<$ 45), and Head ($>$ 45 favourites), for comparison and analysis purposes. Finally, a set of beauty scores was obtained through the crowdsourcing experiment, named CrowdFlower, that Schifanella et al conducted. In this experiment, each image was shown to at least five independent human judges. To rate the aesthetic nature of the image, each judge would assign an integer score between 1 and 5. The criteria for each "beauty score" is provided in Table II.

To illustrate the available data, Table III summarizes each column in the dataset. It also provides an example of each field, for a representative sample in the dataset.

Note that while the majority of images (74.3%) were assigned five beauty scores during CrowdFlower, some images

TABLE II: "Beauty Score" Criteria [10]

Beauty Score	Descriptor	Criteria
1	Unacceptable	Extremely low quality, out of focus, underexposed, badly framed images.
2	Flawed	Low quality images with some technical flaws (slightly blurred, slightly over/underexposed, incorrectly framed) and without any artistic value.
3	Ordinary	Standard quality images without technical flaws (subject well framed, in focus, and easily recognizable) and without any artistic value.
4	Professional	Professional-quality images (flawless framing, focus, and lightning) or with some artistic value.
5	Exceptional	Very appealing images, showing both outstanding professional quality (photographic and/or editing & techniques) and high artistic value.

TABLE III: CrowdFlower Dataset Description

Column Name	Description	Example
flickr_photo_id	ID for retrieval using Flickr API	3621844364
category	Semantic category based on Flickr Machine Tags	animals
beauty_scores	Beauty scores assigned by CrowdFlower judges	4,5,5,4,5

were assigned several more. Of the 4,027 images with more than five beauty scores, most had either six scores (30%) or ten scores (50%). Since the majority of subsequent analysis focuses on the average beauty score assigned to each image, this discrepancy was considered negligible and was ignored.

The first step in this investigation was retrieving the images paired to the provided photo ID using the Flickr API. Of the 15,686 images in the CrowdFlower dataset, only 14,869 (94.8%) could be successfully downloaded using the Flickr API. The remaining images were likely deleted since the dataset's release. Though missing images affected the "people" category nearly twice as much as other categories, the proportional representation of each category did not change by more than 1%. Overall, the missing images did not drastically impact the dataset. Thus, all subsequent analysis focused exclusively on the 14,869 available images.

A. Exploratory Data Analysis

Before attempting to extract features from the downloaded images, it was important to understand the nature of the data without making any assumptions. An initial exploratory data analysis was conducted to visualize and understand the tendencies of the data.

The data retrieved from Flickr was semi-structured. Although the images are unstructured, there are relationships between each image and its category and beauty scores that adds some structure to the data. Each image only belongs to one of these four categories. The four categories are quite evenly distributed, despite being unable to retrieve a few images. 24% of the images are categorized as animals, 30% as nature, 23% as people and 23% as urban. Each image was also given five or more beauty scores. These numerical attributes and their relationship to each category were explored next. The distribution of the average beauty score for each category is shown in Figure 1. The mean of average beauty scores for the animals category was the highest with an average beauty score of 3.5 while having a large standard deviation. Conversely, the standard deviation of beauty scores for nature images were the lowest, concentrated around 3.25 average beauty scores. The beauty scores were found to be normally distributed around 3, as expected, slightly skewed left.

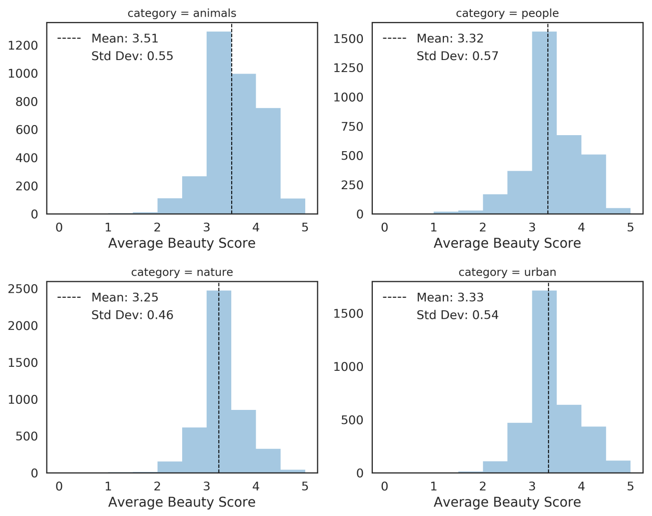

Fig. 1: Average beauty scores per category.

Another breakdown of the average beauty score per category is shown in Figure 2 in the form of a box plot. The box plot provides insight into the spread of scores and their relationship with each category. Animals had a higher median beauty score, of approximately 3.6, than the rest of the categories that center around 3.2, similar to mean. This similarity between mean and median implies that the data is fairly symmetrical. Only animals median is visibly greater than mean, which further shows that the distribution is slightly skewed to the left. In addition, the people category has a large interquartile range, which shows that around 50% of the values are spread over 3.0 and 3.8 approximately, while the other categories of urban and nature are slightly more concentrated between 3.0 and 3.6.

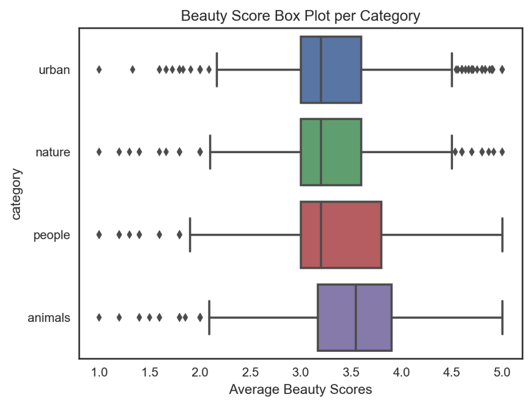

Fig. 2: Box plot of average beauty scores per category

Lastly, the image properties were investigated. Figure 3 shows a density scatter plot of the image dimensions in pixels. The plot shows high density at width = 500 and height =

333, and similarly at width = 333 and height = 500. This provides insight that majority of the images are either landscape (500x333) or portrait (333x500) and not square images. The max value for both height and width is 500 and 96.8% of the images are either 500 in height or width. The reason for this is that when downloading images from the Flickr API, the default image size is set to *Medium 500*, where the maximum width or height of the image is 500 pixels.

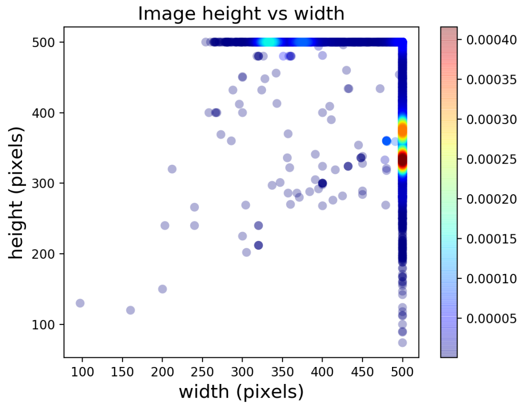

Fig. 3: Image shape density scatter plot

IV. Methods

A. Feature Extraction

As described previously, Schifanella et al. manually generated a 47-dimensional feature vector encompassing colour, spatial arrangement, and texture characteristics. In this paper, features were automatically extracted from the images using two different techniques: local binary pattern (LBP) texture analysis and deep feature extraction.

Local Binary Patterns (LBP). LBP was first introduced in 1996 by Ojala et al. as a method for characterizing texture in images [11]. In this method, the LBP operator is iteratively centered at a every pixel in the image, and used to threshold the eight neighbouring pixels. Starting with the upper-left neighbour and moving clockwise, the eight thresholded pixels each represent a bit in a collective binary number that is assigned to the center pixel. Since this binary number is therefore comprised of eight bits, the range of values returned by the LBP operator is (0, 256). Figure 4 shows an example of the LBP operator.

In 2002, the same authors enhanced the method to provide multiresolution LBP analysis, rather than restricting to the eight pixels directly neighbouring each center pixel [13]. To do so, P points are selected evenly on the circumference of a circle of radius R. If a selected point does not fall exactly at the center of a pixel, its value is locally interpolated based on nearby pixels. In this way, textures of varying scale can be characterized. Figure 5 shows examples of the circular neighbourhoods selected for various values of P and R.

Although the enhanced method allows for various values of P and R to be used, the method for computing the LBP value is the same. The P neighbouring points are thresholded

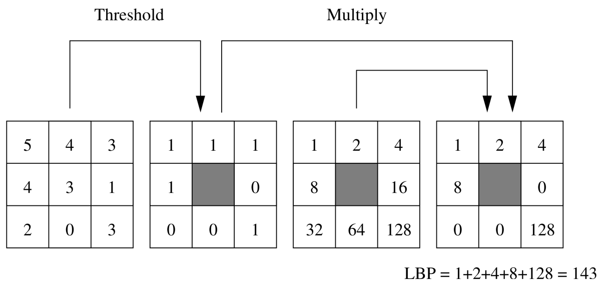

Fig. 4: The LBP operator applied to a selected image region (left). The neighbouring pixels are thresholded by the value of the center pixel (middle-left). The binary bit positions assigned to each neighbour (middle-right) multiply the thresholded values in order to compute the final LBP value between (0, 256). Adapted from Maenpaa et al. [12].

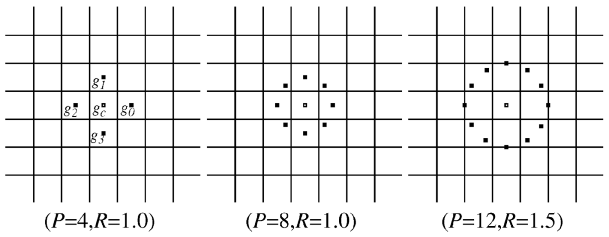

Fig. 5: Various "circular neighbourhood" geometries for LBP. Adapted from Ojala et al. [13].

by the center pixel and together form a P-bit LBP value that is assigned to the center pixel. In addition, Ojala et al. proposed that P circular bit-wise shifts of the P-bit LBP value should be performed, and that the minimum value in this set should be assigned to the center pixel, so that rotational invariance could be achieved [13]. However, the rotationally variant operator was ultimately used for feature extraction in this report.

Since the LBP operator can be centered at every pixel in an NxM image, the resulting LBP values will be an NxM matrix. When $P = 8$, every element of this matrix will be between 0 and 256. Thus, to convert the matrix to a one-dimensional feature vector that describes the LBP textures, a 256-bin histogram is generated. The method to generate feature vectors using LBP is described in Algorithm 1. Note that the `local_binary_pattern` function in Python's Scikit-image library was used to perform LBP analysis.

Algorithm 1: LBP Feature Extraction Algorithm

```
input : A set Q of k images of size $N_k \times M_k$
output: A $k \times 256$ matrix F of feature vectors.
1 i ⟵ 0
2 foreach image $X_{N_k,M_k}$ of the image set Q do
3     $L_{N_k,M_k} \longleftarrow$ lbp ($X_{N_k,M_k}, P = 8, R = 1$)
4     $f_{1,256} \longleftarrow$ histogram ($L_{N_k,M_k}, bins = 256$)
5     F[i, :] ⟵ $f_{1,256}$
6     i ⟵ i + 1
```

To demonstrate this feature extraction technique, the method was applied to an image in the CrowdFlower dataset. Figure 6 shows the LBP image that is generated by LBP analysis with $P = 8$ and $R = 1$ on the original image. The LBP histogram constitutes a 256-element feature vector for the image. Subsequent regression techniques will focus on learning to predict beauty scores from these histogram feature vectors.

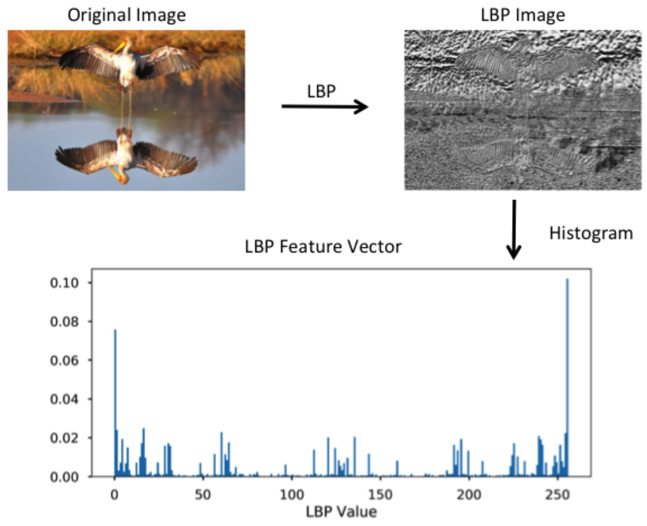

Fig. 6: LBP analysis applied to a CrowdFlower image.

Deep Feature Extraction. Over the past few years, deep Convolutional Neural Networks (CNNs) have achieved great success in the field of computer vision, specifically in image classification and object recognition tasks [14]. CNNs have much fewer connections and parameters than standard feed-forward neural networks of similar size, which makes them much easier to train and only reduces theoretical performance slightly [15]. Python's deep learning library Keras provides access to nine different CNNs with weights that have been pre-trained on the popular ImageNet dataset [16].

Pre-trained models can be used for three different purposes on new images: prediction, feature extraction and fine-tuning [16]. In this paper, only feature extraction is explored, using two of the nine available pre-trained CNNs in Keras: ResNet50 (25,636,712 parameters) and InceptionV3 (23,851,784 parameters) [16]. The goal of feature extraction using CNNs, like in LBP, is to extract a feature vector for each image that can subsequently be used to train models for predicting beauty scores. This process, known as "transfer learning", where a CNN trained on images for one application is used to extract features from images related to another application, has been shown to be a promising method for feature design [14]. This method has been shown to achieve better results than hand-crafted features, like the ones used by Schifanella et al. [17].

Documentation for the Keras `Applications` module indicates that a feature vector can be extracted from a pre-trained CNN at any arbitrary layer of the network. However, a common choice is to remove the fully-connected layer at the top of the network (which performs classification) and instead output activations from the last convolutional layer [17]. The last convolution layer is chosen because later network layers typically represent higher level features of the image (e.g. shapes, patterns) than earlier network layers which represent more simple features (e.g. edges). It would be worthwhile however, for subsequent research to investigate the performance achieved when earlier layers are used instead.

In this paper, average pooling is applied to the output of the last convolution layer. This ensures that the network returns a vector rather than a tensor for each image. Max pooling can also be used for this purpose, and further research should assess whether it can lead to improved performance over average pooling. When ResNet50 (ResNet) and InceptionV3 (Inception) are modified in this way, and used to predict the class of a particular image, they will return a 2048-dimensional feature vector rather than a classification label. This can be repeated for all k images in the CrowdFlower dataset.

It is worth noting that 2048-dimensional vectors are quite large for the regression techniques that are explored in this paper. Though they might ultimately perform well, training models on such high-dimensional data would be computationally expensive. To reduce the computational complexity, a dimensionality reduction algorithm such as principal component analysis (PCA) can be used to reduce the number of dimensions whilst maintaining a threshold percentage of the variance in the data. For visualization purposes, the dimensionality reduction technique t-SNE can be used to assess whether images with similar beauty scores have deep feature vectors that are close in high-dimensional space. Figure 7 shows a t-SNE visualization of the ResNet features generated for the 'animals' category. The points are coloured by the half-score bin that the image's average beauty score falls in. Though we ultimately intend to perform regression, assigning the beauty scores to bins make the visualization easier to interpret.

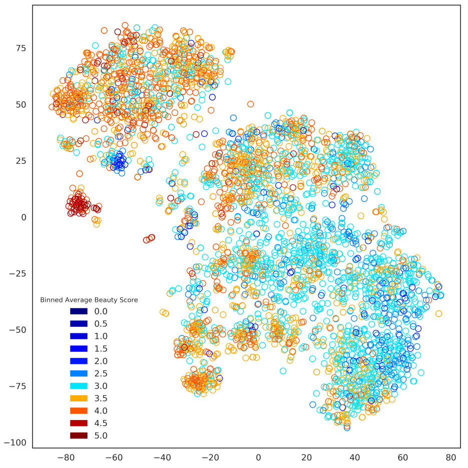

Fig. 7: t-SNE visualization of ResNet feature vectors in the 'animals' category.

The most beautiful images cluster tightly above the legend in dark red. In addition, more beautiful images (score $\geq$ 4.0) seem to cluster in the top-left, and less beautiful images (score $\leq$ 3) seem to cluster in the bottom right. The middle region includes images over a wider range of scores. This visualization is encouraging because it suggests that, in the high-dimensional deep feature space, more beautiful images are closer to one another than to less beautiful images.

B. Regression Techniques

In this paper, regression techniques were used instead of classification. The main reason for this choice was that Schifanella et al. performed regression in their paper. Since their goal was to rank each image in the dataset according to the predicted beauty scores, classification would be insufficient. All images assigned to the "most beautiful" class would be considered equally beautiful, and arbitrarily surfacing the top n most beautiful images would not be possible. By predicting beauty scores continuously via regression, Schifanella et al. were able to rank each image from most beautiful to least beautiful with high granularity.

Another reason for this choice was to be able to measure model performance using the Spearman Correlation Coefficient and benchmark it against the results achieved by Schifanella et al. (shown in Table I). At a high level, Spearman correlation measures the degree to which the rank ordering of two sets of measurements is the same. Although this metric could theoretically be used to measure performance in both classification and regression, it cannot be fairly compared between the two settings. This is because, in classification, all images assigned to the same class would be considered rank-equivalent. Regression cannot benefit from intra-class rank equivalence, and so there is much greater opportunity for error. This nuance is especially significant in the CrowdFlower dataset, where the vast majority of images have an average beauty score between 3 and 4 (Figure 1). As a result, to fairly compare model performance to the regression results reported by Schifanella et al., a regression approach is necessary in this paper as well. Specifically, two separate regression methods were used to predict beauty scores: support vector regression (SVR) and random forest regression (RFR). Both regression methods were used on all three of the different feature extraction methods.

Support Vector Regression (SVR). SVR is a supervised regression method based off of support vector machines (SVM). SVR works by constructing a hyperplane, or set of hyperplanes that maximizes the margin of tolerance, ϵ, where the error within the margin is tolerable, and minimizes error outside of the margin. Within the margin of tolerance, there is no penalty associated with the error. In general, the larger the margin, the lower the generalization error. For regression, SVR can be written as an optimization problem. When it is not feasible to solve the optimization problem given the margin of tolerance constraints, additional slack variables are added to the minimization problem to minimize error outside of the margin, similar to a soft margin loss function [18]. The optimization problem is shown in Equation 1, with the soft margin, where C is the penalty of the error term.

$$\begin{aligned} \text{min} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}(\zeta_i + \zeta_i^*) \\ \text{s.t.} \quad & y_i - \mathbf{w}\cdot\mathbf{x_i} - b \leq \epsilon + \zeta_i \\ & \mathbf{w}\cdot\mathbf{x_i} + b - y_i \leq \epsilon + \zeta_i^* \\ & \zeta_i, \zeta_i^* \geq 0, i = 1, ..., m \end{aligned} \tag{1}$$

A visual representation of SVR including the soft margin errors, ζ, can be seen in Figure 8.

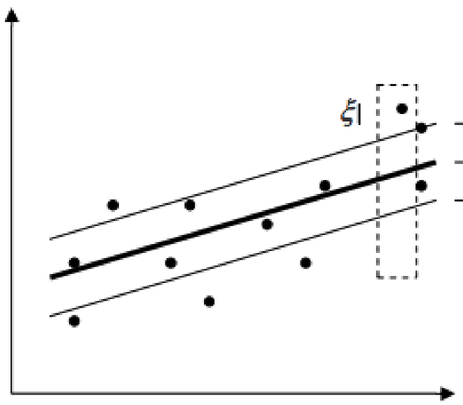

Fig. 8: Visual representation of SVR highlighting the margin of tolerance and the soft margin [18]

SVR can also be used for non-linear cases, by using a kernel function to map the data into a higher dimensional space, where the data can then be linearly separable. A common kernel that is used is the radial basis function (RBF), which can be seen in Equation 2, as well as polynomial functions and sigmoid functions. When using different kernels, different parameters (i.e. σ in the RBF kernel) can be used depending on what best linearizes the data.

$$K(x, y) = exp(-\frac{1}{2\sigma^2}\|x - y\|^2) \tag{2}$$

Random Forest Regression (RFR). Random Forest is an ensemble learning method that can be used for both classification and regression, and works by constructing many decision trees on a random subset of features and samples, and then aggregating all of the trees. Since individual deep decision trees are prone to overfitting, random forest averages many deep decisions trees for better generalization. RFR works by first drawing k sets of samples, with replacement, where k is also the number of decision trees. Each tree is trained by randomly splitting features at each node, and choosing the best split among those. Because of the randomness in RFR, results vary slightly every time RFR is run. For regression, the final prediction is the mean value of the predictions from all of the trees [19]. A visualization of how RFR works is shown in Figure 9.

For both techniques, their respective implementations in Python's Scikit-learn were used. In addition, Scikit-learn's `GridSearchCV` module was used to optimally select the hyperparameters for each model, by iteratively fitting models with different parameters and evaluating performance using k-fold cross-validation.

C. Measuring Performance

Schifanella et al. aimed to surface the most beautiful images within four image categories. Since they intended to train separate models for each category, their first step was to split each category into a training set and a test set. For each category, the regression models were trained on the images in the training set, and then the trained models were used to

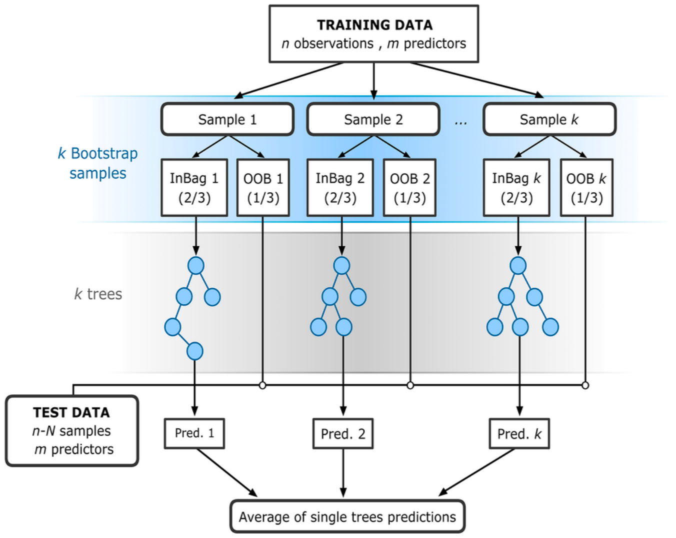

Fig. 9: Visualization of RFR [20]

predict beauty scores for the images in the test set. To evaluate how well their regression models performed, Schifanella et al. used the Spearman Correlation Coefficient. Specifically, they measured the Spearman correlation between the predicted beauty scores and the corresponding "true" beauty score (i.e. average of crowd-sourced ratings) within the test set [10]. Their results were presented in Table I.

Spearman correlation is a good performance metric for this application. To identify the n most beautiful images in a test set, it is most important that the predicted scores correctly order the images from most-to-least beautiful. As described previously, the Spearman Correlation Coefficient measures the strength and direction of the monotonic relationship between two variables [21]. Consider a set of predicted beauty scores $\hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$ and "true" beauty scores $Y = \{y_1, y_2, ..., y_n\}$. The Spearman correlation r_s between $\hat{Y}$ and Y is identical to the Pearson (i.e. linear) correlation ρ between the rank values of $\hat{Y}$ and Y [22]. That is, each predicted beauty score $\hat{y}_i$ can be assigned a rank r_i, indicating that image was predicted to be the r_i'th most beautiful in the set. The same can be done for the true beauty scores in Y. The linear correlation between the rank value sets, $R_{\hat{Y}}$ and R_Y, is therefore equal to the Spearman correlation between $\hat{Y}$ and Y, as expressed in Equation 3 [23].

$$r_s = \rho_{R_{\hat{Y}}, R_Y} = \frac{cov(R_{\hat{Y}}, R_Y)}{\sigma_{R_{\hat{Y}}} \sigma_{R_Y}} \tag{3}$$

If the predicted beauty scores ranked the images the same as the "true" beauty scores (i.e. $R_{\hat{Y}} = R_Y$), the Spearman correlation between the two sets would be 1.0. If the predicted beauty scores reverse-ranked the images, the Spearman correlation with the actual beauty scores would be -1.0. As a result regression techniques should aim to maximize the Spearman correlation between the predicted beauty scores and the "true" beauty scores.

V. Results

As mentioned earlier, three different methods of feature extraction were performed on the images: LBP, Deep (Inception), and Deep (ResNet). LBP features were successfully extracted from all of the images, resulting in 14,869 LBP histogram feature vectors, each with 256-dimensions. However, when deep feature extraction was performed, one image failed to generate features when using Inception, and 35 images failed when using ResNet. The ResNet failures occurred because one dimension of the image was too small to be processed by the model. These images were excluded from the training and testing sets for their respective methods. Since at most 0.24% of images are affected by these failures, the performance the feature extraction techniques can still be fairly compared.

Each category was trained separately, since that was how Schifanella et al. trained their CrowdBeauty models. Thus, the images in each category were split into training (66.66% of category images) and test sets (33.33% of category images). Table IV shows the size of the training and test sets for each category. Note that due to the 35 missing ResNet images, 11 images that are in the LBP test set are not in the ResNet test set, and subsequently 24 images were not trained on in the ResNet test set. The missing image from the Inception features was in the training set, so the Inception training set is missing an image and the Inception and LBP test sets are identical. In comparison, the test sets that Schifanella et al. used were 800 images each, with a larger training set size than used here.

TABLE IV: Number of Images Assigned to Training and Test Sets Per Category

Category	Total Images	Training Set Size	Test Set Size
Animals	3541	2372	1169
Nature	4471	2995	1476
People	3368	2256	1112
Urban	3489	2337	1152
Total	14869	9960	4909

Before training the models, PCA was performed on deep features to attempt to reduce the dimensionality of the feature vectors in order to speed up training. This was done by using Scikit-learn's `PCA` module, and was set to reduce to a number of components such that 95% of the original variance is retained. Prior to PCA, the original deep feature vectors for both Inception and ResNet had 2048 components. For each deep feature method, the number of components after PCA in each category are shown in Table V.

TABLE V: Number of Components after PCA Dimensionality Reduction

	Animals	Nature	People	Urban
Inception	47	45	30	55
ResNet	579	412	545	473

A separate model was then trained for each category. As mentioned earlier, Scikit-learn's `GridSearchCV` module was used to find the optimal parameters for each model. The metric that was used to evaluate the performance of each iteration was the Spearman Correlation Coefficient, and each iteration was done with k-fold cross-validation where $k = 5$. First, in order to get an idea of what parameters performed well, larger intervals of parameters were used. Once a general idea of the range of parameters that performed well was known, the intervals were narrowed down to obtain better results with more precise parameters and to speed up the grid-search. For SVR, the RBF kernel was used, and the parameters that were varied were `C`, `epsilon`, and `gamma`. `C` is the penalty parameter for the error term, `epsilon` is the margin of tolerance within which the error is negligible, and `gamma` is the parameter that is passed into the RBF kernel, corresponding to σ in Equation 2. For RFR, the parameters that were varied were `max_features`, `max_depth`, and `min_weight_fraction_leaf`, using the default value of 10 decision trees. `max_features` represents the number of features to look for when looking for the best fit, `max_depth` represents the maximum depth that a tree can be, and `min_weight_fraction_leaf` is the minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node [24].

The Spearman correlation of the best performing results with optimal parameters from regression (SVR and RFR) on different features (LBP, Inception, ResNet) are shown in Table VI. Results from Schifanella et al.'s CrowdBeauty model are also included in Table VI as a baseline. All results, with corresponding best parameters can be seen in Appendix A.

TABLE VI: Best Performing Spearman Correlation of Different Methods on Separate Categories

Method	**Animals**	**Nature**	**People**	**Urban**
LBP + SVR	0.47	0.43	0.42	0.37
LBP + RFR	0.42	0.37	0.36	0.28
Inception + SVR	0.31	0.26	0.11	0.30
Inception + RFR	0.20	0.30	0.29	0.21
ResNet + SVR	0.77	0.66	0.71	0.70
ResNet + RFR	0.66	0.50	0.62	0.52
CrowdBeauty [10]	0.54	0.34	0.42	0.46

From looking at these results, the best performing methods used the ResNet deep features. All categories using ResNet were able to outperform the results of Schifanella et al., and LBP with SVR was able to outperform or match CrowdBeauty in two categories (nature and people). The parameters that gave the best results for each category can be seen in Table VII, along with the percentage increase in performance compared to CrowdBeauty. The significantly better performance using ResNet deep features is impressive, given that the training set used is smaller than the CrowdBeauty training set. Interestingly, the methods that used the Inception deep features had the worst performance in all categories.

Seeing that the models that used the ResNet deep features performed significantly better than Schifanella et al.'s CrowdBeauty results, a model was trained on all categories, rather than separately on the individual categories. This was done to see if the model could perform well without knowing what category the images fall under. Results can be seen in Table VIII. Note that results from Schifanella et al. are not included as they did not test a model on all categories together.

The overall results from the models trained on all categories using ResNet deep features were better than any individual category from Schifanella et al., although less than the best performing individual categories for the respective regression methods from this paper shown in Table VI.

TABLE VII: ResNet + SVR Results with Best Parameters and Percentage Increase in Performance

Category	**Spearman**	**Best Parameters**	**% Performance Increase**
Animals	0.77	**C**: 1 **gamma**: auto **epsilon**: 0.1	42.6%
Nature	0.66	**C**: 5 **gamma**: 0.0001 **epsilon**: 0.1	94.1%
People	0.71	**C**: 5 **gamma**: 0.0001 **epsilon**: 0.15	69.0%
Urban	0.70	**C**: 1 **gamma**: auto **epsilon**: 0.1	52.2%

TABLE VIII: Spearman Correlation and Best Parameters of ResNet with SVR and RFR on All Categories

	Spearman	**Best Parameters**
ResNet + SVR	0.73	**C**: 1 **gamma**: auto **epsilon**: 0.1
ResNet + RFR	0.57	**max_depth**: 40 **max_features**: auto **min_weight_fraction**: 0.01

VI. Discussion

As shown in Section V, all of the best performing models used the ResNet deep features, specifically with SVR as the regression method. The ResNet features with RFR also performed very well, outperforming CrowdBeauty in every category. From this, it can be inferred that the features extracted using ResNet were able to identify features that could predict beauty. However, it is difficult to interpret the ResNet deep features to understand what differentiates a more beautiful image from a less beautiful image. This lack of interpretability is a common caveat of deep learning methods. RFR could be further analyzed to determine which deep features have the strongest influence on the beauty score, but without deeper insight into the meaning of these features, it would still be difficult to interpret why they inform beauty.

Though it is difficult to know exactly why these features are able to predict beauty scores so well, the high performance of these models indicates that transfer learning with pre-trained CNNs can be quite effective. The ResNet CNN was not trained with the intention of predicting beauty, yet the features it generates are quite effective for this purpose. Note that the ResNet features had many more components after dimensionality reduction than Inception features, which may have some influence on ResNet's superior performance.

It is also significant that both the SVR and RFR models trained on ResNet features were able to perform well even when the images were not separated by category. Schifanella et al. did not report results for models trained on all categories at once. However, the Spearman correlations achieved by both of these category-agnostic models were higher than any of the category-specific CrowdBeauty models correlations. This might be because the features extracted from the last convolutional layer of a CNN typically represent higher level image features, which might capture information that inherently separates the various image categories.

Furthermore, the model trained on LBP features with SVR was also able to perform decently well, outperforming CrowdBeauty in the "nature" category and matching CrowdBeauty in the "people" category. LBP only works on greyscale images and thus does not capture any information about colour, but typically works well for texture classification. The better performance in the "nature" category might be able to be explained by the different types of textures that are widely prevalent in nature, which may have some influence on their beauty scores.

VII. Conclusions

In conclusion, the techniques used in this paper were successful in outperforming Schifanella et al.'s CrowdBeauty for predicting the beauty of images in four categories. Specifically, SVR on ResNet deep features performed best, increasing the Spearman correlation between predicted beauty and the "true" beauty of images by between 42.6% and 94.1%, depending on the image category. In addition, both SVR and RFR were able to outperform CrowdBeauty when the models were trained on deep features without prior knowledge of the image category. Next steps and recommendations include further investigation to be done on deep features, including attempting to use different pre-trained CNN models and extracting features from different layers of the network.

From this experiment, it can be said that it is possible to train a machine to identify beautiful pictures. While the machine might not truly understand the notion of beauty, it was certainly quite successful in learning the relationship between an image's characteristics and its human-ascribed beauty. Thus, the models were capable of developing an objective notion of what makes an image beautiful. It is important to remember though, that the notion of beauty learned is entirely dependent on the beauty scores were provided to the model. Thus perhaps beauty is still in the eye of the beholder, and the machine's subjective notion of beauty is biased to represent the collective intuition of the individuals who contributed to the beauty scores in this dataset.

References

[1] M. W. Hamilton, *Molly Bawn*. Biblio Bazaar, 2008.

[2] W. Shakespeare, *The complete works of William Shakespeare*. Race Point Publishing, 2014.

[3] D. Hume, *Moral and political philosophy*. Simon and Schuster, 2010.

[4] C. Sartwell, "Beauty," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Winter 2017, Metaphysics Research Lab, Stanford University, 2017.

[5] J. Barnes, *The Complete Works of Aristotle. The Revised Oxford Translation. 2 vols.(Bollingen Series, 71: 2)*. Princeton: Princeton University Press, 1984.

[6] S. Aloufi, S. Zhu, and A. El Saddik, "On the prediction of flickr image popularity by analyzing heterogeneous social sensory data," *Sensors*, vol. 17, no. 3, p. 631, 2017.

[7] D. Gray, K. Yu, W. Xu, and Y. Gong, "Predicting facial beauty without landmarks," in *European Conference on Computer Vision*, Springer, 2010, pp. 434–447.

[8] F. Chen, X. Xiao, and D. Zhang, "Data-driven facial beauty analysis: Prediction, retrieval and manipulation," *IEEE Transactions on Affective Computing*, 2016.

[9] Y. Wu, C. Bauckhage, and C. Thurau, "The good, the bad, and the ugly: Predicting aesthetic image labels," in *Pattern Recognition (ICPR), 2010 20th International Conference on*, IEEE, 2010, pp. 1586–1589.

[10] R. Schifanella, M. Redi, and L. M. Aiello, "An image is worth more than a thousand favorites: Surfacing the hidden beauty of flickr pictures.," 2015.

[11] T. Ojala, M. Pietikainen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[12] T. Maenpaa, *The local binary pattern approach to texture analysis: extensions and applications*. Oulun yliopisto Oulu, 2003.

[13] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[14] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 1329–1335.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[16] F. Chollet *et al.*, *Keras*, https://keras.io, 2015.

[17] B. Athiwaratkun and K. Kang, "Feature representation in convolutional neural networks," *arXiv preprint arXiv:1507.02313*, 2015.

[18] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.

[19] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[20] V. F. Rodriguez-Galiano and P. M. Atkinson, "Modelling interannual variation in the spring and autumn land surface phenology of the european forest," *Biogeosciences*, vol. 13, no. 11, p. 3305, 2016.

[21] *Spearman's rank-order correlation*. [Online]. Available: https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php (visited on 04/23/2018).

[22] J. L. Myers, A. D. Well, and R. F. Lorch Jr, *Research design and statistical analysis*. Routledge, 2013.

[23] *The correlation coefficient (pearson's r)*. [Online]. Available: http://pages.ucsd.edu/~aronatas/corr.html (visited on 04/23/2018).

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Appendix A
Complete Model Results

TABLE IX: LBP + SVR Results

Category	R^2	Spearman	Best Parameters
Animals	0.22	0.47	**C**: 64 **gamma**: 16 **epsilon**: 0.01
Nature	0.13	0.43	**C**: 64 **gamma**: 8 **epsilon**: 0.01
People	0.13	0.42	**C**: 64 **gamma**: 16 **epsilon**: 0.01
Urban	0.15	0.37	**C**: 64 **gamma**: 16 **epsilon**: 0.01

TABLE X: LBP + RFR Results

Category	R^2	Spearman	Best Parameters
Animals	0.19	0.42	**max_depth**: 60 **max_features**: auto **min_weight_fraction**: 0.01
Nature	0.12	0.37	**max_depth**: 90 **max_features**: log2 **min_weight_fraction**: 0.01
People	0.16	0.36	**max_depth**: 70 **max_features**: auto **min_weight_fraction**: 0.01
Urban	0.10	0.28	**max_depth**: 40 **max_features**: auto **min_weight_fraction**: 0.01

TABLE XI: Inception + SVR Results

Category	R^2	Spearman	Best Parameters
Animals	0.07	0.31	**C**: 1 **gamma**: 0.0001 **epsilon**: 0.1
Nature	0.02	0.26	**C**: 1 **gamma**: 0.0001 **epsilon**: 0.01
People	-0.07	0.11	**C**: 5 **gamma**: 0.0001 **epsilon**: 0.15
Urban	0.08	0.30	**C**: 1 **gamma**: 0.0001 **epsilon**: 0.15

TABLE XII: Inception + RFR Results

Category	R^2	Spearman	Best Parameters
Animals	0.12	0.30	**max_depth**: 40 **max_features**: auto **min_weight_fraction**: 0.01
Nature	0.06	0.29	**max_depth**: 10 **max_features**: auto **min_weight_fraction**: 0.05
People	-0.02	0.21	**max_depth**: 90 **max_features**: auto **min_weight_fraction**: 0.01
Urban	0.08	0.29	**max_depth**: 60 **max_features**: auto **min_weight_fraction**: 0.05

TABLE XIII: ResNet + SVR Results

Category	R^2	Spearman	Best Parameters
Animals	0.58	0.77	**C**: 1 **gamma**: auto **epsilon**: 0.1
Nature	0.40	0.66	**C**: 5 **gamma**: 0.0001 **epsilon**: 0.1
People	0.50	0.71	**C**: 5 **gamma**: 0.0001 **epsilon**: 0.15
Urban	0.51	0.70	**C**: 1 **gamma**: auto **epsilon**: 0.1

TABLE XIV: ResNet + RFR Results

Category	R^2	Spearman	Best Parameters
Animals	0.41	0.66	**max_depth**: 80 **max_features**: auto **min_weight_fraction**: 0.01
Nature	0.19	0.50	**max_depth**: 70 **max_features**: auto **min_weight_fraction**: 0.05
People	0.33	0.62	**max_depth**: 30 **max_features**: auto **min_weight_fraction**: 0.05
Urban	0.27	0.54	**max_depth**: 40 **max_features**: auto **min_weight_fraction**: 0.05

Appendix B
Code

A. Helper Functions: prepare_features.py

```
import numpy as np
import pandas as pd
from scipy.misc import imread


def load_descriptions(base_path=None):

    """Load the CrowdFlower features DataFrame."""

    if base_path is None: base_path = ''

    df = pd.read_csv(base_path + './download_summary.tsv', sep="\t", index_col=False)

    df = df[df.downloaded == True]

    df = df.rename(columns={'#flickr_photo_id': 'flickr_photo_id'})
    df['img_path'] = df.flickr_photo_id.apply(
        lambda x: base_path + './photos/' + str(x) + '.jpg'
    )

    return df


def read_colour_img(path):

    """Read colour image from disk."""

    return imread(path, mode='RGB')


def read_grayscale_img(path):

    """Read gray-scale image from disk."""

    img = imread(path, mode='F')
    img /= np.max(img)
    return img
```

B. Feature Extraction: Local Binary Patterns and Deep Features

```python
from keras import applications
from keras.applications.resnet50 import preprocess_input
import numpy as np
from skimage import feature as skimage_feature

import prepare_features

### Local Binary Patterns ###

def extract_lbp_features(base_path=None, output_path='./img.w.lbp.features.tab'):

    """Iterate through the dataset, read in each image, and create the LBP feature vector."""

    # Load the dataset from the 'base_path'.
    df = prepare_features.load_descriptions(base_path)
    df['lbp_feature'] = None

    for idx, row in df.iterrows():
        # Extract LBP features for image
        img = prepare_features.read_grayscale_img(row.img_path)
        df.at[idx, 'lbp_feature'] = lbp(img)

    # Write the features to a CSV
    if output_path is not None:
        df.to_csv(output_path, index=False, sep="\t")

def lbp(x, p=8, r=1):

    """Generate an LBP Feature vector for the input image 'x'."""

    features = np.histogram(skimage_feature.local_binary_pattern(x, P=p, R=r).ravel(),
                            bins=2 ** p, range=(0, 2 ** p), normed=True)[0].tolist()
    return features

### Deep Features ###

def get_deep_feature(model, img):

    """Generate a deep feature vector for the input image using the input model."""

    # prepare the image for the CNN
    img = img.astype(np.float64)
    img = np.expand_dims(img, axis=0)
    preprocessed_img = preprocess_input(img)

    # extract the deep features
    features = model.predict(preprocessed_img)
    return features[0]

def extract_deep_features(output_path, base_path=None):

    """Iterate through the dataset, read in each image, and create the deep feature vector."""

    # Load the pre-trained CNN.
    model = applications.resnet50.ResNet50(weights='imagenet', include_top=False, pooling='avg')
    # model = applications.inception_v3.InceptionV3(weights='imagenet', include_top=False, pooling='avg')

    df = prepare_features.load_descriptions(base_path)
    df['deep_feature'] = None

    for idx, row in df.iterrows():
        # Extract Deep Features
        img = prepare_features.read_colour_img(row.img_path)
        # Try-except is necessary because images that are inappropriately sized for the CNN will fail.
        try:
            feature = get_deep_feature(model, img)
            df.at[idx, 'deep_feature'] = feature
        except (KeyboardInterrupt, SystemExit):
            raise
        except:
            df.at[idx, 'deep_feature'] = None

    # Pickle the dataframe
    df.to_pickle(output_path + '.p')
```

C. Model Training and Model Evaluation: regression.py

```python
import numpy as np
import pandas as pd
from scipy.stats import spearmanr
from sklearn.decomposition import PCA
from sklearn.metrics import r2_score, make_scorer
from sklearn.model_selection import train_test_split, GridSearchCV


def gridsearch(classifier, param_grid, X_train, y_train, X_validation, y_validation):

    """Helper function to perform GridSearchCV on the input classifier and param grid."""

    print("Param Grid: " + str(param_grid))

    X_train = np.array(X_train)
    y_train = np.array(y_train).ravel()
    X_validation = np.array(X_validation)
    y_validation = np.array(y_validation).ravel()

    # Define the Spearman correlation scoring function
    score_func = make_scorer(lambda truth, predictions: spearmanr(truth, predictions)[0],
                             greater_is_better=True)

    print("Starting Gridsearch...")
    # Train the classifier on the training data using GridSearch
    classifier = GridSearchCV(classifier, param_grid, cv=5, scoring=score_func, verbose=0)
    classifier_fit = classifier.fit(X_train, y_train)
    print("Completed Gridsearch.")

    # Print the best parameters and results on the training data
    print('Best Params:' + str(classifier_fit.best_params_))
    print("Best Score: " + str(classifier_fit.best_score_))

    # Use the best fit to predict the beauty scores of the test set
    y_validation_pred = classifier_fit.predict(X_validation)
    print("Validation R^2: " + str(r2_score(y_true=y_validation, y_pred=y_validation_pred)))
    print("Spearman Rank Coefficient: " + str(spearmanr(y_validation_pred, y_validation)))

    return y_validation_pred


def regress_lbp(input_path, classifier, param_grid, output_path):

    """Perform GridSearch using the input classifier and param grid to predict beauty scores using LBP
    features."""

    # Load the LBP features
    df = pd.read_csv(input_path, sep="\t", index_col=False,
                     converters={"lbp_feature":
                                     lambda x: ([float(i) for i in x.strip("[]").split(", ")])})

    # Take the average of the beauty scores (this is the target variable)
    df['avg_beauty_score'] = df.beauty_scores.apply(
        lambda x: np.mean(np.asarray([int(i) for i in x.split(',')]))
    )
    df['predicted_score'] = None

    for group, data in df.groupby(['category']):
        print("\n\t\tTraining SVR models for category: " + str(group) + "\n")

        X = pd.DataFrame(data.lbp_feature.tolist())
        y = pd.DataFrame(data.avg_beauty_score)

        (X_train, X_test, y_train, y_test) = train_test_split(X, y, test_size=0.33, random_state=42)

        pred = gridsearch(classifier, param_grid, X_train, y_train, X_test, y_test)

        df.ix[y_test.index.values, 'predicted_score'] = pred

    # Write results to CSV
    df.to_csv(output_path)



```

```python
def regress_deep_features(input_path, classifier, param_grid, output_path):

    """Perform GridSearch using the input classifier and param grid to predict beauty scores using deep
     features."""

    # Load the deep features
    # These were pickled because the vectors were too big to save to CSV.
    df = pd.read_pickle(input_path)

    # Take the average of the beauty scores (this is the target variable)
    df['avg_beauty_score'] = df.beauty_scores.apply(
        lambda x: np.mean(np.asarray([int(i) for i in x.split(',')]))
    )
    df['predicted_score'] = None

    # Remove any invalid deep features
    df = df[df.deep_feature.notnull()]
    df['valid_deep_feature'] = df.deep_feature.apply(
        lambda x: not np.any(np.isnan(x)) and np.all(np.isfinite(x)))
    df = df[df.valid_deep_feature]

    for group, data in df.groupby(['category']):
        print("\n\t\tTraining SVR models for category: " + str(group) + "\n")

        # For deep features, training and test datasets were manually labelled
        # to make sure that they matched those generated in LBP. This was necessary
        # because deep feature extraction failed for 1 image in Inception and for
        # 35 images in ResNet.
        X_train = pd.DataFrame(data[data.group == 'Train'].deep_feature.tolist())
        X_test = pd.DataFrame(data[data.group == 'Test'].deep_feature.tolist())
        y_train = pd.DataFrame(data[data.group == 'Train'].avg_beauty_score)
        y_test = pd.DataFrame(data[data.group == 'Test'].avg_beauty_score)

        print("Reducing dimensionality with PCA.")
        pca = PCA(0.95, random_state=42)
        pca.fit(X_train)
        X_train = pca.transform(X_train)
        print("PCA Reduce Dimensionality to: " + str(X_train.shape[1]) + " components.")
        X_test = pca.transform(X_test)

        pred = gridsearch(classifier, param_grid, X_train, y_train, X_test, y_test)

        df.ix[y_test.index.values, 'predicted_score'] = pred

    # Write results to CSV
    df.to_csv(output_path)
```

D. Examples: Reproducing a subset of results in Appendix A

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR

import regression


def perform_svr_on_lbp():

    """This example generates the best performing SVR LBP results in Appendix A."""

    input_path = '../img.w.lbp.features.tab'
    output_path = '../test.new.code.csv'
    classifier = SVR()
    param_grid = {
        'C': [64],
        'gamma': [8, 16],
        'epsilon': [0.01],
        'kernel': ['rbf'],
    }
    regression.regress_lbp(input_path, classifier, param_grid, output_path)


def perform_svr_on_deep_features():

    """This example generates the best performing SVR ResNet results in Appendix A."""

    input_path = '../resnet.grouped.deep.features.p'
    output_path = '../test.new.code.csv'

    classifier = SVR()
    param_grid = {
        'C': [1, 5],
        'gamma': ['auto', 0.0001],
        'epsilon': [0.1, 0.15],
        'kernel': ['rbf'],
    }
    regression.regress_deep_features(input_path, classifier, param_grid, output_path)


def perform_rfr_on_lbp():

    """This example generates the best performing RFR LBP results in Appendix A."""

    input_path = '../img.w.lbp.features.tab'
    output_path = '../test.new.code.csv'

    classifier = RandomForestRegressor()
    param_grid = {
        'max_features': ['auto', 'log2'],
        'max_depth': [40, 60, 70, 90],
        'min_weight_fraction_leaf': [0.01]
    }
    regression.regress_lbp(input_path, classifier, param_grid, output_path)


def perform_rfr_on_deep_features():

    """This example generates the best performing RFR ResNet results in Appendix A."""

    input_path = '../resnet.grouped.deep.features.p'
    output_path = '../test.new.code.csv'

    classifier = RandomForestRegressor()
    param_grid = {
        'max_features': ['auto'],
        'max_depth': [30, 40, 70, 80],
        'min_weight_fraction_leaf': [0.01, 0.05]
    }
    regression.regress_deep_features(input_path, classifier, param_grid, output_path)
```