Automated Image Timestamp Inference Using Convolutional Neural Networks

Prafull Sharma
prafull7@stanford.edu

Michel Schoemaker
michel92@stanford.edu

David Pan
napdivad@stanford.edu

Stanford University

Abstract

With the rise in amateur and professional photography, metadata associated to images could be valuable for both users and companies. In particular, the prediction of the time a photograph has been taken is not currently an active area of public research, and vast amounts of accurately labeled data is not available. In this paper, we propose methods to classify the time taken of a picture in two ways: using user submitted tags, namely "morning", "afternoon", "evening" and "night" and four time buckets (i.e. 12 AM to 6 AM, 6 AM to 12 PM, etc.). Among the prediction models used were vanilla SVMs and their variants, along with Convolutional Neural Networks ranging from three layer architectures to deeper networks, namely, AlexNet and VGGNet. The best performing models were the vanilla SVM and the three layer AlexNet (50 and 60 percent accuracy, respectively) suggesting deeper networks that are better equipped to deal with complex features do not necessarily perform better in this particular task.

1. Introduction

Amateur photography has become a novel and popular interest around the world with the emergence of social applications like Flickr, Instagram and Snapchat. These applications allow users to take and upload photographs taken on the spot along with the tags that describe the image. On the other hand, professional photography involves the usage of high-end cameras that record all the information involving location, settings and time taken is also rising in popularity. This information is called the EXIF data which includes ISO, shutter speed, aperture among many other information categories. Perhaps the most insightful and necessary information one can gather from EXIF data is the time of the day when a photograph was taken, as it can allow users to search through their photographs more efficiently as well as provide invaluable data to users and image hosting companies. Most photographs on the internet either don't have the time information or are tagged with the incorrect time.

In this paper, we apply Convolutional Neural Networks to predict when a given input image was taken during the day both using both time windows from EXIF data as well as user submitted time tags. Convolutional Neural Networks have been proven to be efficient in recognizing vision features such as edges, curvatures, corners, etc. We will explore how Convolutional Neural Networks perform in identifying brightness and contrast, among other features, for this task. Specifically, we interpret time in two ways for the purpose of the paper: 1) by tags like morning, afternoon, evening and night and 2) by time bucketing like 00:00 to 6:00, 6:00 to 12:00 etc. The input to our algorithm will be an image which will be classified into a time tag or bucket using SVM and several variants of convolutional neural networks.

2. Previous Work

The premise of inferring time windows or time stamps from images is fairly novel, as such, there is no previous public work to be found that closely aligns to our project. There is however a myriad of related work, such as geospatial location detection. In a collaborative effort between Google and RWTH Aachen University, the publication PlaNet - Photo Geolocation with Convolutional Neural Networks attempted to determine the location where a photo was taken merely from its pixels [1]. The task is straightforward when the image contains a famous landmark or recognizable patterns, but it becomes more complex as abstract features related to weather, markings, architectural details among others need to be inferred. Similarly, we presumed our task would face similar issues, since time detection relies on a variety of factors as well. Their classification approach (they subdivided the earth's surface into thousands of cells) achieved accuracies ranging from $\sim$15% by street ($\sim$1km) to $\sim$42% by country and $\sim$62% by continent. The results improved significantly by introducing LSTMs to solve the problem. Google, however, has access to millions of pictures with extremely accurate location tags, whereas time tags are rarer to find and not very reliable.

Other related work concerns the use of tags on pictures in social media. Though the relationship is not entirely intuitive between the usual tags and tags that we use for this project, one of our approaches relies on self-tagged pictures from a social media platform, Flickr, that give hints to the time (for example, a picture might have the tag "afternoon" or "night" included). In the publication User Conditional Hashtag Prediction for Images (a collaborative effort between New York University and Facebook AI Research), users were modeled by their metadata to perform "hashtag" prediction by photo [2]. The team found that the addition of user information gave a significant performance boost. They found, however, that when working on real world datasets rather than curated ones, the highly skewed natures of "hashtags" needs to be addressed by downsampling the more frequent hashtags to produce more varied predictions. While modeling Flickr users is beyond the scope of this project, this conclusion led us to the hypothesis that introducing related but rarer image tags (along with the common "afternoon", or "morning" ones) would allow us to gather a more diverse dataset as well.

3. Dataset

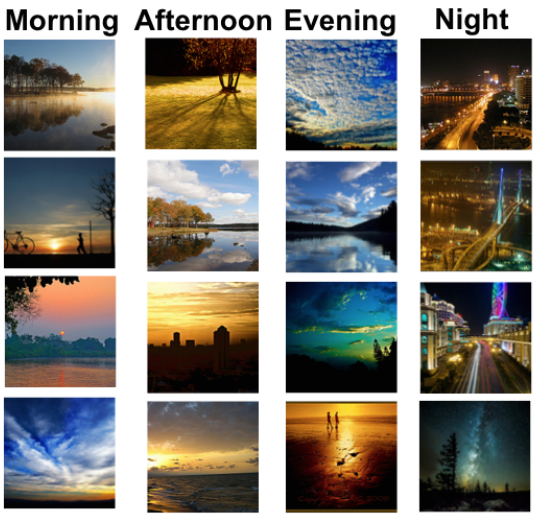

Figure 1: Sample data from the dataset

The dataset of images was collected from Flickr using the Flickr API. We collected a dataset of 3766 images which are all the images on Flickr that contained EXIF data and were relevant to the scope of the project. All images had 3 channels, Red, Green and Blue and were 150x150 pixels large. Figure 1 shows sample data from our collected dataset. We originally intended to gather all of the most recent images related to the corresponding tags to generalize our algorithm but we realized that many of the images in the dataset (for example, of close-up shots of food or faces) would not be suitable for our purposes. There were many grayscale images that would most likely only introduce noise to our model, as such, we had to filter them out. Additionally, many pictures taken indoors did not clearly correspond to the time tags presented. In keeping with our hypothesis that introducing additional tags leads to a more diverse dataset, we experimented with tags such as "outdoor" and "sky", though ultimately we reverted back to the original ones (namely "morning", "afternoon", "evening", and "night") since we decided to be consistent with the tags used for collecting the datasets. The images were sorted using the Flickr API sorting tag "most interesting" descending, as we observed that these were usually more vivid and accurate portrayals of the time tags they depicted.

Table 1: Data distribution based on image tag

Image Tag	Count
Morning	989
Afternoon	759
Evening	990
Night	1,028
Total	**3,766**

Table 2: Data distribution based on time window

Time Window	Count
[12am, 6am)	367
[6am, 12pm)	930
[12pm, 6pm)	933
[6pm, 12am)	1,536
Total	**3,766**

We approach two image classification problems: time window and tag. In the time window problem, the goal is to classify the time window in which the photo was taken, where we have the four time windows [12am, 6am), [6am, 12pm), [12pm, 6pm), and [6pm, 12am). In the tag problem, the goal is to classify the tag which was used to search and collect the image, where we have the four tags "morning", "afternoon", "evening", and "night". Table 1 illustrates the data distribution by image tag and Table 2 illustrates the data distribution by time window.

3.1. Pre-processing

We preprocessed the images to increase the accuracy of our models. The two techniques that we used were data augmentation and application of adaptive histogram equalization on the dataset to make a new dataset.

3.1.1 Data Augmentation

Figure 2: Result of flipping an image horizontally

To expand our training set, we used a traditional method of data augmentation. Originally the training set had 2974 images, and we doubled the training set to 5948 images by flipping the images horizontally. Flipping the images horizontally can help prevent a model from biasing towards one side of an image. Figure 2 shows an example of an image after being flipped for reference.

3.1.2 Adaptive Histogram Equalization

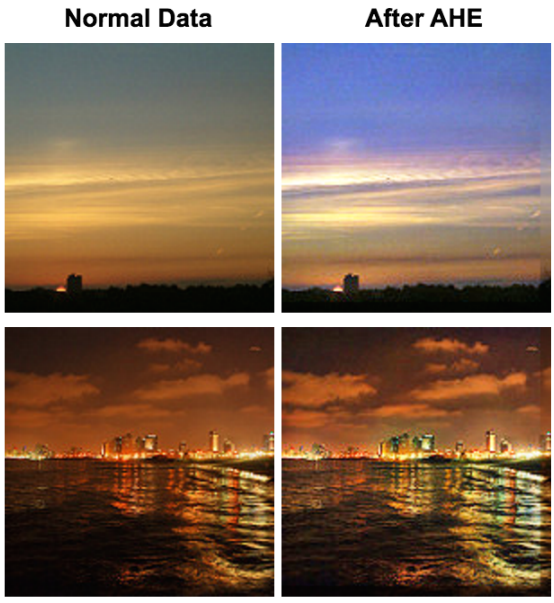

Figure 3: Result of applying Adaptive Histogram Equalization

The dataset contained similar images taken in different dynamic ranges. This problem can be solved by manipulating the histogram of the images. Manipulation of histogram will change the density function which results in a a better dynamic distribution [3]. We applied adaptive histogram equalization (AHE) to all the images in our original dataset to make a new dataset. The adaptive histogram equalization uses multiple histograms in different sections of an image to enhance the contrast of the image. Algorithm 1 describes adaptive histogram equalization and the result can be seen in Figure 3.

Algorithm 1 Adaptive Histogram Equalization

1: **procedure** ADAPTIVE HISTOGRAM EQUALIZATION
Read input image as img
2: Convert img to HSV color space
3: Run CLAHE algorithm on the V (Value) channel of img
4: Convert img back to RGB space
5: **return** img
6: **end procedure**

Contrast Limited Adaptive Histogram Equalization (CLAHE) mentioned in Algorithm 1 is an processing technique to improve contract in images. We used the sci-kit-image library to perform the adaptive histogram equalization for this project [4].

4. Approach

We used TensorFlow along with other python packages such as numpy, matplotlib, skimage, etc to code all the experiments in this project [4], [9], [10]. We applied the following models to our dataset to classify images into their correct categories:

1. Multiclass Support Vector Machine (SVM)
2. Multiclass SVM + HOG + HSV
3. Three Layer ConvNet
4. Five Layer ConvNet
5. AlexNet
6. VGGNet

4.1. Multiclass Support Vector Machine (SVM)

The multiclass Support Vector Machine (SVM) is the generalization of ordinary SVM for multiple classes. It uses the following loss function

$$L_i = \sum_{j \neq y_i} max(0, s_j - s_{y_i} + \Delta)$$

where L_i is the loss for the i^{th} example, s_i is the score for the i^{th} class, y_i is the true label for the i^{th} example, and delta is the margin. Our multi class SVM takes in a numpy vector of flattened raw image data of length 150*150*3 = 67500 (150 being the width and height pixels, and 3 being the RGB channels) and outputs the raw score for each class.

The class that the SVM predicts is the one with the highest score. The optimization method is Stochastic Gradient Descent (SGD) using mini batches of data.

4.2. Multiclass SVM + HOG + HSV

This multi class SVM+HOG+HSV is very similar as the vanilla version described above, except that it takes in HOG and HSV features of the image instead of raw image data. Conceptually, HOG captures texture but no color information from the image, while HSV captures color but no texture information from the image. By extracting these features independently and then concatenating during training time, we obtain a richer feature landscape.

4.3. 3-Layer ConvNet

The architecture of the 3-layer ConvNet consists of three sections. The first section consists of a convolutional layer, followed by a ReLU activation, and ending with max-pooling. The second section is the same as the first. The third section consists of a fully-connected layer, followed by a ReLU activation, and ending with a linear affine. The convolution layers use 32 5x5 filters with a stride of 1. Max-pooling is 2x2 (which essentially halves the planar dimensions) with a stride of 2. The fully-connected layer has 1024 nodes. For training, SGD with Adam optimization is used, and dropout is used for regularization [5],[6].

Adam is the state-of-the-art gradient update rule for ConvNets. It combines elements from RMSProp and momentum update. Dropout is a regularization technique that helps prevent ConvNets from overfitting. The idea that during each a training step, a random group of neurons are disabled, which helps prevent neurons from co-adapting (i.e. developing an overly strong dependence on one another). The 3-layer ConvNet takes in a raw image as a 150x150x3 dimensional array and classifies the input image to one particular tag.

4.4. 5-Layer ConvNet

The architecture for our 5-layer ConvNet is the similar to the 3-layer ConvNet, except there are two more [conv - relu - pool] layers appended. The parameters for the convolutional layer, max-pooling, and fully-connected layer are the same, and SGD with Adam optimization and dropout are used as well[6].

4.5. AlexNet

We use AlexNet as presented in this paper [7]. A unique feature of AlexNet is that is uses local response normalization to normalize the "brightness" of neurons. The local response normalization use the following equation to normalize the brightness of the neurons:

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

Where $a^i_{x,y}$ denotes the activity of a neuron computed by applying kernel i at position (x, y) [7].

The AlexNet consists of four sections. The first, second, and third sections consist of a convolutional layer, followed by a ReLU activation, max-pooling, and ending with local response normalization. The fourth section consists of a fully-connected layer, followed by a ReLU activation, and ending with a linear affine to obtain the class scores.

Max-pooling is 2x2 with stride 2 throughout the AlexNet (which essentially halves the planar dimensions at each step). The first convolutional layer has 64 filters, the second 128 filters, and the third 256 filters, where the filters are of size 3x3 with stride 1. The fully-connected layer has 1024 nodes. For training, SGD+Adam optimization is used along with dropout.

4.6. VGGNet

VGGNet uses very small convolution filters (3x3), which allows the depth to be increased with less overhead than if it used larger filters [8]. The VGGNet consists of 8 sections. The first 5 sections consist of two pairs of convolution layers and a ReLU activation, which are followed by max-pooling. The last 3 sections consist of fully-connected layers.

Max-pooling is 4x4 with a stride of 4 in the first section and 2x2 with a stride of 2 in the other sections. The convolution layer has 3x3 filters throughout the VGGNet, while the number of filters varies per section. The number of filters are 64, 128, 256, 512, and 512 for the first five sections respectively. The number of neurons in the fully-connected layers are 4096, 10, and 4 respectively for the last three sections. For training, SGD+Adam optimization is used along with dropout.

5. Results and Discussion

We experimented with methods like SVM, SVM+HOG+HSV, 3 and 5 layer Convolutional Neural Network, AlexNet and VGGNet. One of us also manually performed the task to get a measure of human performance for this task. The person trained on around 400 images and predicted the tags (i.e. morning, afternoon, evening, night) for around 200 images. Several aspects of the data make this task very difficult, which was made obvious when the person scored a test accuracy of around 40%.

We implemented the convolutional neural networks using the recently-released framework TensorFlow [9]. We used a keep-rate of 75% for dropout, and we used learning

Table 3: SVM results with tag classification

Model	Train	Val	Test
SVM	0.525	0.47	0.475
SVM + HOG + HSV	0.426	0.458	0.428

Table 4: SVM results with time bucket classification

Model	Train	Val	Test
SVM	0.517	0.455	0.453
SVM + HOG + HSV	0.371	0.383	0.343

rates ranging from 1-e2 to 1-e4. For all of the models, we used the same training, validation, and test sets, where the sizes are 5948, 400, and 400, respectively. The sets were obtained by randomly shuffling the dataset and partitioning.

5.1. SVM

The vanilla SVM had a test and validation accuracies of about 47 percent for tag classification task, and about two percent lower for bucket classification as shown in Table 3 and Table 4 respectively. The SVM+HOG+HSV performed significantly lower, which suggests that isolating texture and color is not beneficial for the time inference tasks.
Figure 4 is the loss function for SVM applied to tag classification task. It converges after ∼250 iterations.

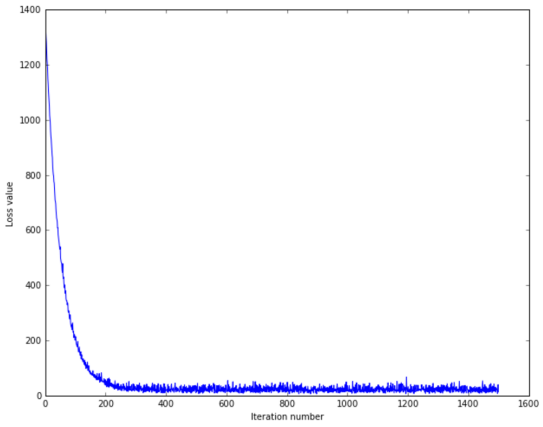

Figure 4: SVM loss for tag classification

Figure 5 is the confusion matrix for the multiclass SVM applied to tag classification problem. The horizontal axis represents the predicted labels, and the vertical axis represents the actual labels. The color of the square indicates the number of examples that have the vertical label that were classified as the horizontal label. In this instance, the SVM confused evening for night in 90/400 examples, and confused night for evening and afternoon. It did the best for the night label, mediocre for evening, and subpar for morning

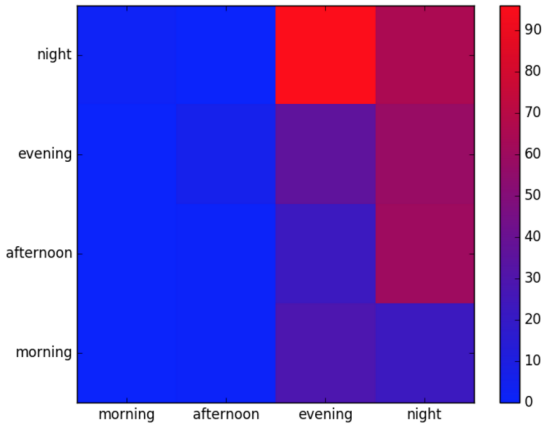

Figure 5: Confusion matrix for SVM applied to tag classification

Table 5: SVM on adaptive histogram equalized dataset

	Train	Val	Test
SVM	0.504	0.495	0.5125

Table 6: Results of 3-Layer ConvNet

	Train	Val	Test
Tag Classification	0.44	0.41	0.4
Time Bucket Classification	0.46	0.43	0.41

Table 7: Results of 3-Layer ConvNet on adaptive histogram equalized dataset

	Val	Test
Tag Classification	0.45	0.455
Time Bucket Classification	0.4175	0.415

and afternoon. The vanilla SVM performed best with the adaptive histogram equalized dataset, with a validation and test accuracy of about 50 percent as shown in Table 5. This was the second-best accuracy achieved from all the models.

5.2. 3-Layer ConvNet

We trained the 3-layer ConvNet for 10,000 iterations with a mini-batch size of 20. The ConvNet performed better on the Adaptive Histogram Equalized dataset than the normal one. In particular, test accuracy increased by 5.5% for tags. Table 6 shows the results of 3 layer on both tag classification and bucket classification problem. Results of the 3-layer ConvNet are shown in Table 7.

Table 8: Results of 5-layer ConvNet

	Val	Test
Tag Classification	0.47	0.4675
Time Bucket Classification	0.3425	0.35

Table 9: Results of 5-layer ConvNet on adaptive histogram equalized dataset

	Val	Test
Tag Classification	0.49	0.465
Time Bucket Classification	0.3425	0.35

5.3. 5-Layer ConvNet

We trained the 5-layer ConvNet for 100,000 iterations with a minibatch size of 20. The ConvNet performed equally well on the normal and adaptive histogram equalized datasets. Table 8 shows results of 5-layer ConvNet on the normal dataset and Table 9 shows its performance on adaptive histogram equalized dataset. It performed better than the 3-layer ConvNet with tags as expected. Comparing its performance on the two datasets, it does equally well on both the datasets.

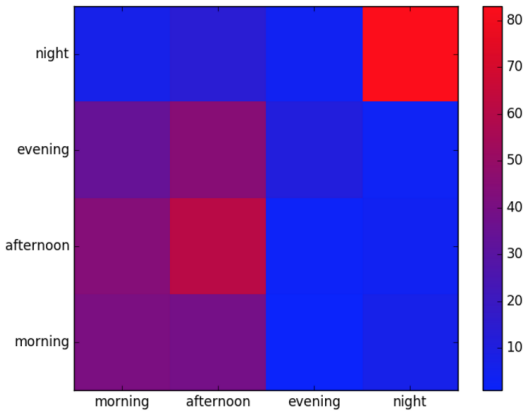

Figure 6: Confusion matrix for 5-layer ConvNet on tag classification with adaptive histogram equalized dataset

Figure 6 the confusion matrix for the 5-layer convnet. It classifies night correctly almost all the time, does well on afternoon, average on morning, and subpar on evening. It confuses afternoon for morning and vice versa

5.4. AlexNet

AlexNet was the most fine-tuned and trained model from the bunch, and is the only model to score test and train accuracies above 50. The learning rate was 0.0012, the batch size was 20 and the training iterations were 20,000. Table 10 shows the results of AlexNet on the tag classification task. AlexNet did not benefit from the adaptive histogram equalization as the accuracies were very close to human accuracy as shown in Table 11.

Table 10: Results of AlexNet

	Val	Test
Tag Classification	0.55	0.6

Table 11: Results of AlexNet on adaptive histogram equalized dataset

	Val	Test
Tag Classification	0.4425	0.42
Time Bucket Classification	0.4	0.42

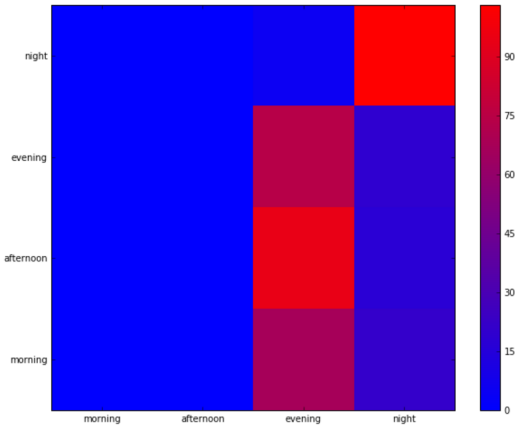

Figure 7: Confusion matrix of AlexNet

Figure 7 the confusion matrix for the AlexNet. Like the 5-layer net, the AlexNet does very well for night. It does well for evening, but does subpar for morning and afternoon. It tends to confuse evening as afternoon and morning.

5.5. VGGNet

VGGnet was the weakest performer of all the models, with accuracies as 0.35 for training set, 0.3 for validation set and 0.33 for test set. This is likely due to the VGG being a more suitable architecture for complex and feature sensitive tasks. In addition, it was extremely tricky to tune the VGG, as the range between under and overfitting was very narrow, so our results may not have reached an optimal level. Table 12 shows the results for VGGNet on adaptive histogram dataset.

5.6. Discussion

All the models perform equal or better as compared to the human accuracy. During the project, we came across

Table 12: Results of VGGNet

	Train	Val	Test
Tag Classification	0.35	0.3	0.33

several noticable insights which can effectively explain the difficulty of the task and the behavior of the models.

5.6.1 Errors in EXIF data

Since the data was collected from Flickr, which is a public image portal, it contained some disparities with the provided and most likely "photo taken" time.

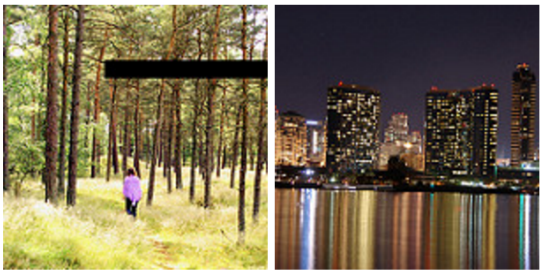

Figure 8: Example images with EXIF errors

In Figure 8, the left image seems to be taken during the day, but the provided time was 00:00:00, which is midnight. The right image seems to have been taken at night but the provided time was 11:37:19, which is around noon time. The cause of these errors cannot be properly deduced , as they may vary from photo to photo. One of the possibilities is that the time obtained from the device could be in a different time zone or may have a miscalibrated system time. The user could have as well misreported or edited the time taken at upload time or afterwards. Furthermore, such errors cannot be manually corrected, even on a small scale, as we don't know the actual time when the image was taken.

5.6.2 Ambiguity in Categories

This task is really difficult even for humans due to the ambiguity between categories. For example, it is very easy to confuse morning and evening images as both have a very similar sky color palette, brightness and contrast due to how similar sunrises and sundowns are. Figure 9 show the ambiguity between images from morning and evening. Furthermore, some of the photographs such as in Figure 10 were verging on the border of two categories, such as evening and night or morning and afternoon. Since the convolutional neural network needs to pick one, it becomes a significantly hard choice to make.

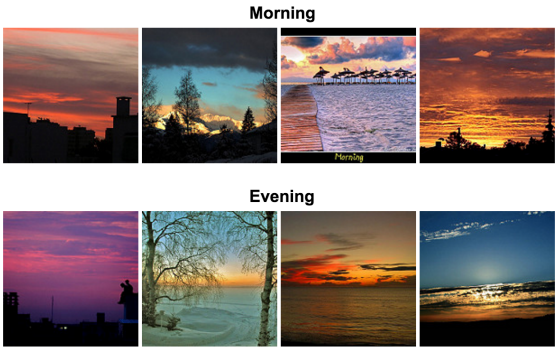

Figure 9: Example images showing ambiguity in categories

Figure 10: Images on the border of evening and night

5.6.3 Edited photographs

Figure 11: Examples of edited photographs in the dataset

Many of the photographs on Flickr are preprocessed in some way for aesthetic purposes, which causes a loss of original information and a distortion between perceived and actual time or category. This distortion in data makes the task of predicting the correct time trickier and might induce a bias towards a particular bucket or label. The images in Figure 11 were taken in the evening but the photographers seem to have toned down the brightness of the image, causing the network to misclassify the images as having been taken at night.

6. Conclusion

In this project, we attempted to predict time bucket or tag for when an image was taken. The best performing algorithms were Support Vector Machines and AlexNet, not the deeper network, the VGGNet, that we assumed would outperform most other models. This suggests that for our dataset, basic feature metrics such as color palettes were much more valuable in predicting the time taken for a picture as opposed to subtler and more complex detailed features that deeper networks are best known for. The most limiting challenge was the dataset, which was not as diverse as we had hoped and contained some glaring mislabelings.

Future work should involve collecting correctly labeled dataset which has accurate EXIF data with time adjusted to the local timezone where the image was taken. It would be interesting to use a model which is pretrained on the places dataset from MIT [11]. In this project we tested individual models on our dataset, it will be interesting to train an ensemble of models on the dataset. The images taken at the same time would be different for different locations. For example, a photo taken at 7 AM in California will be very different from an image taken at 7 AM in Greenland. Weather also plays an important role as a cloudy photograph might be darker and has a good chance of getting classified as evening or night. The dataset can also include geolocation and weather information to better guide the classifier. This problem is very challenging and an open question to the research community. More accurate data with thoughtful strategies can improve the results provided in this paper.

7. References

[1] Weyand, Tobias, Ilya Kostrikov, and James Philbin. "PlaNet-Photo Geolocation with Convolutional Neural Networks." arXiv preprint arXiv:1602.05314 (2016).

[2] Denton, Emily, et al. "User Conditional Hashtag Prediction for Images." Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2015.

[3] Zhang, Hong et al. "Perceptual Contrast Enhancement with Dynamic Range Adjustment." Optik 124.23 (2013): 10.1016/j.ijleo.2013.04.046. PMC. Web. 14 Mar. 2016.

[4] Stfan van der Walt, Johannes L. Schnberger, Juan Nunez-Iglesias, Franois Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014)

[5] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[6] Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." The Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[7] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems. 2012.

[9] Abadi, M. et al. (2015). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

[10] J.D. Hunter, "Matplotlib: A 2D Graphics Environment," Computing in Science and Eng., vol. 9, no. 3, 2007, pp. 90-95.

[11] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. "Learning Deep Features for Scene Recognition using Places Database." Advances in Neural Information Processing Systems 27 (NIPS), 2014.