Geo-Locating Images: Where in the world was this picture taken?

Angela Sy
angelasy@stanford.edu

Cynthia Day
cyndia@stanford.edu

Abstract

Our project focuses on geo-location for a subset of the 2015 MediaEval Placing dataset and produces comparable performance to existing submissions on the classification task and superior performance on the regression task. We restrict our training set to images only, and attempt to extract sufficient information using a Convolutional Neural Network (CNN). We tackle two image location tasks – first a classification task to predict the country, and second a regression task to predict the exact coordinates. For the classification task, we use as our baseline simple feature extraction followed by an SVM, and for our final runs experiment with different sets of pre-trained models and CNN architectures. These methods yield results slightly above baseline. For the regression task, we use pre-trained models to predict latitude and longitude coordinates. This approach shows great improvements on the baseline.

1. Introduction

For our project, we focus on the locale-based subtask of the 2015 MediaEval Placing task [8]. The dataset includes images, videos, and textual metadata, as well as pre-extracted features from each media source. We are interested in the amount of geographic information that can be regained from solely the images. This challenge is especially interesting because the images, stripped from social media sites like Flickr, lack the iconic landmarks and broad street view that are common in datasets on which geo-location is performed. Yet automatic geo-tagging of such images would be useful for many services.

We will investigate the use of CNNs on the classification task of country prediction and the regression task of latitude and longitude prediction. We will experiment with training a CNN from scratch versus fine tuning on pre-trained CNN models.

2. Related Work

While geo-location has gotten a great deal of attention lately, the research tends to focus on specific types of data. For example, over the past decade, several papers have been published on landmark-based classification. These papers focus on building a strong dataset around each landmark and then employing clustering techniques [1, 10, 13]. Another common focus is classifying street view images for select cities. These papers place a strong emphasis on feature detection [2, 6] and use methods like nearest neighbor matching [12].

In contrast, the Flickr dataset provided by MediaEval contains a general set of images that include landmarks, objects, and people, creating a significantly harder geo-location problem. The previous participants of the MediaEval Placing competition used approaches like clustering, SVMs, and k-nearest-neighbor, and achieved very good performance when utilizing the photo textual metadata. In those cases, accuracy within 100 km was as high as 54.33% [7], and as high as 69.33% for 1000 km [5].

However, when the participants performed runs on the visual data only, accuracies dropped enormously. Results were as low as 5.47% of correct location predictions within 1000 km [10]. The team that had the best results on visual data, with 9.07% of results lying within 100 km of the truth and 23.98% of results lying within 1000 km of the truth, was also the only team to use CNNs in their approach [7]. This serves as our inspiration to investigate the potential gains of CNNs for this geo-location task.

Recently, Google released a paper addressing a task very similar to ours, geo-locating general images using a deep network and an LSTM architecture. Because of their superior dataset size and computing resources, they are able to subdivide the earth into thousands of regions and classify over these regions. While it initially appears that the increased number of classes makes this a harder problem, it is also interesting to note that with smaller regions, the variance within each region decreases greatly. Thus, this approach seems like a fruitful one for future work.

3. Methods

3.1 Baseline

As a simple baseline, we extracted HOG (Histogram of Oriented Gradients) features for each image, then used these features to train a multi-class SVM. We chose this feature set because it was the most effective out of the ones that we investigated for Assignment 1.

3.2 Vanilla CNN

Our vanilla CNN takes as input 48x52 sized images. Its architecture consists of three double-convolution plus max pooling layers followed by three fully connected layers with 256 hidden units each. The architecture can be seen below.

INPUT →[[CONV -> RELU] *2 -> POOL]*3 → [FC → RELU]*2 →FC → SOFTMAX

Initially we tried an architecture that performed a max pooling step after each convolution layer. This produced poor accuracy results on both training and validation sets. This is probably due to the already small size of input images. The pooling step discards too much information early on by removing a large percentage of the image's features before the next convolution layer, preventing the CNN from learning deeply about those features before they are discarded. Because of this, we moved to the current architecture which contains two convolution layers before the max pooling step. This preserved more information about the images and produced better accuracy results on both training and validation sets.

We used 32 filters of size 3x3. We chose this smaller filter size in order to extract as many features as possible from the small images being fed to the CNN.

Each convolution and fully connected layer is followed by a ReLu activation function to accelerate convergence during training. The final layer is capped with a softmax classifier to predict the class label given the scores for each class. We use categorical cross-entropy loss to update the network's weights after each training batch.

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Fig. 1: Equation for Cross-Entropy Loss

The relative simplicity of this model allowed us to implement in Lasagne [14] and train locally.

3.3 Pre-trained Models

We reduce computation time by applying two common pre-trained models, GoogLeNet and VGG 16 Net, to extract features from our dataset. To be more specific, we initialize the pre-trained model with its known weights, and run each of our images through the model a single time, pulling out the input to the model's final softmax layer as our feature vector for the image. In this way we rely on the pre-trained models to be able to pick out the most interesting or characteristic features of the raw images. This has the advantage of reducing the dimensions of each input image to a 1 x 1000 vector, allowing us to process many more images at full resolution.

After retrieving features for each image, we fine-tune these features through several dropout and fully-connected layers to adjust their performance for our classification problem. The fine-tuning architecture is as follows:

INPUT → [DROPOUT → FC]*3

The first dropout layer is 20%, while the next two are 50%. These dropout layers help prevent our model from over-fitting. The intermediate fully-connected layers each have 800 hidden units, and allow us to fine-tune the extracted features to suit our task. Although our classification and regression tasks are different from the original tasks on which the pre-trained models are based, we make the assumption that the features picked out by the models will still be relevant to our task. If we suspected that this were not the case, we could extract activations higher up in the pre-trained model, and add a few convolution layers to our fine-tuning. For now, we leave this as an area of future investigation.

For the country classification task, we add a final softmax layer and use categorical cross-entropy to calculate loss. For the regression latitude/longitude prediction task, we have the final fully-connected layer output a coordinate pair and use a squared-error loss.

For this model, we used the GoogLeNet and VGG 16 Net models provided by Lasagne/Recipes [9]. We ran the models on K520 GPU, via an Amazon g2 instance.

4. Dataset and Features

4.1 Dataset

Our full dataset contains 4,672,382 images from the MediaEval challenge. For this project, we are using a subset of the data – specifically 163,486 images. The MediaEval images were taken from Flickr and vary widely in appearance, ranging from iconic buildings to pictures of people and flowers. The images are all around 500x300 pixels.

We chose a smaller data set to make training manageable as well as to test the efficacy of our CNN methods. Future work would include running the same methods on the full data set. We split our data into 80% training, 10% validation, and 10% test sets.

4.2 Data Pre-Processing

For the vanilla CNN, we compressed the images to a size of 48x52. This seemed reasonable given the accuracy of object classification on the CIFAR10 dataset of 32x32 size images. Grey scale images which do not have 3 color channels and provide inadequate information about the image were discarded.

We also pre-processed the images by using mean subtraction and normalization. We subtracted the mean across every individual feature in the image data, centering our data around the origin. After zero-centering our data, we also normalized the data dimensions by dividing each dimension by its standard deviation to produce features of the same scale.

Our pre-trained models expect input of 224x224, so we began by compressing the smaller dimension of our image to 224. We then took a centered 224x224 crop of this for our final image. We subtracted the mean values (104,117,123) of the images on which the models were originally trained, so that our images would undergo the same pre-processing as the original images used on the pre-trained models [9].

4.3 Evaluation

Each image is labeled with a set of geographic labels ranging from coarse (country) to exact (latitude and longitude coordinates). We began with trying to predict the country label for each image. Additionally, because the dataset has images from over 200 different countries, and there is high variance in the number of images between each country, we selected 5 countries with similar numbers of images to avoid biased training of our CNN. The images we chose for our dataset are roughly equally distributed across Germany, Canada, Italy, Spain, and France.

After experimenting with a number of different models on the classification problem, we will use the best-performing models on the more exact problem of latitude and longitude coordinates prediction. We approximate the distance in km between the predicted coordinates (x1, y1) and the true coordinates (x2, y2) using the equirectangular approximation of

$$R \cdot \left\| \left\langle (x1 - x2) \cos \frac{y1 + y2}{2}, y1 - y2 \right\rangle \right\|$$

where R is the radius of the Earth (6371 km) and the coordinates are given in radians [4].

4.4 Expected results

We expected this to be a hard problem. Upon inspecting the dataset, we saw that many images, being stripped from social media, lack the context one would expect for successful geo-location. Indeed, the best performance on visual data is within 100 km (the radius of a small country) only 9.07% of the time [1]. Thus, even when using coarse country labels, we should expect performance only slightly above that of random guessing.

5. Experiments and Results

5.1 Baseline: Multi-Class SVM

To get a simple baseline, we decided to make use of an approach that mirrored the approach of one of the teams that competed in the original MediaEval 2015 challenge [3]. We extracted HOG features from the images after compressing the images to 150x90 due to memory limitations. We than ran these features through a multi-class SVM. We used a subset of 1880 images spread over five countries for our training set, and set aside a test set of roughly 10% this size, with the same spread over the countries.

We achieved a validation accuracy of 20.99%. This is very close to the figure that would result from random guessing. This makes sense since the teams that did not use convolutional neural networks for the MediaEval challenge got very poor results when restricting runs to visual data only.

5.2 Vanilla CNN

5.2.1 Training

In training our CNN, we maximized accuracy by tuning hyper-parameters and settled on the following values:

- learning rate of 0.045
- regularization strength of 0.01
- batch size of 500
- 20 epochs

We initially chose a lower learning rate but found that a higher and more aggressive learning rate was necessary for the CNN to process enough features from the small images. We chose to train with a batch size of 500 images. After trying different batch sizes, we chose this as the appropriate batch size since it balanced the amount of "wiggle" or variation in the calculated loss. After about 15 epochs, we see the loss and the training accuracy converging and so we stop training at 20 epochs.

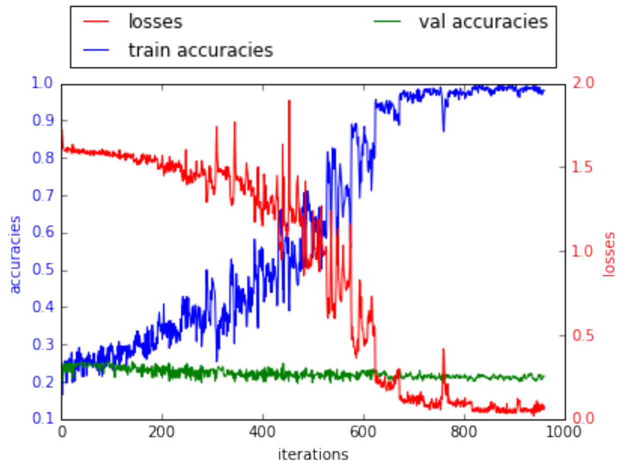

Fig. 2: Vanilla CNN – Loss, Training and Val Accuracy Using Nesterov Momentum update for 20 epochs

In Figure 2, we see a steady increase in training accuracy peaking at close to 100% as desired. Meanwhile, the validation accuracy remains very low, with a maximum of about 24%. In addition, the loss decreases steadily as the CNN is trained.

Our initial validation accuracy was around 22.3%. From these results, it appears that our

CNN is over-fitting on the training set. To mitigate this, we regularized the CNN weight initializations and added regularization strength, but ultimately accuracy still remained quite low at close to 24%. In future work, we will experiment with adding spatial batch normalization layers to further regularize the network.

5.2.2 Optimization

We initially chose the Nesterov Momentum update technique to train the RNN until convergence. However, this optimization method still seemed relatively slow and the CNN required at least 15 epochs before loss converged. We attempted a different optimization method that uses a per-parameter adaptive method to train the CNN. In particular, we chose the Adam method to adaptively tune learning rates for every parameter.

The results of our CNN using the Adam update are shown in Figure 3 below. Note that the loss and accuracy converge extremely quickly. Within the first 50 iterations (1 epoch), the loss jumps to close to 0 and the validation accuracy reaches about 25% and remains in that range until the end of training.

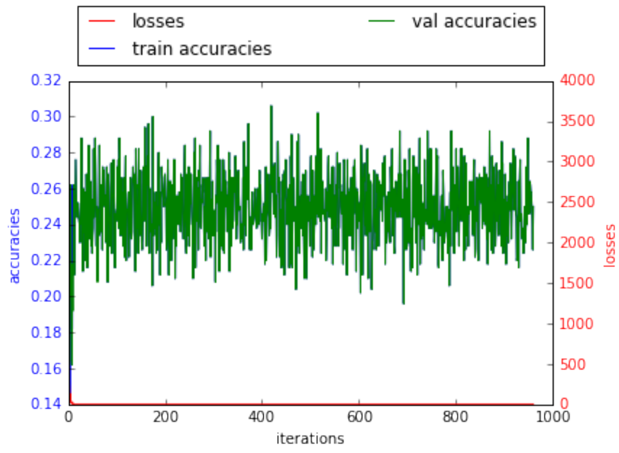

Fig. 3: Vanilla CNN – Loss, Training and Val Accuracy Using Adam update for 20 epochs

5.2.3 Results

Classification on the vanilla CNN performed almost twice as well as the baseline of 9.07%. We reached a test accuracy of 22.3% using the Nesterov Momentum update and 23.4% using the Adam update.

5.3 Pre-trained Models: Country Classification

5.3.1 Training

For both GoogLeNet and VGG 16 Net, we used the following parameters:

- learning rate of 0.001
- batch size of 500
- 200 epochs
- Nesterov momentum of 0.9

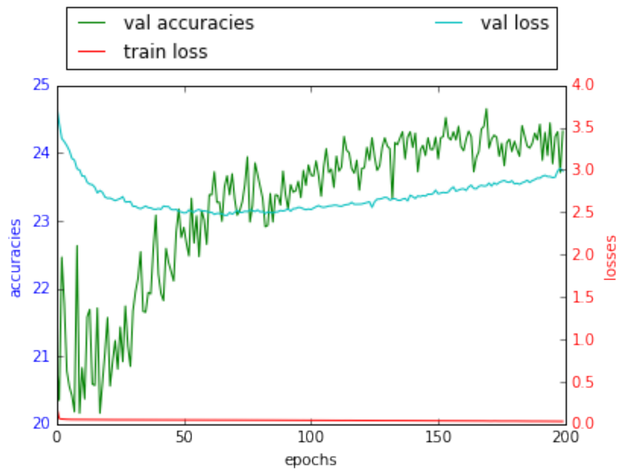

Fig. 4: Pretrained GoogLeNet – Loss and Val Accuracy for classification task

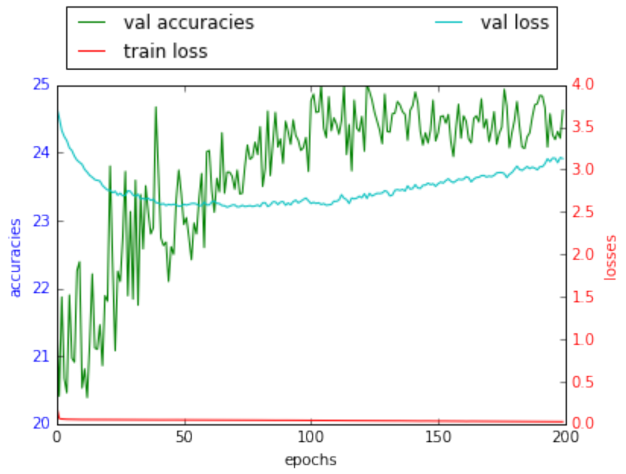

Fig. 5: VGG16 Net – Loss and Val Accuracy for classification task

As can be seen above in Figures 4 and 5, the validation accuracy converged under these parameters, as desired. Hence, we can use these models to test our classification accuracy.

5.3.2 Results

GoogLeNet gave a final test accuracy of 24.3% and VGG 16 gave a final test accuracy of 23.8%. Overall, both pre-trained models performed better than the random baseline. The differences between the performance of the two models seem too small to be statistically significant.

5.4 Country Classification Model Comparison

Using the methods described earlier, we compared classification accuracy results on the test set for each method, namely the SVM, Vanilla CNN, VGG Net, and GoogLeNet.

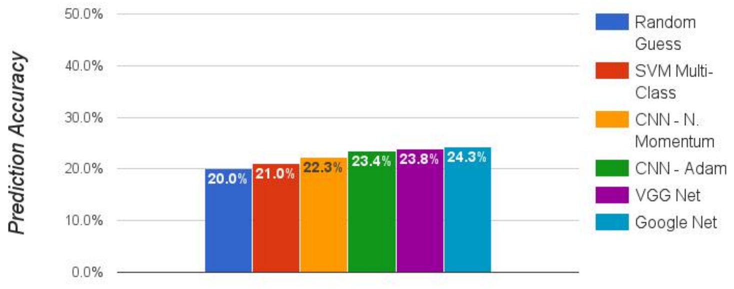

Fig. 6: Classification Prediction Accuracy Comparison

The best results came from our pre-trained models. This was expected as the pre-trained networks have been trained on other images and thus produce features that have more useful information for our CNN to train on. Additionally, because of the increased efficiency of the pre-trained models, we were able to use additional data and train over more epochs. Thus, we chose to use the pre-trained models for the regression task of predicting latitude and longitude coordinates for each image.

5.5 Pre-trained Models: Latitude and Longitude Prediction

5.5.1 Training

For both GoogLeNet and VGG 16 Net, we used the following parameters:

- learning rate of 0.0001
- batch size of 500
- 500 epochs
- Nesterov momentum of 0.9

With this regression problem, we found the initial loss to be very high with slow convergence, as visualized in Figure 6. Thus, we chose to use the highest learning rate that did not cause divergence in the training loss. Additionally, since epochs are relatively inexpensive for this architecture, we were able to run for a large number of epochs. While the training loss did not fully converge within 500 epochs, we found the accuracies to be stagnant over the last hundred epochs, and thus ended training at that point.

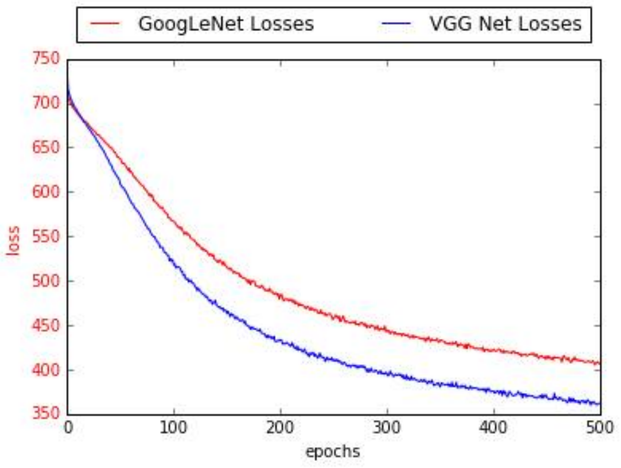

Fig. 7: Loss for regression task with pre-trained models

5.5.2 Results

In the style of the MediaEval papers, we evaluated our performance by counting the percentage of images whose predictions were within 10 km, 100 km, 1000 km, and 10000 km. We use a random baseline obtained by evaluating the test accuracy at 0 epochs. This baseline places 20-25% of images within 1000 km of the true location, which is also the accuracy achieved by the best MediaEval submission [7]. Thus, this seems like a reasonable baseline.

The prediction accuracy of our trained CNN is twice as accurate compared to the baseline. In both pre-trained models, over twice as many test images fall into the 100 km and 1000 km buckets after 500 epochs of training compared to the baseline performance at 0 epochs. This is demonstrated visually in Figure 8. Note that we only show results for the 1000 km bucket because in all cases, less than 1% of images fall into the 10 km and 100 km buckets.

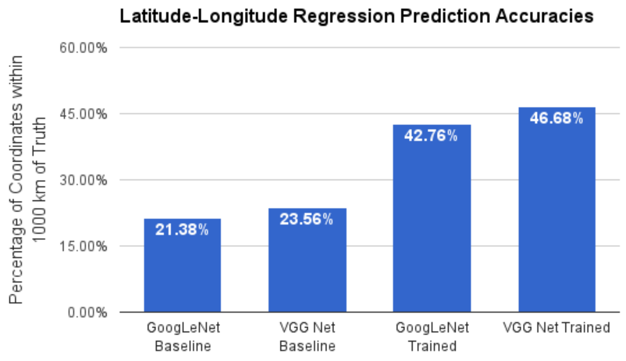

Fig. 8: Prediction Accuracy of Image Location Coordinates within 1000 km of actual image location

In our best performance on the test set, 46.68% of our images lie in the 1000 km bucket. For comparison, only 23.98% of images for the best MediaEval competition submission fell into the 1000 km bucket. In addition, only 53.6% of images for Google PlaNet fell into the 750 km bucket. While these numbers cannot be directly compared to our results since our images were drawn from a subset of five countries which included some countries that were close neighbors, these numbers still provide some assurance that our performance is comparable, if not superior, to the MediaEval performance.

5.6 Qualitative Results

We look at some examples of correctly and incorrectly classified images to understand the strengths and potential improvements to our model.

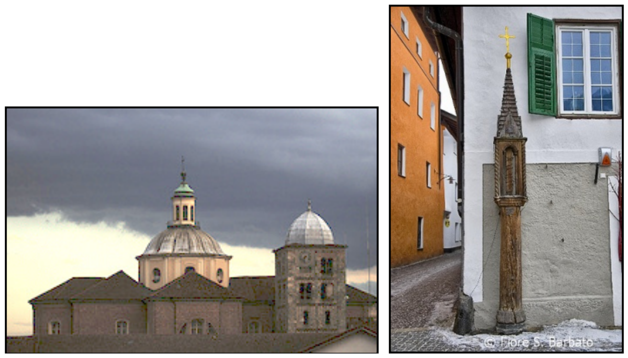

Fig. 9: Correctly Classified Images
Left: Italy, Right: Spain

As we can see from the images above, our models successfully classify the location of images that contain landmarks and iconic objects. In the image on the left, the rounded dome with its distinctive light blue green color is an easily identifiable depiction of an Italian cathedral. Similarly, the image on the right shows an iconic post with a golden cross on top, a common street-side sight in Spanish cities. In addition, the dark green shutters and orange walls along the sidewalk are additional markers that the scene is an image in Spain and act as strong indicators for the CNN.

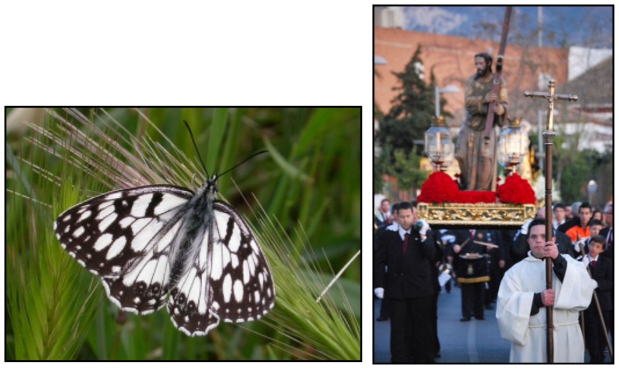

Fig. 10: Incorrectly Classified Images
Left: Correct Germany, Predicted Spain
Right: Correct Spain, Predicted Italy

Two examples of incorrectly classified images are seen above. The image of the butterfly is difficult to classify because of the focus on a single object without location-specific context. Since our dataset is from a set of images uploaded on Flickr, many of the images in our dataset are similar to the butterfly, depicting close-ups of specific objects that are not easily identifiable, even by an oracle.

Images common across countries are also difficult for our CNN to classify. The second image in Figure 10 shows a procession of people wearing religious garb and carrying many religious icons. These symbolic objects are common in pictures taken in predominantly Catholic countries like Spain and Italy. Thus, it is difficult for the CNN to differentiate between the locations of these common symbolic images when taken in multiple locations.

6. Future Work

Due to limitations in computing resources, we work with only a subset of the full dataset in this paper. This limited our ability to make fine-grained coordinate predictions, as evidenced by the percentages for the 100 km bucket which were below benchmark compared to the percentages for the 1000 km bucket which were much better than the best MediaEval submission.

To further improve results, data augmentation could also be useful. Currently we take the center crop when using pre-trained models, and thus lose some information from each image. Sampling random crops and scaling would increase the likelihood that we capture specific landmarks or objects with distinctive geographic features.

We can also improve our results by using more advanced CNN architectures. For example, we can upgrade our vanilla CNN to a more complicated architecture such as ResNet and see what accuracy gains it supplies. For the pre-trained models, we can use activations from an earlier layer as our features, then manually add convolutional layers for fine-tuning. If implemented, both of these options would demand higher computing resources.

References

[1] Y. Avrithis, Y. Kalantidis, G. Tolias, and E. Spyrou. Retrieving Landmark and Non-Landmark Images from Community Photo Collections. In ACM Multimedia, 2010.

[2] D. Chen, G. Baatz, K. Koser, S. Tsai, R. Vedantham, T. Pylvan, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In CVPR, 2011.

[3] N. Duong-Trung, M. Wistuba, L. Drumond, and L. Schmidt-Thieme. Geo_ML @ MediaEval Placing Task 2015. *MediaEval 2015 Workshop*, 2015.

[4] Calculate distance, bearing and more between Latitude/Longitude points. Moving Type Scripts. http://www.movable-type.co.uk/scripts/latlong.html

[5] P. Kelm, S. Schmiedeke, and L. Goldmann. Imcube @ MediaEval 2015 Placing Task: A Hierarchical Approach for Geo-referencing Large-Scale Datasets. *MediaEval 2015 Workshop*, 2015.

[6] H. J. Kim, E. Dunn, and J.-M. Frahm. Predicting Good Features for Image Geo-Localization Using Per-Bundle VLAD. In ICCV, 2015.

[7] G. Kordopatis-Zilos, A. Popescu, S. Papadopoulos, and Y. Kompatsiaris. CERTH/CEA LIST at MediaEval Placing Task 2015.*MediaEval 2015 Workshop*, 2015.

[8] MediaEval Benchmarking Initiative for Multimedia Evaluation. 2015 Placing Task: Multimodal geo-location prediction. http://www.multimediaeval.org/mediaeval2015/placing2015

[9] E. Olson, Lasagne recipes: examples, IPython notebooks, ... https://github.com/ebenolson/Recipes, 2015.

[10] T. Quack, B. Leibe, and L. Van Gool. World-Scale Mining of Objects and Events from Community Photo Collections. In CIVR, 2008.

[11] T. Weyand, I. Kostrikov, and J. Philbin. PlaNet-Photo Geolocation with Convolutional Neural Networks. *arXiv preprint arXiv:1602.05314*, 2016.

[12] A. R. Zamir and M. Shah. Image Geo-Localization Based on Multiple Nearest Neighbor Feature Matching Using Generalized Graphs. PAMI, 36(8), 2014.

[13] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: Building a Web-Scale Landmark Recognition Engine. In CVPR, 2009.

[14] Code Libraries: scikit-learn, Theano, Lasagne