LCAV - EPFL

Communication Systems

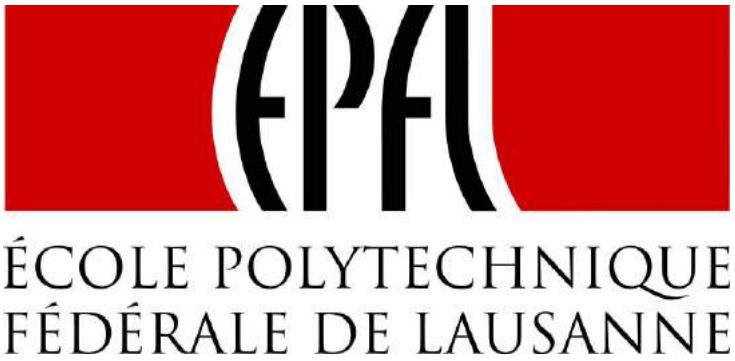
Master Semester Project

# Visually Pleasing Panoramas

*Author:*
Dunai Fuentes
*Email:*
dunai.fuenteshitos@epfl.ch

*Supervisors:*
Adam Scholefield
Benjamín Béjar

June 10, 2016



**Abstract**

In this semester project we develop an automated system to rank panoramas captured by the EPFL Livecam according to how visually appealing they are. In other words, a system able to predict an average of the rating of human users for such panoramas. As a way to achieve it, we explore the capabilities of deep neural networks to solve the problem given three different datasets: a general dataset for aesthetics (Photo.net), a custom dataset built with panoramas extracted from Flickr, and a domain specific dataset built from evaluated panoramas extracted from the camera itself. We show that shallow network architectures, limited computational resources, and scarce data (as opposed to highlighted literature on deep nets), are sufficient to provide satisfactory rankings. Finally a roadmap for further development is presented.

# Contents

# 1 Introduction

## 1.1 Motivation

Our panoramic camera takes roughly one image every 10 minutes from 6 a.m. to 8 p.m. In the course of three months it has taken over 5700 images of the landscape we see from EPFL. At the website `http://panorama.epfl.ch/` the user can retrieve images or galleries by date and hour. However, we would very much like to offer the user a curated selection of some of the best panoramas that we have captured so far. To create this shortlist, a human professional would have to evaluate thousands of images, being not only tedious but also causing fatigue that could potentially alter the professional's reliability. Creating a system that can automate the task, not only shortlisting but ranking the totality of our database, is of substantial value. Even if it is only used to exclude uninteresting pictures (foggy weather, completely dark images at early morning or late night, etc.), it would dramatically reduce the number of man hours needed to evaluate the remaining subset. A system of this characteristics would be easy to export to other services alike ours. Furthermore, if trained appropriately, it can agglomerate the criteria of more than just one worker, standardizing the subjective nature of this task.

In more general terms, allowing algorithms to grasp the concept of beauty or visually pleasantness, can help to develop more accurate recommender systems, better scene description programs, human-like artificial intelligence, etc. Commercialization of such technology is already happening. EyeEm Vision, for instance, provides in their software "The Roll" capabilities to rank and tag pictures from your camera roll, facilitating posterior browsing of your content.

## 1.2 Why deep learning

Although a first step goal of the project was to check the effectiveness of deep learning in the particular problem of ranking panoramas in terms of beauty, there are several reason beyond mere exploration to consider this particular methodology.

It is well known the contemporary success of deep learning [12] in complex tasks such speech recognition [7], translation [20] and basically any learning

task with enough supporting data to prevent them from overfitting (a drawback of their great learning capacity). But it is in computer vision tasks where a particular kind of neural network, convolutional neural networks, have gotten the most attention. Object recognition [11] and more advanced scene description through deep networks is arguably our best immediate solution to provide machines with understanding of the physical world (trying to emulate our own way of understanding it). Deep convolutional networks can show vast discriminative capacity as it has been studied in challenges such as ImageNet [18], or been used to gain intuition of a relevant zone in which to perform a move in a game of Go [19]. In combination with recurrent neural networks that equip the system with memory capacity they can be used in general game playing [6] and as a persistent search and focus guidance of vision for robotics. At the light of these achievements and previous work on the field of aesthetics [13], it is clear that convolutional neural networks can be indeed used to tackle a rather abstract concept that is visually pleasantness. There is particular interest in achieving understating of the concept with these techniques, and no others, because of the implication that more general convolutional neural networks can incorporate this knowledge within more extensive capabilities: for example, a scene description software able to deliver subjective evaluation of the scene or particular objects on it; all without integrating new modules with different algorithms but rather by just adding new labeled data to the training phase of the network.

Convolutional neural networks can bring implementation simplicity in a way that is different from the "one rule for everything" in the learning process. It is the case of feature selection. It is not trivial to handcraft features that would be descriptive of the beauty of an image because we do not really have a well defined set of characteristics that would make an image beautiful. Previous work has used information about color[16] or image composition, that are extensively studied in photography; entropy of the image is also intuitively a good criteria as low entropy images are probably boring (ie: a plain color) and very high entropy images could well be just noise. Nevertheless, a small set of handcrafted features has been shown to be insufficient (or perhaps our selection inadequate) to separate the beautiful from the not so beautiful. We took 1000 binary labeled images ("below average beauty" and "above average beauty") from the Photo.net dataset (to be described later) and computed the map to an eight dimensional feature space: entropy, mean saturation, mean brightness, and the five most frequent hues on the image.

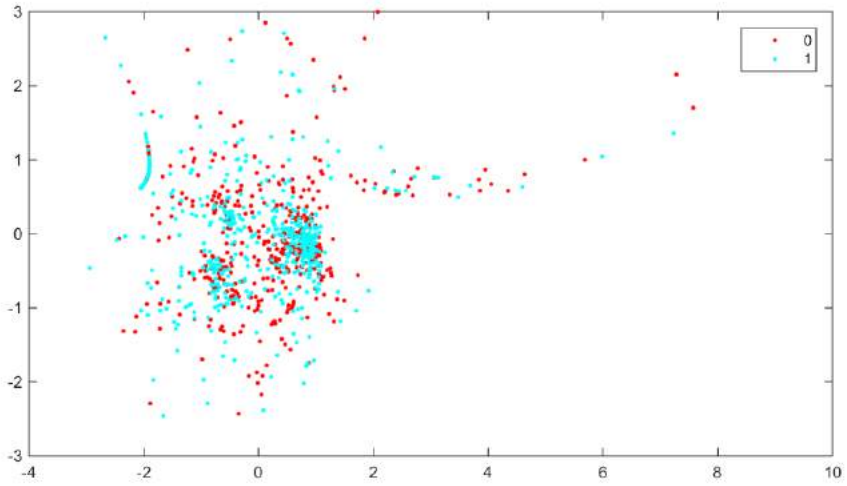



Figure 1: Multidimensional scaling visualization of binary labeled images from Photo.net displayed in a handcrafted feature space.

A linear classifier trained on these could only fit 50% of the training data (the same as random classification); visually we confirm that there is little discriminative power on the selected feature space. See Figure 1.

Our aim is for the convolutional layers contemplated as a feature extractor (trained in combination with the non-linear classifier than the dense hidden layers and final layer constitute) to deliver a better feature space that the one we just improvised, freeing us not only from crafting a set of relevant features, but also from preprocessing the images. The final schema should look as simple as Figure 2. As a drawback, our handcrafted features were size-independent, meaning that we could train with and evaluate images of different resolutions, while with our deep neural network we will have to chose the dimensions of the images in advance. *(Note that we may always crop, wrap or zero-pad).*

## 1.3 Resources

In this section we present the exact resources that had been employed in the course of the project, as reference and for those interested in replicating the

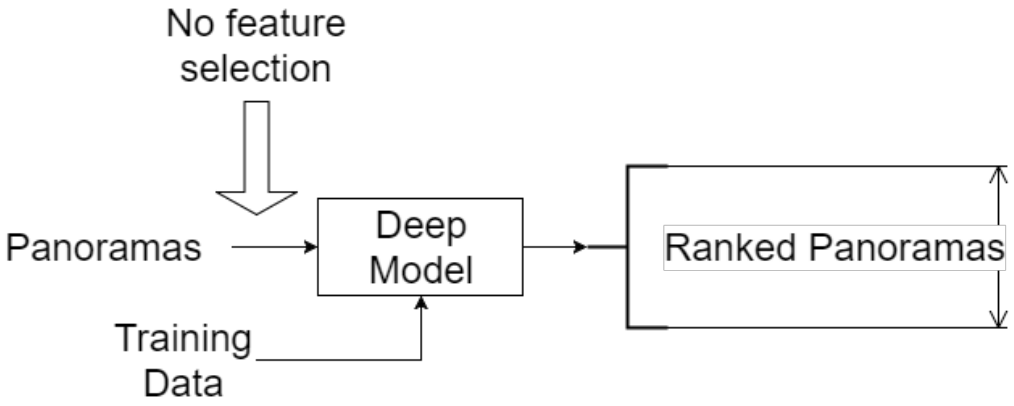



Figure 2: Descritive schema of the desired system and information flow. The complexity of the the deep network is the only point of computational hardness as there is no preprocessing of the images is necessary beyond matching they input size that the network requires, the same size as the training images.

experiments.

### 1.3.1 Hardware

One of the key findings of this project lies on the hardware, more particularly in the lack of extremely powerful computers at our disposal. This has forced us to build simpler networks (because memory capacity of the GPU could not afford a higher number of parameters) and to work with a moderate number of samples (because of the time it takes to train the networks).
Unless specified otherwise, featured neural networks had been trained using the following:

- SO: Windows 10 Home - 64 bits
- CPU: i7-4720HQ
- RAM: 16 GB
- GPU: GeForce GTX 960M

*Note that as we will be using CUDA most of the workload would lay on the GPU!*

### 1.3.2 CNTK

Computational Network Toolkit or CNTK is a unified deep-learning toolkit recently released by Microsoft Research. Still as a beta by the time of this project. All the information about CNTK is accessible through GitHub at `https://github.com/Microsoft/CNTK/wiki` and in their book[1].
Some of the reasons that have driven us to opt for CNTK instead of TensorFlow[1], Caffe[2], or any other valid packages for similar purpose and CUDA integration, had been the support for Windows OS as well as promised speed in performance. An important note is that CNTK supported only cuDNN 4.0 at the time of the installation (now 5.0!) while the GeForce GTX 960M required a newer version of the software. Custom installation of CNTK and modification of some of the files referring to cuDNN 4.0 (that are not indicated in the "custom installation" guide) had to be done.
Experiments should be easy to replicate with different software if all the parameters defined in .cntk files and the network architectures defined in .ndl files are translated to their correspondent form in the new environment. As a drawback, CNTK currently requires a new language developed exclusively for it, BrainScript.

### 1.3.3 Other software

For data mining and other scripts we have used MATLAB® R2014b as well as Python 2.7 based Anaconda® 4.0.0.

### 1.3.4 Files

In an effort of good practices and to boost reproducible research, all files related to the project have been made available at `https://infoscience.epfl.ch` as well as instructions to gather the images and construct the required datasets.

## 1.4 Quick review on Neural Networks

*Note: I would greatly recommend reading Nature's review [12] on this matter.*

[1] Google's solution: `https://www.tensorflow.org/`

[2] A deep learning framework Berkeley Vision and Learning Center: `http://caffe.berkeleyvision.org/`

The simplest neural network performs a linear combination of the inputs plus some bias, and then evaluates the result through a non-linear function. For the $j$th output we can write the following equation:

$$y_j = NLF(bias_j + \sum_i w_{ij} x_i).$$

The idea is to learn the weights and biases that approximate $y$ to a desired output (say, the labels associated to the inputs). Choosing the non-linear function is a design option although the most popular one is perhaps the logistic function, particularly for the output unit on binary classification (which can be understood as a probability):

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

In multi-class classification the logistic function evolves into the softmax function:

$$Softmax(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}},$$

where K is the total number of classes.

Stacking several layers of eligible sizes, one can obtain more powerful mappings and learn more complex relationships between input and output. An example of these architectures is presented in Figure 3[3].

To train the networks we define a cost function to minimize that depends on the outputs of the networks and the ground-truth we want to approximate. Most basics ones would be the Mean Squared Error or the Cross-Entropy error:

$$MSE = \sum_j (DesiredOutput_j - y_j)^2.$$

The most popular way to minimize these cost functions is to apply gradient descent using a trick called backpropagation to effectively tune the weights among several layers. Many changes to basic gradient decent had

[3]Thanks to Michael Nielsen's free book *Neural Networks and Deep Learning* at: `http://neuralnetworksanddeeplearning.com/`

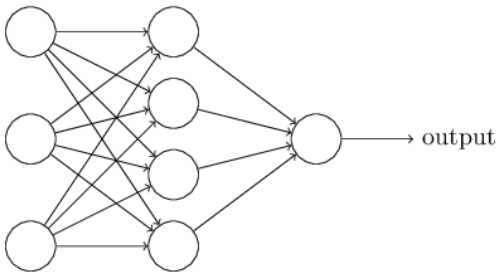



Figure 3: A neural network without biases, 3 inputs, 1 hidden layer with 4 units, and a single output. Each arrow is a weight, and on the circles of the hidden units and the output unit we perform the non-linearity.

been developed to improve the performance, such as mini-batch gradient descent or the use of momentum, both implemented in our experiments.

Finally, in the particular case of images, it is common to use convolutional layers that resemble in many cases Gabor filter banks for feature extraction. A descriptive example can be seen in Figure 4[3].

## 1.5 Convolutional Neural Networks for MNIST

As a warm up, and to test the correct functioning of CNTK, we started by solving classical digit recognition problem MNIST, presented by Yann LeCun and available at `http://yann.lecun.com/exdb/mnist/`.

Two convolutional layers followed by max-pooling, a dense or fully-connected layer and a softmax gave us a prediction error rate per sample of 0.0092, that is, less than 1 digit (0-9) every one 100 is wrongly classified. State of the art currently approaches 0.0023 [3]. Details on the parameters of our model can be found among the project files previously referenced. Note that due to random events in CNTK that do not depend on any seed, in this and all the experiments presented small variations can occur when replicating the results.

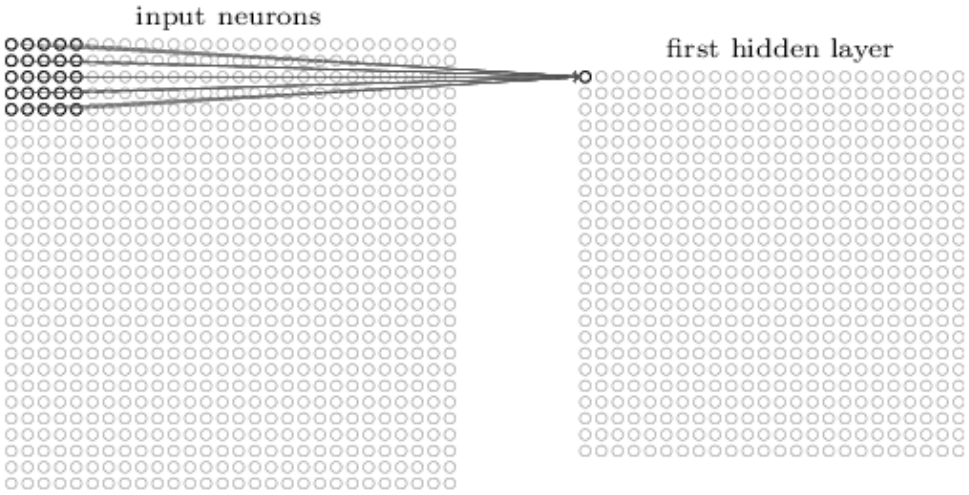



Figure 4: A 5x5 convolutional filter plus bias is applied to the input neurons (ie. the pixels of our image). The result can then be passed to yet another convolutional layer or vectorized and passed to a normal hidden layer. Normally in the middle of this transition a pooling layer is used to further reduce the dimensionality. The stride presented in the image is 1 both vertically and horizontally but this can also be modified with the purpose of reducing dimensionality.

# 2 Datasets

We have not been the firsts to tackle the problem of learning aesthetics. In [5] we can find a summary of some of the datasets that have already been created by the community and that could help us get straight to point. (They also explore the problem of finding good features for aesthetics that we discussed in the previous section). Unfortunately only an extended version of the Photo.net collection could be reused for our experiments. Also MIRFlickr [8], which in spite of not having the right tags for our main task, was useful in a tangential problem for which we explored scene recognition. CUHKPQ [14] proved impossible to obtain (constant interruptions while downloading the file from the server); AVA [15], in spite of being the most promising one because of its size, presented some copyright issues; DPChallenge [10] presented the same inconveniences as AVA.

Apart from these sets available to the public, we also built two more datasets specifically for our particular problem (both can be easily replicated from our files). The first with panoramas extracted from Flickr, and the second with panoramas extracted directly from our camera.

## 2.1 Dataset I - Photo.net

The version of Photo.net dataset employed on this project was the one made available by Ritendra Datta at his website `http://ritendra.weebly.com/aesthetics-datasets.html` which he used to study aesthetics and emotions in natural images [9]. After crawling through the IDs available at the dataset file, following the instructions specified at the dataset description file, 9206 images are downloaded out of the 20278 listed. The difference is mostly due to the fact that we weren't aiming for the whole set at once, but more training data won't hurt to those who can manage it. Note also that many of the photos are no longer available to download. The final subset is labeled with scores ranging from 1 to 7 that indicate higher aesthetic value for higher score. These scores are the average rating of at least 10 different users. We can see two examples of images depicted in this dataset in Figure 5. In spite of the two images having similar sizes and aspect ratio, this is not common to all the images of the dataset, that show great diversity on the matter. This fact will present a problem to us and it will be discussed in further detail in the "Implementation" section.

Figure 5: Images from Photo.net dataset. On the left, the image with ID: 1009826 has a rating of 5.36 out of 7. On the right, the image with ID: 1008958 has a rating of 5.17 out of 7.

## 2.2 Dataset II - MIRFlickr

There are two versions of the MIRFlickr dataset made available by LIACS Medialab at `http://press.liacs.nl/mirflickr/`, MIRFLICKR25000 with 25000 Flickr images and MIRFLICKR-1M with one million Flickr images. Both with metadata, annotations and descriptors. We proceeded with MIRFLICKR25000, but as it was mentioned for the Photo.net dataset, more training data for our networks is always useful for those with the resources to handle it. Both datasets can be downloaded at `http://press.liacs.nl/mirflickr/mirdownload.html`, all of the MIRFLICKR25000 dataset under Creative Commons license.

These images however, do not have any score associated to them as it was the case for Photo.net. Each image is associated to a collection of tags that range from description of some of the elements of the image, to specifications about the equipment employed to capture the scene. A list of common tags and number of instances is provided along with the files of the dataset, some of them are: explore, sky, nickon, 2007, blue. We can see two examples of images depicted in this dataset in Figure 6.

## 2.3 Dataset III - Custom panoramic dataset

While in MIRFlickr we can find some images tagged as panoramas, we lack the scores to actually use them to predict more scores. While in Photo.net we

Figure 6: Images from MIRFLICKR25000 dataset. On the left, image 1 of the dataset labeled with the tags: governorsisland, cigarette, tattoos, smoke, red, lipstick, dress, sunglasses, shades, belt. On the right, image 2 of the dataset labeled with the tags: explore.

have the necessary labels, the theme of the pictures is variable and does not apply directly to our problem of ranking panoramas. To get a new dataset containing panoramas and only panoramas, with a measurement of the image popularity, we decided to gather some data of our own.

Twitter and Instagram feature APIs for developers that can facilitate the collection of such kind of data, however, both presented copyright difficulties that encouraged us to move to a third option. Flickr was once again the repository that would provide us what we needed. With a simple script in Python using the package *flickrapi*[4], we were able to download a set of 200 images in high resolution (1600 pixels on the largest side) and a set of 400 images in medium resolution (1024 pixels on the largest side). Note that in spite of the restriction to the largest side, images still vary widely on ratio, which as stated for Photo.net supposes a problem for our system. We found a good labeling criteria (one that divides Flickr images into two similarly large groups, and only a few in the middle) to label as 0 or "not visually pleasing" those without likes and with 1 or "visually pleasing" those with 10 or more likes. Only images following this criteria were downloaded; the

[4] `https://stuvel.eu/flickrapi`

Figure 7: Images from the custom panoramic dataset. On the left, the first image with 0 likes of the medium resolution set. On the right, the first image with 10 or more likes of the medium resolution set.

script to do so has been made available and is easily modifiable for different settings. Our particular set is composed of images uploaded no sooner than the 20th of January of 2015 for no good reason; this is also easy to modify. A couple of examples can been seen in Figure 7.

## 2.4 Dataset IV - Domain specific dataset

For a final experiment we used a domain specific dataset built out of images extracted directly from the panoramic camera. That is, images very similar to the ones we will rank later on. A random subset of 1197 (1000 for training and 197 for validation) images was extracted from a total of 5703 downloaded images that constituted the months of January, February and March of 2016. This subset was labeled with scores from 0 to 9 **exclusively by myself**. Please note the huge bias of this dataset towards my own liking and taste. Note also that due to the specified selected months, some climatic conditions might be more prone to appear than some others. Do also consider the fact that I probably became more strict with my ratings as the time passed on, a factor for which no correction has been implemented. (I rated a permutation of the subset, so the position of the image is not related to the date or time in which it was taken). Scores are not uniformly spread (as it is normally the case with scores): for instance, I reserved the nines for pictures of outstanding beauty while I was much more open-handed with threes and fours.

The images of this collection have a common size of 4821 x 400 pixels

Figure 8: Images from the domain specific dataset. On top, the image captured the 21st of February at 10:10, with a score of 4. At the bottom, the image captured the 16th of January at 16:40, with a score of 8.

that corresponds with the low resolution version of the images displayed by the panoramic camera. Fixed size and suitability of the data are the two great advantages of this crafted dataset over the others. However, that very same suitability makes it less propitious for generalization. Two examples of images belonging to this dataset can be viewed in Figure 8.

## 2.5 Dataset 0 - Images to Rank

Datasets I to IV will play the role of training and validation datasets, but as our final mission is to deliver a ranking of different panoramas, we also made a manual selection of 19 of these pictures to rank, that would remain invariable as a comparison point of different approaches to the problem. Variety is a first priority on this selection to be able to determine certain preferences for certain kind of pictures as opposed to some others. Two sunsets, one very foggy image, a completely black image taken in the early morning, a sunrise... All of them images belonging to the year 2015 to avoid duplication of Dataset IV. A few examples are presented in Figure 9. Note the similarities with Dataset IV in contrast with the radically different imagery of datasets I to III.

Provided with the images, a set of three different rankings by human users. To evaluate the performance of our algorithms we will compare the matching against these three rankings. The favored metric that will be applied in this project will be Top-3 + Bottom-3 correct classification, but other metrics could well be of interest. The reason to take only the top and the bottom of the list is that it serves the two main purposes of the project: shortlisting by deleting the least beautiful ones (may also help us to save memory

Figure 9: A subset of 4 images belonging to the group of 19 images that we will attempt to rank automatically.

for instance), and identifying potential best pictures to enhance user experience. The benchmark to have as reference is the performance of a random ranking of these 19 images with the same criteria (ie. we randomly order the images and count the number of lucky guesses for each different user; the maximum score for one user is 6 while the maximum score for the average among the users is constrained by the similarity of the user's rankings). It has been approximated by simulating 1000 of these random permutations. The result can be observed in Figure 10.

# 3 Implementation

## 3.1 Approach 1 - General dataset for aesthetics

We decided to start tackling the problem of aesthetics in the most general way. Dataset I described above was our choosing. The idea is that a classifier able to correctly label the images in Photo.net dataset should also be able to *correctly label* (note that there is no ground truth) the images in our ranking dataset. Provided the generated labels, only sorting of the images remain for the rank to be constructed. This presents a couple of problems.

Instead of formulating it like a classification problem it would actually be better to formulate it as a regression one. This would not only make a more accurate ranking by leaving the constraints of integers for real numbers (take the case when multiple images have the same label and we still want to know

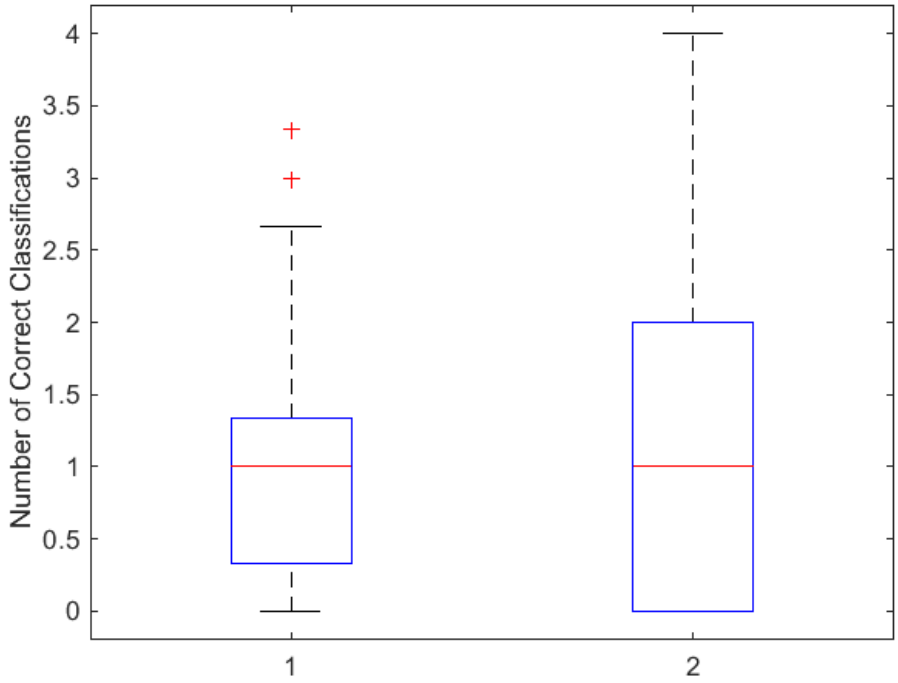



Figure 10: In this image we show how accurate a random ordering is when matching against one user (right side) or the average combination of the three (left side). Each boxplot is computed for 1000 elements. Note that because the users have ranked the images differently, a correct guess for user 1 may imply an incorrect guess for user 2, making it more difficult to excel in average. On the left, boxplot of the average (among the 3 human rankings) Top 3 + Bottom 3 correct classifications. On the right, boxplot of the Top 3 + Bottom 3 correct classifications for one specific user (because of the random nature of the ranking it is not relevant which specific user).

which one is best), but it is also straightforward from the fact that mean scores provided in Photo.net are already convenient to regression.

Yet we identify another way to solve the problem as a classification one. By computing binary labels for the whole set, "beautiful" and "not beautiful", we can train a classifier and finally use the probabilities of the image being "beautiful" as the ranking criteria in our test set. In this case we will proceed with this last solution for it's simplicity and not until Approach 3 will we shift to the regression task. Please note that there is a loss of information when converting the scores of the dataset to binary labels.

Now, we have set the threshold at the mean value of all the mean scores for the first 1200 images of the Dataset I (1000 for training and 200 for validation). Images with mean score above the threshold are labeled as "1" or "beautiful" and images with mean score below the threshold with "0" or "not beautiful". The final split between train and validation data leaves almost balanced numbers of each label, preventing the model from simply learning the statistics of the data.
A different option here would have been to define two thresholds, holding a gap between the "beautiful" and the "not beautiful" and simply not considering the controversial images that lie in between during the train phase. This would undoubtedly boost the classification performance because it dispenses of the most ambiguous images, yet again, these more subtle techniques will be adopted in a further stage into the project.

Only the structure of the classifier remains. Firstly, the image dimension has to be fixed, and it has to be the same for the whole training and validation sets. In an object classification task this does not suppose a big problem because we can explore the image taking multiple separated patches, and if in any of them the desired object is identified, then one can say that that object appears in the image. We, however, cannot extrapolate the concept of beauty to the whole image from its appearance in one of many patches. For the training phase we will have to agree on a common dimension for all images. Pre-processing and conversion by interpolating values in the expansions and removing them in the reductions is a possibility. Nevertheless, it is easy to agree that deformed images do not look so good; in some extreme cases we would have to completely invert the ratio of an image, dramatically changing its appearance and composition. It is not a good solution. What we

opted to do was to crop an established 200 times 200 patch from the center of the images. This is implemented in CNTK readers so no preprocessing is required. A patch size of 200x200 gives us a good compromise between the overall number of inputs (and parameters to train) and the natural size of the image itself. The fact that it is centered helps us disregard the borders that will probably contain less relevant information than the focus.

As we said, 200x200 is good enough for low resolution images that make the majority of Dataset I, however, even the low resolution version of panoramic images taken from the camera are way larger than this. Ranking a panorama based only on a small fraction from its center is not a good way to go.
To solve this issue, testing images have been fragmented into 48 disjoint 200x200 patches, each patch evaluated independently and then their results aggregated to determine the final score of the image. We move away from ranking according to the probability of "1", but rather according to an aggregation of different probabilities. We considered partitioning the high resolution images to bring more detail, but as the number of patches skyrocketed, we finally kept with the more conservative approach.

The definition of the aggregation function is controversial and research can be done to come up with more objective and better performing ones. For our experiment, we take the eight maximum probabilities (ie. the scores for the 8 best patches) and add them up. By doing so we do not take into account the whole image, but we emulate the human behavior of focusing our attention on the most appealing parts, ignoring uninteresting zones. "8" is an heuristic value for the parameter, and other values may also be found useful.

Other heuristics are also those belonging to the description of the neural network. A convolutional layer of size 5x5 receives the 3 channels of the color image and outputs a mapping of 36 channels. The stride of the sweep is set to one both vertically and horizontally. The initial values are taken randomly from uniform distributions from 0 to 1 in the case of the biases and 0 to 10 for the weights. The result of the convolutions passes through a rectified linear unit that returns the activations.
Because zero-padding is disabled, the output of the convolutional layer has a slightly smaller dimension than the input, which gets reduced even further by a pooling layer that takes the maximum every 4 values resulting in a 49x49x36 multidimensional array.

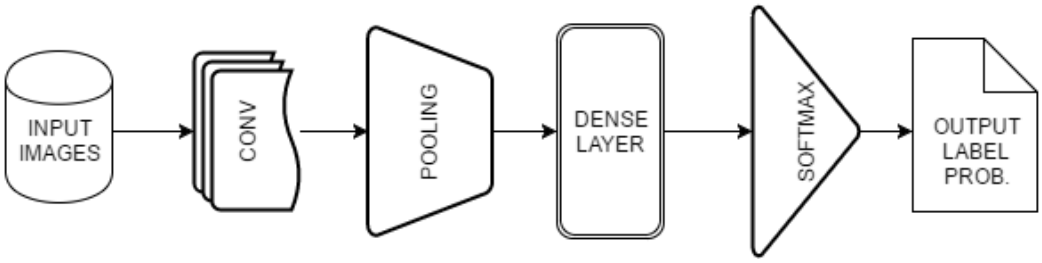



Figure 11: Conceptual schema for the Approach 1 neural network.

These 86436 nodes are fully connected to a 1000 units hidden layer featuring the logistic function, which at its time passes the activations to the final layer. Only one simple logistic unit is necessary in binary classification, but for simplicity with the CNTK configuration we chose a softmax from which only one output will be considered (the other is just be 1 minus the first one). The training criterion will be to minimize the cross-entropy error at the softmax output.

Note the simplicity of the network. A conceptual schema is presented in Figure 11. Normally several convolutional layers constitute these type of networks (remember that we had two of them for our MNIST experiment) but the limitations of our hardware made it impossible to grow more complex ones (these relatively large images are much more difficult to handle than MNIST's digits). The trade-off is mostly with the other memory consuming parameter, the minibatch size, set to 10 for this experiment. Larger minibatch size will produce a more accurate gradient when training the model while taking only a set of 10 images will give only a guess of where the real gradient is pointing, but it is much cheaper to compute.
To speed up the training process only 15 epochs are computed. Because we will not be using any regularization or dropout to prevent the network from overfitting (some tests were made, but no interesting results were obtained), early-stopping taking the form of a small number of epochs can help with this issue.
Overall, both the guessing nature of the small minibatch and of the small number of epochs, coincide with the the fact that beauty is not something we can fully determine, but rather intuit, and thus should be tackled with methods like the ones presented that are flexible on this matter.

Finally the results: 42% of the training data is fitted, but no transference to validation. Hasn't it learned anything? Well, it has. Perhaps not enough to predict whether an image from Dataset 1 is beautiful or not, but enough to determine which are the most and least beautiful panoramas. Have a look at the generated ranking in Figure 12. We can see that it has selected the two sunsets as the best images, but not only that, it also identifies the completely black image as the worst of the nineteen and the most plain foggy one as the 18th on the list. When we match the predictions to our three human made rankings, the "Top 3 + Bottom 3" scores are 1 for user 1, 4 for user 2 and 3 for user 3. Comparing to random rankings we can see that despite we would have failed to please user 1, we obtain significant improvements for users 2 and 3 (random ranking mean performance is 0.9780). On medium terms the score is 2.6667 against 0.9417 for the mean of the average random case.

## 3.2 Approach 1.5 - Tagging

We can already foresee a problem with the results of Figure 12. If a user wants to retrieve the top 10 pictures of the last year and the system responds back with 10 very similar sunsets, the user will surely be disappointed. To provide the user with a certain degree of diversity in our selection we could compute a pairwise distance metric between the panoramas and try to find a balance between high score and sufficiently distant images. However this is not only inconvenient due to the very high number of images our databases may store, but also fails to add general information that can be used for other tasks. Tagging, on the other hand, allows us to be able to provide such diversity, increments the number of services to the user (for example, now the user could query "top 10 sunrises") and doesn't need to store all the pairwise relationships.

For tagging we have moved to Dataset II, MIRFlickr, from where we have selected a balanced list pictures with common interesting tags: "sky", "sunset", "sunrise", "clouds" and "night" are the ones we have experimented with.

Again we proceed with 200x200 patches and the idea is to tag one of our panoramas as a sunset if it has more than $\alpha$ patches classified with high probability as sunset, as sunrise if it has more than $\beta$ patches classified with high probability as sunrise, and so on and so forth in exclusive order. We took a slightly different approach from our initial idea classifying each patch

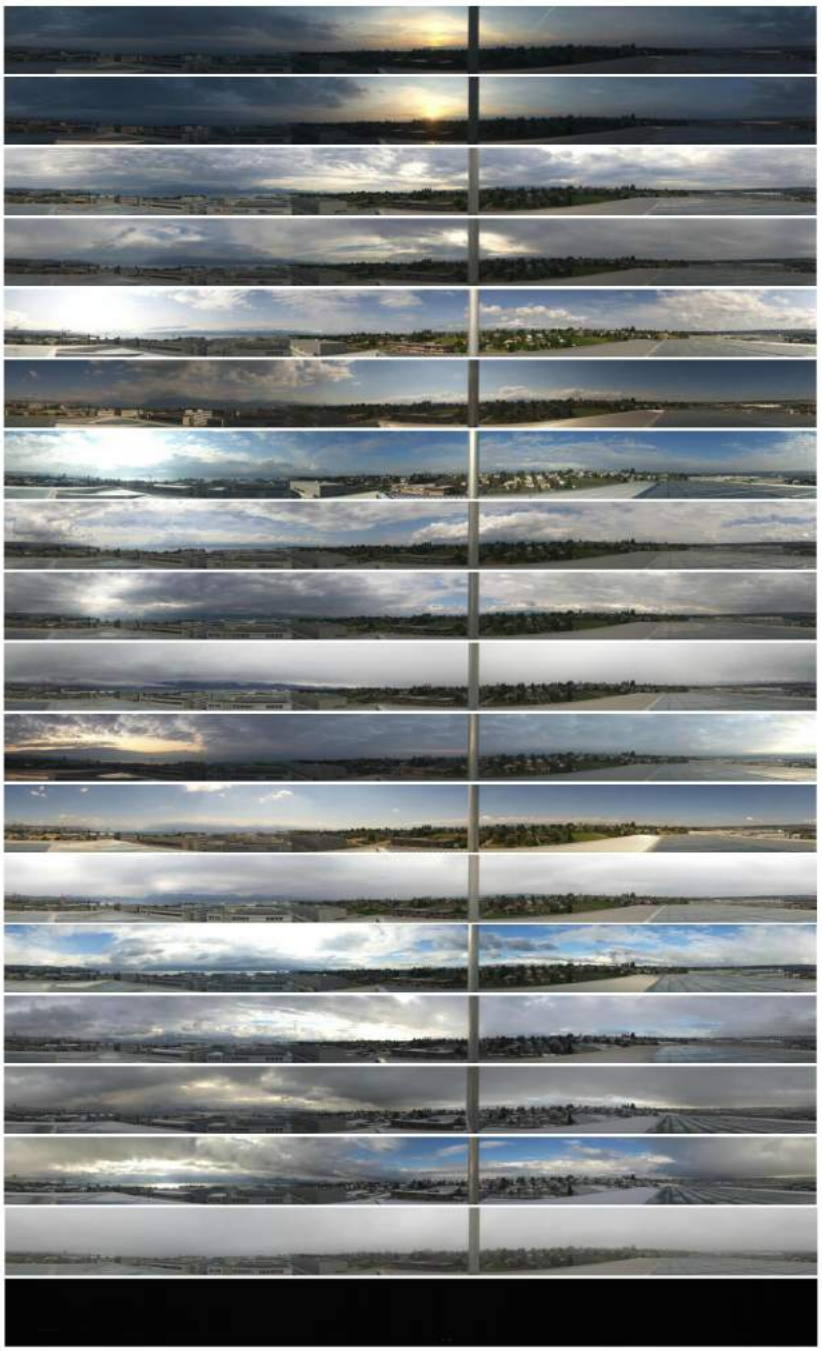


Figure 12: Ranking delivered by Approach 1.

Figure 13: Three samples of images classified as "sunset": two of them are correctly identified, another wrong one has passed through the thresholds. If we make the criteria more strict, eventually no images are tagged as sunset.

and then exclusively tagging the image depending on the number of patches with a certain tag. But how much is high probability? And which values should parameters $\alpha, \beta, etc.$ take? We have found that the results are extremely dependent on these questions and that there is no easy way to set this parameter without recurring to yet more machine learning techniques. Unfortunately we do not have a tagged sample database of panoramas taken from our camera from which we could learn the parameter and extend to the rest, and if we did, then we could simply skip the problematic by directly training the classification in the whole images.

The difficulty of the problem is sustained by the fact that exclusive tagging as we are aiming is not truly descriptive of the images. "sky" and "clouds" will surely be on sight during "sunset" and "sunrise", "night" during "sunset", "sky" during "clouds", etc. If on the other hand we decide to multitag, then most pictures will carry the "sky" tag that we would have liked to reserve for clear skies, losing it's meaning.

Some results can be found in Figure 13, and details of the neural network and parameter used for different trials can be found along with the project files in the linked repository. The classifier presents 41.93% error rate on training data, and 48.66% error rate on validation for 4-class classification.

Plenty of improvement can be done in the tagging, both in the strategies and the classifier, but as it wasn't the main focus of the project, we must continue.

## 3.3 Approach 2 - Custom panoramic dataset

Results with Dataset I were satisfactory but we wondered whether narrowing down the concept of beauty to only what makes a panorama beautiful could improve the results.

We switch to Dataset III and everything else is kept very similar to Approach 1 with the new stream of training data. However, results are rather disappointing. The "Top 3 + Bottom 3" criteria only gets a score of 1 for each test user, the completely black one. This is practically as bad as random.

There are a few reasons why this could be happening. The first is that to learn from the panoramic images a centered patch of 200x200 is not enough. The second is that perhaps it's due to the difference in the amount of data (approximately $\frac{1}{3}$ of the total used in Approach 1). We have compensated by reducing the number of hidden units from 1000 to 400 so the network cannot overfit, however, this also affects its capabilities. The third and most probable is that because these images had been extracted from Flickr, and their labels do not come from scores like in Photo.net, but from "likes", it is not a reliable dataset. Take for instance the case of a photograph that was uploaded late at night by a user without followers. Most probably it won't get any likes independently on how good the image is. On the other hand, a very popular user who also links his photos through other social networks to gather visits will surely get more than 10 likes again independently on how good the image is. Finally, the fourth reason is that the overall quality of the images in Dataset III is high, therefore reducing the "beautiful" distance among images. The difference between the good and the better is always more elusive than between the good and the bad.

## 3.4 Approach 3 - Domain specific dataset

For the final approach we are going to try to solve our particular problem in the most precise way. With Dataset IV, we have highly similar train data to that of our test set, and we can get rid of the patches and work directly with the entire images. We are moving away from the simplistic binary approach and building a multiclass classifier that integrates regression components.

Working with the whole images makes it much easier to relate, and the

benefits of a domain specific dataset are substantial, but there is a drawback: now our input dimension has scaled from the 200x200x3 nodes that we had before to 4800x400x3 that constitutes the number of pixels in the image times 3 color channels. This makes it impossible to train the model (as we had it before) in our GPU. To be able to train some modifications to the CNTK learned had to be done. We reduced the minibatch size to 6 (10 being already the minimum recommended in the literature) and to compensate for the misdirected learning we incremented the number of epochs to 30.

The architecture also received a couple of changes. To work with rectangular images it seems convenient to implement a rectangular window in the convolutional layer (10x4) and to speed up the shrink in the number of values to handle, horizontal stride is increased to 5, vertical to 3, and the pooling layer now keeps only 1 every 7 digits. The number of mappings is also reduced to from 36 to 24. The number of hidden units remains the same at 1000.
But the biggest difference comes at the final step of the network. We keep the softmax operating this time in multiclass, but we change now the minimization function:

$$f(\theta, \gamma) = CEE(\theta, \gamma) + w_1 * MSE(\theta, \gamma) - w_2 * VAR(\theta),$$

were $\theta$ is the matrix at the output of the softmax, $\gamma$ the target labels on the training data and $w_x$ simply weights (note that two of them are enough normalizing). CEE stands for Cross Entropy Error, MSE for Mean Square Error and VAR for variance.

Cross-Entropy error on its own does not take into account that the misclassification of an image as "8" when the target score is "3" should be penalized much stronger than if it is misclassified as "4". The mean square error helps with this problem. However, the MSE creates a dangerous tendency towards predicting mean values (ie. in our 0 to 9 scale, 4s would become a safer bet than 7s because, assuming uniformity, they are simply statically closer). To counter that effect we reward the classifier for producing high variance outputs instead of always assigning a fair deal of probability to the middle values (maximum reward to 0s and 9s).

But the 0-9 scale in which the images were rated has actually not been used. Neural networks for classification are prone to learn the statistics of

the data and make predictions only as the most popular labels (this quickly decreases their error rate). Because the ratings I gave to the images did not follow a uniform distribution something had to be done. To solve this same problem in Dataset I we moved to binary classification, setting the threshold at the mean (so we would have approximately the same number of 0s and 1s). This time we made groups of evenly distributed number of members with aggregations of the ratings: $[0, 1, 2]; [3]; [4]; [6, 7, 8, 9]$ form our new four classes. Note that rating 5 has been eliminated. This helps widening the gap between the mediocre 4 and the beautiful 6+, contributing to a more discriminative training and better results.

The classifier presents during training 44.22% error rate on training data, and 50.86% error rate on validation for 4-class classification.
Matching against human rankings, despite computing a slightly different proposition that the one presented in Approach 1, we obtain the same final results on testing: $1, 4, 3$ "Top 3 + Bottom 3" correct classifications for users $1, 2, 3$ respectively. As expected, due to the bias towards my own liking it performs particularly well at emulating my own ranking: a score of 4.

### 3.5 Summary of the results

A comparative recap of the obtained results is presented in Table 1.

Table 1: Top-3 + Bottom-3 correct classification.

| | User 1 | User 2 | User 3 | Average | Author's Rank |
|---|---|---|---|---|---|
| Random (mean) | 0.9780 | 0.9780 | 0.9780 | 0.9417 | 0.9780 |
| Approach 1 | **1** | **4** | **3** | **2.6667** | N/A |
| Approach 2 | **1** | 1 | 1 | 1 | N/A |
| Approach 3 | **1** | **4** | **3** | **2.6667** | **4** |

## 4 Conclusions and roadmap for the future

We have seen that something as abstract and human as beauty can also be treated with algorithms and in particular deep neural networks, gaining valuable insight in our data that can be used to solve a variety of problems. But

there is still a long way to go. Gathering more labeled data to train more complex neural networks is perhaps the most immediate and straightforward option, but while it will surely help to perform better on the classification task (delivering objectively more accurate results in the end), it is a costly solution. Ways to reduce our dependency on data would be to bring unsupervised learning techniques as pre-traning [4]. This could particularly help building the filter banks of the first convolutional layer. Or we may also directly import them from other networks [17], like those trained for object recognition task (as long as they respect the dimensionality constraints of our input space), and fine tuning them for our particular purpose. Nevertheless, I would like to emphasize how this two tasks and others like theme recognition are good candidates to be solve by a single more general network.

In our experiments we have stayed relatively shallow, showing that aesthetics is perhaps not so elusive and complex as we first thought. However, for those following our steps and interested in going deeper, layer-by-layer discriminative pre-training[2] is a possibility within the CNTK framework and a powerful tool when data is scarce, as it is in our case.

Collecting training and testing data, which was one of the main objectives of this project, has presented unexpected difficulties. Labeling images as it was done for Dataset IV suffers from tediousness that ultimately affects the quality of the data itself. Even worse is the case of ranking a set of images. Just with the 19 pictures in Dataset 0 it was detected to be a demanding task to provide their corresponding rank (no labels, respective positions of one to another). Methods to facilitate this necessary stage of the learning pipeline may also be improved. A suggestion for the ranking case would be to breakdown big sets into smaller ones (say, 4 pictures each), grouping combinations of them so we can infer the global ranking from the smaller ones.

So far we have explored averaging methods that try to come up with middle point solutions. We should question however if this is the paradigm we want to pursue. Take the following case: user1 ranks a set as Image1, Image2, Image3 in preference order; user2 ranks the same set as Image3, Image2, Image1. Now, an algorithm that gets highly punished for displeasing the users will probably propose Image2 as the best one, hence failing to truly satisfy any of the users. I would argue that real objective is not to build an algorithm that imitates an averaged group of opinions, but rather one

that imitates an average human opinion. And if we want to get specific, a particular user's opinion. This problem can be explored in two phases. First with clustering techniques detect if there are identifiable groups of users with a particular taste. Later, using feedback techniques, we can learn the preferences and taste of a particular user, identifying the cluster to which the user belongs. Finally, the two combined help the system building upon the averaged group of opinions for the cluster the ranking that the user really wants, his own. We have seen empirical repercussion of this problem all along the project were our algorithm worked nicely for users 2 and 3 but not so well for user 1.

Tagging can also help us to learn a users preference. An admiration for cloudy skies, dark and gloomy for the melancholic, etc. In general tagging provides us with extremely useful information that can solve the problem of diversity in ranking (as is proposed in this project) but also many more.

To finish, a word of advice. Aesthetics is a field centered on the user and user experience, and thus it is important to take into account not only the objective validity of our results, but how they affect the perception of the human who receives them. The diversity on our crafted gallery is an example, because too much of practically the same (even something good, even the best) can cause boredom from repetition. Moreover, it might even be smart to include some "not so good" images in our curated selection to (by contrast) extol the better ones. This questions the necessity of investing in highly accurate systems unless we bring the psychological aspects into the training. Perhaps the key element to improve upon.

# 5 Acknowledgments

Special thanks to both of my supervisors, Adam Scholefield and Benjamín Béjar, for valuable advice; to Geoffrey Hinton and his crew for making freely available the outstanding MOOC "Neural Networks for Machine Learning"; and to the CNTK team that developed the well-oiled machinery that made all of this possible.

# References

[1] Amit Agarwal, Eldar Akchurin, Chris Basoglu, Guoguo Chen, Scott Cyphers, Jasha Droppo, Adam Eversole, Brian Guenter, Mark Hillebrand, Ryan Hoens, Xuedong Huang, Zhiheng Huang, Vladimir Ivanov, Alexey Kamenev, Philipp Kranen, Oleksii Kuchaiev, Wolfgang Manousek, Avner May, Bhaskar Mitra, Olivier Nano, Gaizka Navarro, Alexey Orlov, Marko Padmilac, Hari Parthasarathi, Baolin Peng, Alexey Reznichenko, Frank Seide, Michael L. Seltzer, Malcolm Slaney, Andreas Stolcke, Yongqiang Wang, Huaming Wang, Kaisheng Yao, Dong Yu, Yu Zhang, and Geoffrey Zweig. An introduction to computational networks and the computational network toolkit. Technical Report MSR-TR-2014-112, August 2014.

[2] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. 2007.

[3] Dan C. Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *CoRR*, 2012.

[4] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pretraining help deep learning? *The Journal of Machine Learning Research*, 11:625–660, March 2010.

[5] J. Faria, S. Bagley, S. Rüger, and T. Breckon. Challenges of finding aesthetically pleasing images. In *2013 14th International Workshop on*

*Image Analysis for Multimedia Interactive Services (WIAMIS)*, pages 1–4, July 2013.

[6] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3338–3346. 2014.

[7] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, November 2012.

[8] Mark J. Huiskes and Michael S. Lew. The MIR Flickr Retrieval Evaluation. In *MIR '08: Proceedings of the 2008 ACM International Conference on Multimedia Information Retrieval*, 2008.

[9] D. Joshi, R. Datta, E. Fedorovskaya, Q. T. Luong, J. Z. Wang, J. Li, and J. Luo. Aesthetics and emotions in images. *IEEE Signal Processing Magazine*, 28(5):94–115, September 2011.

[10] Yan Ke, Xiaoou Tang, and Feng Jing. The design of high-level features for photo quality assessment. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 419–426, June 2006.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. 2012.

[12] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, May 2015.

[13] Xin Lu, Zhe Lin, Hailin Jin, Jianchao Yang, and James Z. Wang. Rapid: Rating pictorial aesthetics using deep learning. In *Proceedings of the 22Nd ACM International Conference on Multimedia*, MM '14, pages 457–466, 2014.

[14] Wei Luo, Xiaogang Wang, and X. Tang. Content-based photo quality assessment. In *2011 International Conference on Computer Vision*, pages 2206–2213, November 2011.

[15] N. Murray, L. Marchesotti, and F. Perronnin. Ava: A large-scale database for aesthetic visual analysis. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2408–2415, June 2012.

[16] M. Nishiyama, T. Okabe, I. Sato, and Y. Sato. Aesthetic quality classification of photographs based on color harmony. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 33–40, June 2011.

[17] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1717–1724, 2014.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[19] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[20] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. 2014.