Princeton University

Computer Science

Learning Fine-Grained Image Similarity Through Weak Supervision

Author:
Daway Chou-Ren

Advisor:
Dr. Szymon Rusinkcewicz

Submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Engineering
Department of Computer Science
Princeton University

May 2, 2017

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Daway Chou-Ren

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Daway Chou-Ren

Acknowledgements

Contents

Chapter 1

Abstract

Chapter 2

Introduction

2.1 Image Similarity

What does it mean for two images to be similar? Are two red-hued images similar, even if they are of different objects? Are an image of a cardinal and an image of a blue jay similar because they both show birds? Regardless of the semantic interpretation of the word 'similar', any model that determines whether or not two images are similar must first be able to embed them in a visual representation.

Learning visual representations for images has been important for a variety of tasks, including image classification, semantic segmentation, object detection, and even geolocating images. The methods of extracting image representations have changed greatly over the years, moving from manually defined features like histograms of oriented gradients (HOGs) and scale-invariant feature transforms (SIFT)(30)(10), to the current state of the art of extracting feature vectors from convolutional neural networks, beginning with the seminal work of Krizhevsky et al. which achieved classification results for the 2012 Imagenet ILSVRC image classification contest far surpassing previous state-of-the-art benchmarks(24). Yet despite the change in techniques, the basic requirements for model training have remained the same. First, large quantities of reliably labeled data must be gathered, and then a model, usually a CNN, can trained on this data. With the popularization of CNNs, often Mechanical Turkers have employed to create massive and highly pure labeled datasets.

The popularization of large scale image datasets such as ImageNet, which contains 14,197,122 images belonging to 1000 classes(11), the MIT Places dataset which contains 7 million images for scene classification(50), the SUN scene classification database(46), and the Microsoft COCO dataset of 2.5 million images for common objects in context(29) have allowed researchers to build models highly adept at basic image classification tasks(36). In a review of deep learning models trained on these massive datasets with many basic image classes (ImageNet contains classes for many animals and plants and items such as 'tennis ball', 'fountain pen', and 'tricycle') Russakovsky et al. concluded that deep learning techniques were able to transfer learning from these dataset classes to other generic classes, such as

distinguishing dogs from airplanes.(36) However, more fine-grained image classification, such as between species of flowers, or of dogs of different ages, required learning different image embeddings.

After the deep learning revolution, a major direction of image understanding research has been to develop more fine-grained datasets for the training of models for highly specific tasks. A quick search for image datasets published in 2016 returns ones for irises, ultrasounds, weather property, tumors, light fields, and food calories.(6)(9)(8)(38)(32)(33). Yet, although we have been able to train more and more specific models for finer and finer grained image classification, this research still relies on the gathering of accurately labeled data. It is infeasible to gather large quantities of data for every possible image understanding task.

2.1.1 Fine-Grained Image Similarity

Fine-grained similarity also pushes against a fundamental tenant of machine vision: the desire for algorithms to capture various types of invariance–rotation, translation, scaling, illumination, color, etc. The importance of invariance to the field of machine vision is enshrined in the seminal technique of using scale-invariant feature transformations. Krishevsky et al. note also that their scheme "approximately captures an important property of natural images, namely, that object identity is invariant to changes in the intensity and color fo the illumination."(24) Standard augmentation techniques in deep learning include flipping images horizontally and vertically, rotation images, extracting random crops and zoom, and jittering color and intensity channels with random noise, all with the purpose of forcing models to learn these types of invariants. These techniques make sense for many types of contemporary vision-related tasks such as object recognition, bounding box detecting, and scene classification, since a red car and blue car are still both cars, and the presence of either might inform an algorithm that a scene depicts a highway. Yet, for fine-grained image similarity purposes where we are attempting to determine which two images are most similar out of a red car, blue car, and red car, we do not want our model to learn color invariance. Similar considerations for rotational, translational, and other types of invariance must also be made.

Thus, the challenge in the task attempted is evident: we seek to 1) learn to detect image structures useful for embedding them in a similarity space, 2) learn to do so using a weakly labeled dataset, and 3) learn to distinguish images on a very fine-grained level, which may require learning an embedding that ignores invariants that might make task 1) easier.

2.2 Weakly Supervised Training

TODO: moving toward the Holy Grail of unsupervised training

Some more recent approaches have looked into augmenting highly supervised training with weakly supervised web data, thus greatly increasing the amount of data available for these highly specific image understanding tasks. Xu et al.(47) use existing datasets to learn feature

representations and part-based object classifiers. They then extract accurate part labels from fuzzy web image data. Kraus et al take this line of work even further and use generic image recognition techniques on noisy web data and exceed state-of-the-art classification accuracies on the CUB-200-2011 dataset, without using any manually labeled data.(23)

2.3 Motivations

This paper follows this recent work in utilizing the large quantity of image data available online through search engines and image hosting sites like Google and Flickr. Rather than use web queries to form fuzzy classes for image classification like Kraus et al, we seek to use geo-tagged images uploaded to Flickr to learn an image embedding useful for image similarity tasks. We do not train on any image pairs manually labeled as similar but rather rely on the physical geographic distance between two images to inform the training of our model. We explore how well geographic distance can serve as a stand-in for manually labeled similarity data, with a particular focus on exploring heuristics for sampling pairs of similar images for maximal learning efficiency.

We choose to use deep learning representations of images trained through a Siamese network rather than use manually crafted features such as Gabor filters, scale-invariant feature transforms (SIFT), or histograms of oriented gradients (HOG), believing that allowing a model to learn features on its own will be more robust. In a departure from the research of Xu et al and Kraus et al, we also work with a classless representation of our data. We do not assign an image to a fuzzy class, such as belonging to the category 'bridge', but associate it only with its latitude and longitude data. In a sense, this means we treat each image as belonging to a unique geolocation class of size 1.

Using our developed heuristics for sampling pairs of similar images for comparison with dissimilar images, we train a deep Siamese network to learn a low dimensional feature representation, with an objective of learning that pairs of images close in physical distance should be closer in our image embedding space.

2.4 Contributions

The contributions of this paper are [to be completed]

Chapter 3

Background

3.1 A History of Machine Vision

3.1.1 Classical Techniques

3.1.2 The Deep Learning Revolution

The field of image similarity relies on learning a useful model for embedding images into a feature space. Recent years have seen some groundbreaking advancements in the application of machine learning for vision tasks, especially in the field of deep learning. Convolutional neural networks(26) are capable of learning low, medium, and high level features, using nonlinear transformations to abstract high level features into more and more basic ones. Krizhevsky et al's momentous performance in the 2012 ILSVRC ImageNet classification competition provided the first demonstration of the effeciveness of deep learning.(24) Recent research has shown that deeper models can perform even better at a variety of image tasks.(42) Much work on different activation functions has allowed CNNs to become much more sparse, and combined with work exploring deep network depths(39)(42) as well as with work allowing models to regulate their own depth(18), deep CNNs have proven extremely effective at learning a variety of useful image representations. Athiwaratkun and Kang show that just the image representation extracted by deep CNNs can be combined with simpler classifiers such as SVMs and random forests to achieve high accuracies for clustering tasks.(2) In recent years, some deep learning models have even achieved classification accuracies surpassing even human performance. In 2015, He et al. achieved a 4.94% top-5 test error on the ImageNet 2012 dataset, surpassing the human error performance of 5.1%(18).

Importantly, deep learning methods do not require the manual crafting of features based on domain-level knowledge. Instead, deep learning models learn to abstract patterns from data automatically. This approach stands in contrast to the majority of work done in the 1990s and 2000s, which made extensive use of manually defined image feature extraction techniques, such as Gabor filters, scale-invariant feature transforms (SIFT), and histograms

of oriented gradients (HOG).(19)(30)(10) In the latter half of the 2000s, hierarchical feature representations such as spatial pyramids, which transform images into segmentations, each of which is locally orderless, proved effective in a variety of image tasks as well(48)(13)(25). In the field of content-based image retrieval, spatial envelopes and transformed histograms where used to attempt to capture global scene properties(31)(45). Features extracted using these methods were then used with rigid distance functions such as Euclidean or cosine similarity distances to determine an overall image similarity. Much work has been done on designing better similarity measures for these low-level features. Notably, Jegou et al.(20) adapt the Fisher kernel for use in aggregating local image descriptors into a reduced dimension vector while preserving the bulk of relative distance information.

3.1.3 Deep Convolutional Neural Networks

Convolutional neural networks have become the de-facto standard in image tasks, as stacked convolutional layers are well suited for learning image descriptors.(21)(24)(42) Besides convolutional layers, CNNs typically consist of pooling layers, activation layers, fully connected layers, and a loss layer. A convolutional layer consists of a set of kernels K, each of which typically has width and height dimensions smaller than the dimensions of an inputted image, but with a depth matching the depth of the input. During the forward pass of network training, each filter is convolved with a sliding patch across the input's width and height, producing a feature map associated with that kernel. Each feature map codes the activation of the kernel along with the spatial location of that activation. Because the dimensions of K are smaller than the input, convolutional layers only have local connectivity.

Pooling layers, usually in the form of max-pooling, down sample the feature maps produced by the convolutional layers. Max pooling will output the maximum value in each part of a segmentation of a feature map. Pooling layers drastically reduce the computation required to train a network. Activation layers are used to control which nodes in a layer send output to the next layer. The standard activation currently used is the rectified linear activation unit (ReLU), which takes the form

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{3.1.1}$$

Various forms of ReLU have been proposed, notably the parametric ReLU (pReLU), which adds a learnable parameter, α, to control a slope for the negative activation domain and which was used by He et al. to achieve better than human performance for ImageNet classification.(18)

$$f(x) = \begin{cases} x & x > 0 \\ \alpha x & x \leq 0 \end{cases} \tag{3.1.2}$$

Since convolutional layers only produce feature maps on local scales, fully connected layers at the end of a CNN allow for high level features to be learned. These layers have full connections to all activated neurons from previous layers, which allow for global mixing of activated feature maps. To complete our discussion of the CNN, the loss layer specifies how

a network should penalize incorrect predictions. Typically sigmoid cross-entropy loss is used. Regularization is also used, typically in the form of L1 or L2 weight decay. Dropout layers, which deactivate a random subset of a layer's neurons in each iteration of training, have also proved highly effective for neural network regularization.(40)

The term deep CNN refers to a CNN that has many layers. This definition is highly variable: the popular VGG16 model has 16 layers(39) and popular versions of ResNet contain 34, 50, 101, and 152 layers(18).

We can more precisely define a CNN as a function f that takes parameters θ and an input image I to produce an image embedding x and a loss L.

3.1.4 Deep Embeddings

3.1.5 Image Invariants

3.2 Deep Learning for Image Similarity

3.2.1 Siamese Neural Networks

As noted by Wang et al(43), the network structures that are effective at classifying images into object classes are not necessarily well-designed for detecting image similarity, especially when image similar is defined not just as whether two objects are in the same class, but when our desired similar metric must be fine enough to rank similarities within classes. For example, a red book should be judged more similar to a maroon book and should a light green one. The Siamese architecture, first proposed by Bromley et al. in 1994(7) for the purposes of verifying signatures, is well suited for this task. Siamese networks have since been used in a variety of similarity tasks, such as ground-to-aerial geolocalization(28), matching visual similarity for product design(3), comparing image patches(49), and one-shot image classification(22).

A Siamese net is a formulation of two copies of a CNN that share parameters and hyperparameters as well as a loss layer and thus weight updates. If we represent a Siamese net as f, then f takes θ, two images I_1, I_2 as well as an indicator variable p to indicate if these images form a positive (similar) or negative (dissimilar) pair, produces two embeddings x_1, x_2 and one loss L. We wish to find a locally optimal θ such that for a triplet of embeddings x_1, x_2, x_3 produced by f, if $|x_1 - x_2|_2^2 < |x_1 - x_3|_2^2$, then we expect I_1 and I_2 to be much more semantically similar than I_1 and I_3.

As discussed by Bell and Bala(3), Siamese networks can be tweaked in various ways to produce both an embedding and a classification for each pair of images. The output from the CNN layer can regularized either for the embedding predictions or the class predictions or both. If there are multiple output from the CNN layer, multiple loss layers can be used as well.

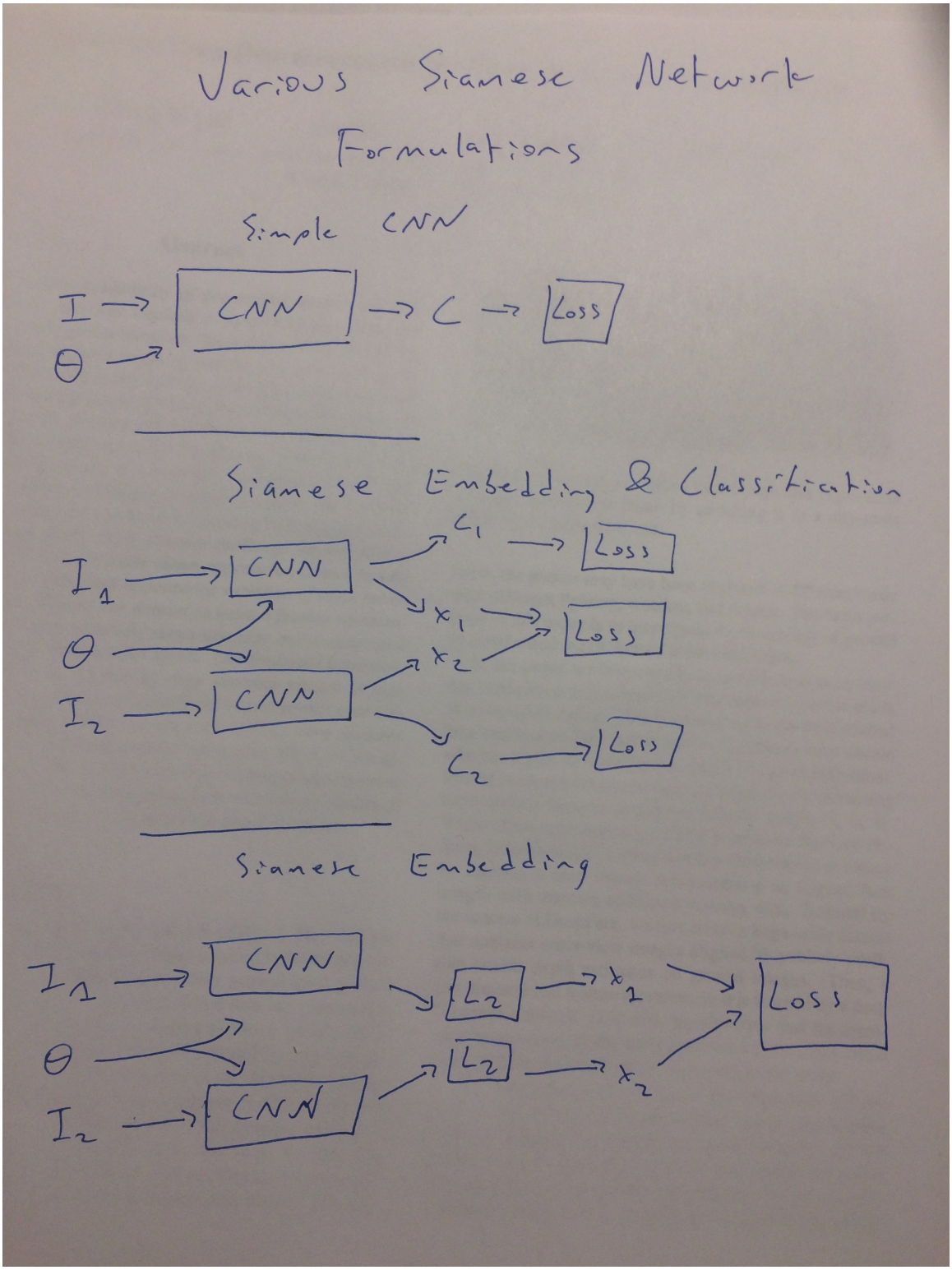


Figure 3.1: Some configurations from Bell and Bala

Distance Metric for Siamese Network

The similarity of two images, $S(A, B)$, can be defined as the Euclidean distance of their feature embedded vectors, $f(A)$ and $f(B)$:

$$S(A, B) = ||f(A) - f(B)||_2^2 \tag{3.2.1}$$

A cosine similarity $\frac{f(A) \cdot f(B)}{||f(A)||||f(B)||}$ can also be used but is less common. Here $f(\cdot)$ might be a feature embedding such as the weight representation of the final convolutional block in a convolutional neural network pretrained on ImageNet, or a CNN trained from scratch.

A contrastive loss function proposed by Hadsell et al and followed by Lin et al. for pairs of images is as follows:

$$L(l, p_1, p_2) = \frac{1}{2} l S(p_1, p_2) + \frac{1}{2}(1 - l)\max(0, (g - S(p_1, p_2))) \tag{3.2.2}$$

where l is an indicator variable equal to 1 if the pair is similar and 0 if not, and g is a regulator for the margin between unmatched pairs.(15)(28) This loss function assigns a low loss to similar pairs and a high loss for dissimilar pairs.

We use pairwise comparisons of images, grouping sets of three images, p_i, p_i^+, and p_i^-, into two pairs, (p_i, p_i^+) and (p_i, p_i^-). We can extend this pair loss function into a hinge loss for triplets as follows(43):

$$l(p_i, p_i^+, p_i^-) = \max\{0, g + ||f(p_i) - f(p_i^+)||_2^2 - ||f(p_i) - f(p_i^-)||_2^2\} \tag{3.2.3}$$

We can add L2 regularization so that our objective function is

$$\frac{\lambda}{2}||\mathbf{W}||_2^2 \max\{0, g + ||f(p_i) - f(p_i^+)||_2^2 - ||f(p_i) - f(p_i^-)||_2^2\} \tag{3.2.4}$$

where λ is a regularization parameter and $\mathbf{W}$ is our weight parameter matrix.

3.3 Weakly Supervised Learning

The problems with relying on supervised learning are many. It is difficult and time consuming to create large datasets, and even when these are created, such as for ImageNet, they are often specific to a certain task and do not necessarily extend to novel concepts. As Russakowsky et al. discussed(36), models pretrained on general datsets like Imagenet were only good at segmenting images into basic classes. Russakowsky et al. also demonstrated that these models were better tuned for classification of natural classes such as animals than they were for man-made objects, suggesting that even an expansive 12 million image-large dataset like ImageNet still had flaws for general classification training. In recent years, a proliferation of intra-class datasets, such as for hundreds of species of flowers or birds, have allowed deep

learning techniques to tackle everything from differentiating species of flowers(1), leaves(34), and birds(4), but the limitations of a class-based formulation of image similarity remain apparent. More and more specific class formulas will need to be created, and this is higly reliant on how explicit human annotations are. For example, labeling an image as a dog, which might be reasonable for an animal differentiation task, will create problems if this data is ever used for a more fine-grained image similarity task, such as one that requires differentiation of labradors from golden retrievers.

Much work on using weakly supervised learning has focused on using web images as a source of easily and quickly obtainable weakly labeled dataset.(5)(12)(27)(37). While on the whole successful, researchers note that web created classes suffer from polysemy and noise problems. A search for penguin images will not necessarily return only penguins because of the way images and surrounding text are indexed, and a search for screens might return both door screens and computer screen. In the field of object segmentations, Rubinstein et al achieved better than state of the art benchmark results training on a web gathered corpus, yet also noted difficulty with certain classes because of the low quality of some images.(35)

Krause et al. use fuzzy web-based data to remarkable effectiveness, successfully training classification for over 14,000 image classes, achieving close to human accuracy on the CUB Caltech bird species dataset, and beating state of the art results for CUB and the Stanford dog dataset(23). They quantify the fuzziness (incorrectly labeled images) in their images classes to be an average of 16%, demonstrating that a certain amount of fuzziness is still able to be overcome in training.

3.3.1 Fuzzy Data

Chapter 4

Network Architecture and Training

4.1 Model Design

We build off of the general Siamese network formulation advanced by Hadsell et al. for facial recognition and used by Lin et al. for ground-to-air geolocalization.(15)(28) We extend their formulations by recognizing that fine-grained similarity tasks require more than one base CNN.

Our final model takes an input image p and produces an image embedding E. p is preprocessed by rescaling to (224, 224), leaving the color channel in place and unmodified. p is fed to two deep residual CNNs, I and V, which have been trained to extract the presence of image invariants and variants, respectively. These outputs are concatenated and used as input to a shallow CNN, B, which has been trained to blend the presence of invariants and variants. The output of B represents the final image embedding.

We train I, V, and B through a Siamese configuration. We feed our model pairs of training images, alternating between similar and dissimilar pairs (p, p_+) and (p, p_-). Images are preprocessed by rescaling both the width and height to 224 pixels, leaving the color channel in place and unmodified. Data augmentation, when used, is applied directly after preprocessing. The base image, p, is fed to half of our Siamese network, which we denote as C_p, and the query image, which will be either p_+ or p_-, is fed to the other half of the Siamese net, C_q. C_p and C_q share weight updates.

We use the Flickr dataset to train V and I and the Google dataset to train B. We illustrate these two processes in Figures 4.1 and 4.1. For training V and I, we use three different methods. In the first, we do not train I, but instead use a network pretrained on ImageNet, which has been shown to learn image invariants such as translational, rotational, reflectional, illumination, and crop invariants. We freeze the weights of this network. We train V by running pairs of images through our Siamese network, feeding the images to I and V and concatenating their outputs. For this training, we do not blend their outputs with B, but their concatenation is taken as the output of C_x, and we find the Euclidean distances of these

two output vectors, which represent our image embeddings. Finally, this distance is passed to our contrastive loss layer, which takes our fuzzy binary label for similarity/dissimilarity, and signals the model to update weights so that input pairs labeled as similar produce image embeddings with small Euclidean distances, and input pairs labeled as dissimilar produce image embeddings with larger Euclidean distances. This method is the one illustrated in Figure 4.1.

In the second method, we feed input images only to V and not to I. The output of V is taken as the output of C_x, and the rest of the process is the same as in the first method. In the last method, we feed input images to both V and I and take their concatenated outputs as the output of C_x. In this case, we do not freeze the weights of I, but use the contrastive loss layer to update the weights of both V and I. Otherwise, the third method proceeds following the first.

Regardless of how we have trained weights for V and I, the training of B remains the same. B takes as input the concatenated outputs of V and I and the output of B is taken as the output of C_x. The contrastive loss layer is used to update the weights of B, but the weights of V and I are frozen. B is trained using the Google dataset, which is strongly labeled. Since the Google dataset is quite small, we use 10-fold cross-validation rather than leaving our part of the data to use as validation and test sets.

We refer to our entire configuration as a double Siamese network, since each half of the Siamese network has two base CNNs, I and V.

4.1.1 Proprocessing

We downsample all images to size (224,224), a common size proven useful for feature representation for the ImageNet classification task by the ResNet architectures.(16) We do not grayscale our images or perform any other color manipulation because we expect the time of day a photo is taken to have a significant effect on its representation in our image embedding. However, many photos uploaded to Flickr are in grayscale, so we remove these using an entropy metric. We first apply a Laplacian filter to each image and then find the Shannon entropy, pruning images with an entropy below a threshold. For some experiments, we apply mean pixel subtraction to center our data by subtracting the color channel means for our entire dataset from each training image. This is to bound each input image's pixel values to roughly the same range so that when we share gradients in the backpropagation phase of our CNN training, we don't have weights of varying magnitudes. We find that our use of batch normalization layers is able to share gradients effectively, negating the usefulness of mean pixel subtraction, so we do not use this technique for all experiments.

4.1.2 Data Augmentation

Data augmentation is applied after preprocessing for some of our experiments, mainly to improve model generalizability. Images are randomly flipped, both horizontally and

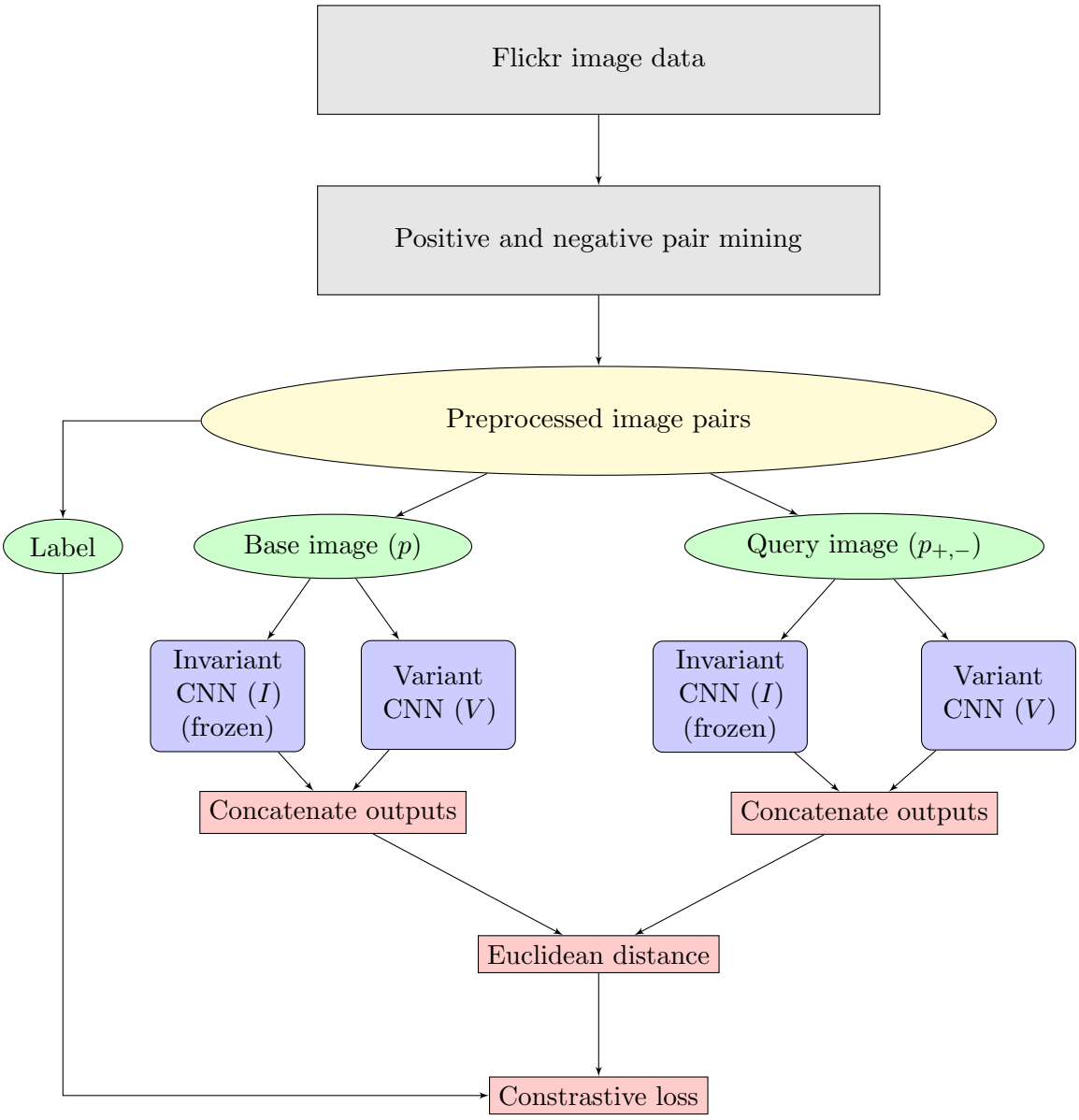


Figure 4.1: Pipeline for training the variant network, V

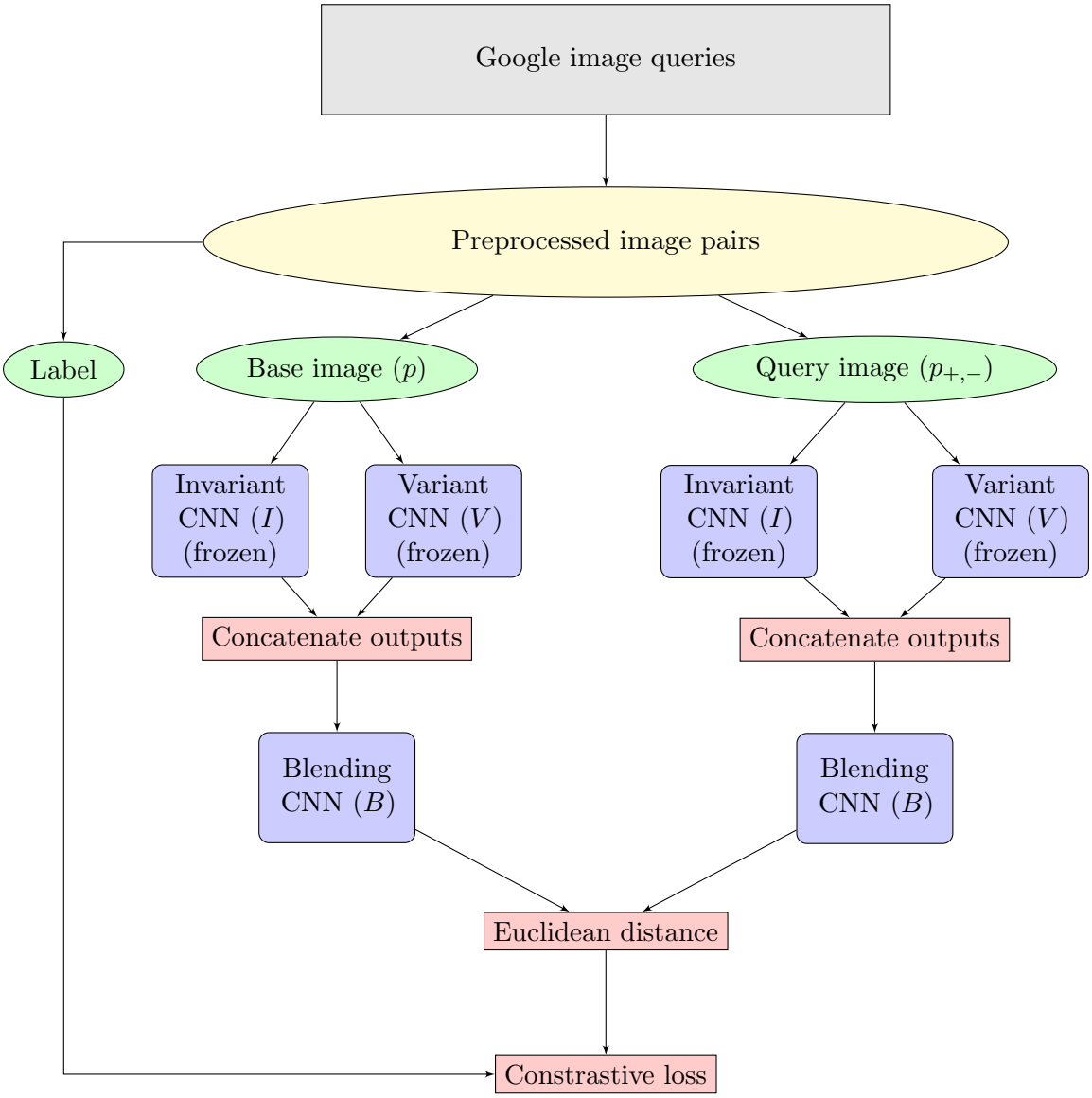


Figure 4.2: Pipeline for training the blending network, B

vertically, rotated up to 20 degrees, and cropped on zooms of up to 5%. We do not apply any augmentation to color channels.

4.1.3 Detecting Variants and Invariants

We do very little exploration of different network configurations, opting to lightly modify a 50 layer ResNet architecture. The ResNet family, which advanced the idea of using residual layers, represented a powerful step forward in deep architectures in 2015, winning first place in the ImageNet classification, ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation competitions.(16) While ResNet is no longer considered a cutting edge architecture, as it has been surpassed by the third generation of Inception architectures as well as by ResNet hybrids like Inception-ResNet, we choose to use a ResNet model as our base model because of the previous work done in demonstrating the effectiveness of various ResNet depths.(16)(41) Popular versions of ResNet include ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152, and ResNet-200. In general, increasing depths produce increased accuracy on standard vision benchmarks, while taking greater penalties in the form of increased memory footprints and slower training speeds for forward and backward passes during backpropagation. The advantage of the ResNet family lies in our ability to easily scale the depth of our network, should we decide our task requires greater visual discriminative power or faster training speeds. More powerful Inception architectures are tuned with a different preprocessing pipeline, so swapping out deeper architectures would not be as easy. The performance gap between ResNet and cutting edge architectures is small enough, as shown in Table 4.1.3, that we choose to stick with the ResNet family for the increased flexibility. Ultimately, we found that within the ResNet family, ResNet-50 had a suitable balance between model depth and training speed.

Network	Layers	Top-1 error	Top-5 error	Speed (ms)
Inception V3	48	21.2	5.6	NA
Inception-ResNet-2	NA	19.6	4.7	NA
ResNet-18	18	30.43	10.76	31.54
ResNet-34	34	26.73	8.74	51.59
ResNet-50	50	24.01	7.02	103.58
ResNet-101	101	22.44	6.21	156.44
ResNet-152	152	22.16	6.16	217.91
ResNet-200	200	21.66	5.79	296.51

Our implementation modifies ResNet-50 by changing its standard weight initializations, changing the activation functions, and by specifying a 1024-dimensional output for our image embedding. The choice of 1024 for our image embedding dimension is somewhat arbitrary. We desire to keep our dimension around the same magnitude as the outputs for ImageNet and Places365–1000 and 365 respectively.

He Normal weight initialization

We use the He Normal weight initialization, which assumes a truncated normal distribution with zero mean and a standard deviation equivalent to $\sqrt{2/f}$, where f is the number of input connections in a layer's weight tensor. Usually, CNNs are initialized with weights drawn from Gaussian distributions. Glorot and Bengio argue that we would like the variance of a layer l_{n+1} to be equivalent to the variance of our output of the previous layer l_n so that our weights neither shrink to 0 nor explode as an input is passed through a deep network.(14) Their Xavier-Glorot initialization preserves the magnitude of weights for both the forward and backwards passes. This necessitates choosing a weight initialization, W, such that $Var(W_i) = \dfrac{2}{n_{in} + n_{out}}$, where n_{in} and n_{out} refer to the number of input and output connections for an i-indexed neuron. He et al. find that this initialization leads to a stall in training for very deep architectures with more than 30 layers and propose the He Normal initialization which requires instead that $Var(W_i) = \dfrac{2}{n_{in}}$.(17)

Parametric ReLU activation

Activation layers are used to control which nodes in a layer send output to the next layer. The standard activation currently used in deep learning is the rectified linear activation unit (ReLU), which takes the form

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{4.1.1}$$

Various forms of ReLU have been proposed, notably the parametric ReLU (pReLU), which adds a learnable parameter, α, to control a slope for the negative activation domain and which was used by He et al. to achieve better than human performance for ImageNet classification.(18)

$$f(x) = \begin{cases} x & x > 0 \\ \alpha x & x \leq 0 \end{cases} \tag{4.1.2}$$

Because we are training on a fairly large and very noisy dataset, we seek to mitigate the possibility of 'dying' ReLU activations by using parametric ReLU activations. ReLU units can 'die' when a large gradient causes the weights to update in such a way that the unit never again activates for the rest of the dataset, causing it to output 0. Since the unit never contributes to the model prediction, it is never updated, and is therefore a dead end in the model. Experimentally, up to 40% of network ReLU units can die. We expect it will be important to mitigate this since our dataset spans a wide variety of images, from all white to all black, and therefore we are likely to have a high chance of killing more ReLU units than normal.

In addition, we clip the norms of our weight gradients at 1.

4.1.4 Blending Variant and Invariant Outputs

We use a shallow CNN to blend our variant and invariant outputs. We experiment with CNNs with one and two fully connected layers and also experiment with the sizes of these layers. Because of the limited size of our dataset, we experiment with some ad-hoc manual architectures as well by including a residual layer for either the variant output or invariant output, or for both. When using two fully connected ayers, we use dropout improved generalizability. Again, we use a He Normal weight initialization and pReLU for our activation function.

4.1.5 Loss Function

Our loss function follows the contrastive loss proposed by Hadsell et al.(15), which assigns a high loss to pairs whose embeddings are far apart and a low loss to pairs whose embeddings are close together. Wang et al.(43) had extended this loss function to take triplets of images, but in order to use this loss, we would have to prepare triplets of images, which proved computationally intractible for certain experiments we wanted to run, which we will discuss in Chapter 5.

$$L(l, p_1, p_2) = \frac{1}{2} l S(p_1, p_2) + \frac{1}{2}(1 - l)\max(0, (g - S(p_1, p_2))) \tag{4.1.3}$$

$$l(p_i, p_i^+, p_i^-) = \max\{0, g + ||f(p_i) - f(p_i^+)||_2^2 - ||f(p_i) - f(p_i^-)||_2^2\} \tag{4.1.4}$$

4.2 Training and Validation

4.2.1 Data Sampling

Chapter 5

Data

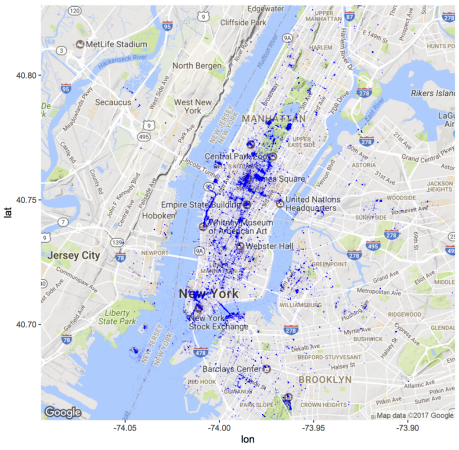
5.1 Flickr Data

Our available data consists of all geo-tagged images uploaded to Flickr between 00:00:00 (GMT) January 1, 2006 to 00:00:00 January 1, 2017 with latitude and longitude inside the [lower left, upper right] bounding box [(-74.052544, 40.525070), (-73.740685, 40.889249)]. This bounding box roughly corresponds to the city limits of New York, New York. There are over 6 million images in the dataset, with roughly 500,000 each for the years from 2009-2016. In addition to the latitude and longitude, each image was downloaded with an associated timestamp, Flickr user identifier, title, and description (user uploaded caption). The vast majority of photos are from Manhattan, and distinct clusters can be seen around typical tourist attractions such as the World Trade Center, the Brooklyn Bridge, the Metropolitan Museum of Art, the Rockefeller Center, and up and down Broadway Avenue. To give an idea of the image densities shown, there are 6745 images with latitude beginning with 40.779 and longitude beginning with -73.963, which corresponds to 100 square meters around Metropolitan Museum of Art from 2014 only.

It is important to note that our latitude and longitude information is not precise. Latitude and longitude coordinates are given with six decimal places, and at the latitude of New York (roughly 40°), a 0.000001 change in latitude represents roughly one tenth of a meter, and a 0.000001 change in longitude represents roughly one thirtieth of a meter. However, though we only use images Flickr has denoted as having their highest accuracy level, Flickr's API describes this as "street-level" accuracy. This would imply an error bar on our image locations of about 10 meters, and therefore we cannot claim with certainty that our distance calculations follow a precise conversion from their latitude and longitude deltas. For simplicity, we take our error to be 10 meters, noting also that for the purposes of creating a weakly labeled dataset, we are not too concerned with our inability to firmly quantify this statistic. This means two images with a Euclidean distance, d, between their coordinates can be, at worst, up to $d + 20$ meters away.

With 6 million images, there are roughly 1.8×10^{13} possible image pairs, which is infeasible

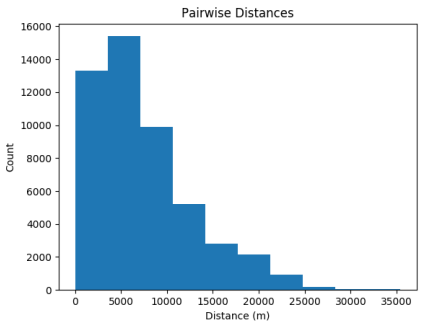
(a) All images



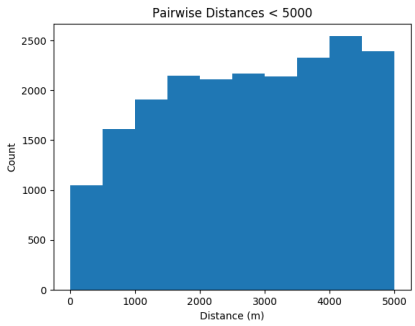
(b) Closer view of highest image density

to train a CNN on, so we develop a heuristic for sampling positive and negative image pairs for efficient training, which we explain in Section 5.1.1.

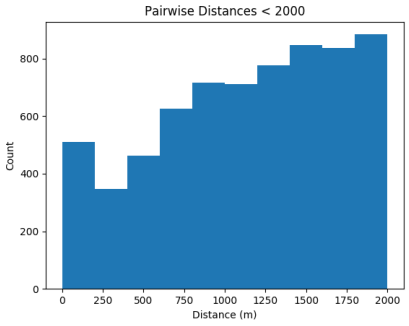
We examine the pairwise distances of our images. Roughly 0.6% of our image pairs have an image distance of one meter or less.



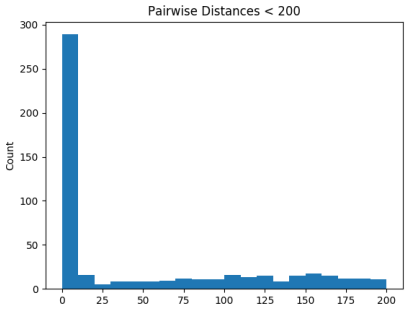
(a) All sampled pairwise distances



(b) Within 5000 meters



(c) Within 2000 meters



(d) Within 200 meters

A simple visualization of randomly selected images at two clusters, the Metropolitan Museum of Art and the Brooklyn Bridge, reveals there is relative intracluster similarity and intercluster

dissimilarity. These two sets of example clusters are extremely far apart in high-level semantic space and are perhaps not representative of the dataset as a whole, which consists of many generic street view photos. But they do demonstrate that there is significant high-level variation in image features as a function of geographic location, particularly when switching from indoor to outdoor settings.

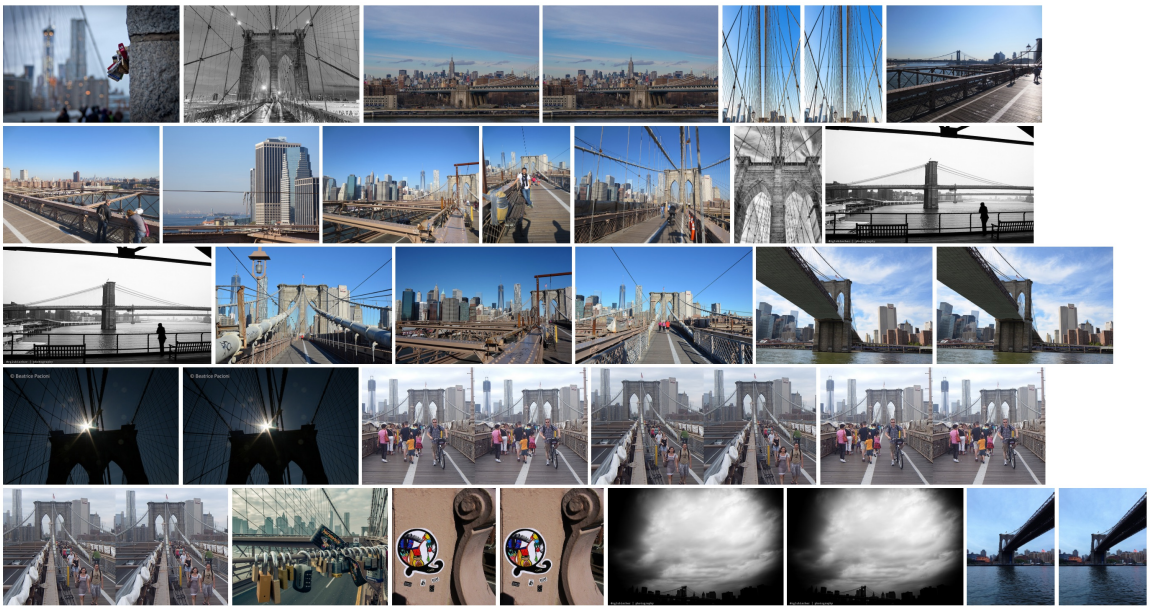


Figure 5.1: Photos from the Brooklyn Bridge (40.706 -73.996)

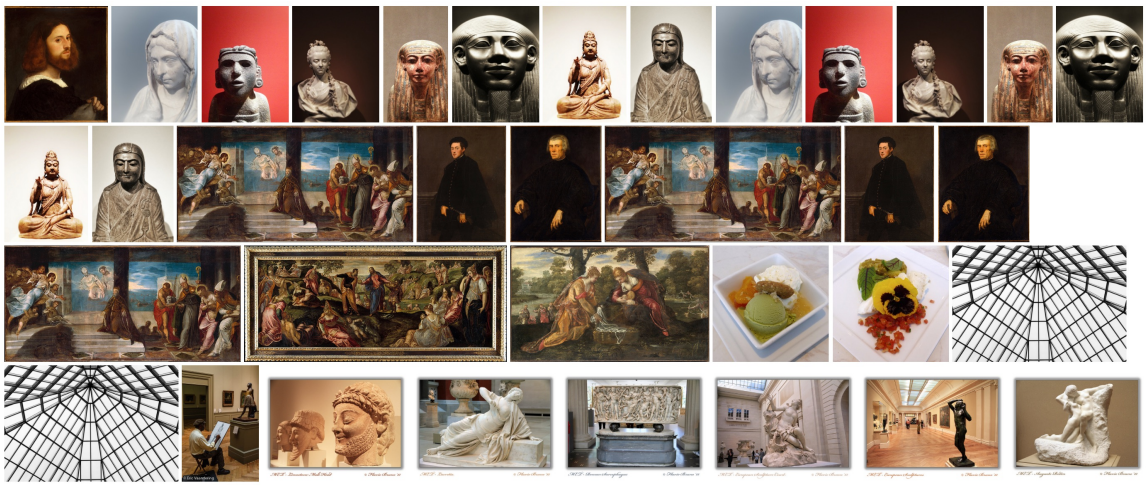


Figure 5.2: Photos from the Metropolitan Museum of Art (40.779 -73.963)

We notice that there are a non-trivial amount of duplicate photos. Some can be seen in the selection of images from the Brooklyn Bridge in Figure 5.1. We use aggressive duplication removal when selecting positive image pairs for training.

Though the existence of semantic clusters provides the basis for our model's learnability, we never explicitly create these clusters or segment our images in any way. In training a CNN to geolocate Flickr images on a global scale, Weyand et al. look at the geographic density

of their photos and divide the earth into variable size latitude longitude bounding boxes which they use as image classes.(44) The existence of classes allows Weyand et al to frame geolocalization as a standard classification problem to which a CNN is easily applied. We do not explicitly apply this sort of segmentation and we do not seek to predict a class but rather to find an image embedding.

5.1.1 Pair Sampling

Since our full dataset is many magnitudes too large to fully use in model training as there are trillions of possible image pairs that can be created, we design sampling heuristics intended to minimize the number of pairs used while maximizing learning potential. We do this by using various distance, temporal, and other meta-data heuristics. We denote a set of pairs as $\mathcal{P}$. Unless otherwise noted, $\mathcal{P}$ will be understood to consist of similar pairs of images (p, p^+) and dissimilar pairs (p, p^-) in equal proportion. When referring to pairs or images, similar and positive should be considered interchangeable, as should dissimilar and negative.

Distance Heuristics

To allow for computationally tractible sampling of image pairs by distance metrics, we load sets of images into a custom implementation of a KD-Tree where images are indexed by their latitude and longitude. We noted in Section 5.1, that, because of imprecision in our image geolocation, two images with a Euclidean distance d between their coordinates can be, at worst, up to $d + 20$ meters away, and at best, $\texttt{max}(0, d - 20)$ meters away.

We form $\mathcal{P}_{21,2000}$ by doing a range query for all pairs p, p^+ within 1 meter of each other. Experimentally, the pairs that are returned have the same latitude, longitude coordinates. Taking into account our maximum geolocation error, this means p, p^+ are within 11 meters of each other. From these positive pairs, we then randomly sample for images p^- that are farther than 2020 meters from p. The distance of 2020 meters is chosen to be longer than the maximum diameter of any eyeballed cluster in Figure 5.1. The longest such cluster diameter is the Brooklyn Bridge at a span of 1825 meters. Thus, $\mathcal{P}_{21,2000}$ enforces a maximum distance for p, p^+ of 11 meters and a minimum distance for p, p^- of 2000 meters.

We form a second set $\mathcal{P}_{30,2000}$ doing a range query for all pairs p, p^+ within 10 meters of each other. Taking into account our geolocation error, this means p and p^+ can actually be up to 30 meters apart. Negative pair sampling proceeds in the same manner as for $\mathcal{P}_{30,2000}$.

Attempts to create a dataset with negative samples p^- lying at minimum a and at maximum b from p by performing a ring query proved to be computationally intractible for the purposes of this project. The KD-Tree algorithm supports a highly scalable ball query for points lying within a certain distance from each other even for datasets of our size. Our attempted ring query can be easily decomposed into the subtraction of the set of results of a ball query with radius a from the set of results of a ball query with radius b, but because of the associated meta-data with our images (including descriptions and other user information), hashing pair

results to a set was actually infeasible. Otherwise we would have explored results for sets like $\mathcal{P}_{10,[30-100]}$, where negative samples lie farther than 30 meters but within 100 meters from the base image.

Temporal Heuristics

We split our original Flickr dataset into subsets divided by year and month. We apply the above distance heuristics to these time divisions, thus creating sets like $\mathcal{P}_{10,2000,June2013})$ and $\mathcal{P}_{10,2000,2013-2015})$, which refer to the distance set described intersected with the set of all possible image pairs from June 2013 and from 2013, 2014, and 2015 respectively. We expect datasets limited to one month to have pairs that are more similar in general, for both the positive pairs and negative pairs, whereas datasets which span years will have pairs that are less similar in general. This intuition is seen by considering that two photos taken in June of 2013 will have similar weather patterns, similar clothing worn by people, and similar ranges of daylight hours, among other factors. Two photos sampled from any time between 2013 and 2015 will be, on average, less likely to share these similarities.

Filtering by User

As can be expected, our distance heuristic is extremely fuzzy. A selfie taken at Times Square will look very different from a family tourist photo taken at the same place, and both will look very different from a photo of a crowd waiting for the New Year. And it is highly possible that two photos taken miles apart, for example at the George Washington Bridge and the Brooklyn Bridge, or the Metropolitan Museum of Art and the Museum of Modern Art, will look fairly similar. We create another dataset designed to have less fuzziness by requiring similar pairs to have been taken by the same user. We can denote this as $\mathcal{P}_{10,2000,June2013,same})$. We also create datasets where we require both positive and negative examples to be within a certain bounding ball but differentiate positive pairs as having the same user and negative pairs as having different users. These would be denoted as $\mathcal{P}_{10,2013,same}$ if the bounding ball is 10 meters and the images are from 2013.

Other

We notice there is a user who has uploaded thousands of photos from a time lapse of construction of the Barclays Arena in Brooklyn. These images slowly change over time as construction progresses, and they change hourly with the arrival and departure of trucks and people as well as with the lighting conditions. We construct a dataset where similar images are pairs sampled entirely from this time lapse, and dissimilar images follow a minimum distance heuristic. We refer to this as $\mathcal{P}_{timelapse,2000,June2013}$. TODO should I put information about how we expect this to force our model to learn invariance or should I leave that for discussion?

Summary

In total, we create TODO(number) datasets designed to have various degrees of label fuzziness and to require our model to focus on different features. We summarize these in Table 5.2. Because we generate our datasets by sampling, it is difficult to force them all to have the same size. We decide we would rather have possibly large variances in dataset size than throw away sampled data by truncating them all to be of size $\min |\mathcal{P}|$. Again, the amount of data we have is truly staggering, so we mostly focus on generating sets from the years 2013, 2014, and 2015, though we did also generate data from other years. For brevity, summarize only a subset of the datasets that we ran experiments on, since many of the results will be redundant.

5.2 Middlebury Stereo Data

While we can augment images to create datasets that include color, rotational, translational, and magnitude invariants, especially with the discovery of timelapse data in the Flickr set, it is more difficult to artificially construct a dataset that includes stereoscopic invariance. In order to study the effects of this type of invariance on image similarity, we use a dataset prepared by Scharstein et al. in a studio of stereo algorithms.(**?**)

The dataset is very small, containing data for just 33 objects such as a motorbike, plants, and umbrella. There are four pictures per object–default left and right stereo images, a right image under different exposure, and a right image under different lighting conditions.

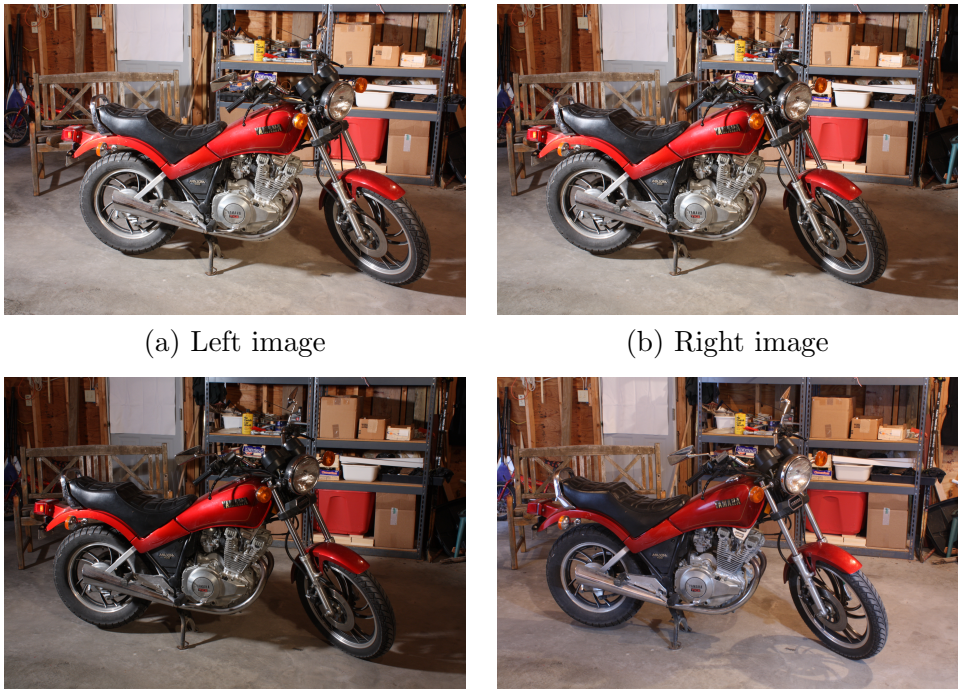
We display examples in Figure 5.2.

We create several datasets with pairs from the Middlebury data as positive pairs, and negative pairs formed from sampling our Flickr set. Positive pairs are sampled either entirely from the set of left, right stereo images associated with each object, or they are sampled by drawing pairs from the four left, right, exposure, and lighting images associated with each object. For negative pairs, we sample images from Flickr that are within 10 meters of each other and either taken by the same or different users.

We take the left and right stereo images as examples of positive pairs, and merge this with random images from our Flickr set for negative pairs. This sampling creates a dataset $\mathcal{P}_{stereoLR}$. We also create $\mathcal{P}_{stereoE}$, which takes the right image and right image with different exposure to be a positive pair, and $\mathcal{P}_{stereoL}$, which takes the right image and right image with different lighting to be a positive pair.

Dataset	Positive Distance	Negative Distance	Time Frame	Size
$\mathcal{P}_{1,2000,2013-2015}$	$<$ 1m	$>$ 2000m	Jan 2013 - Dec 2015	176000
$\mathcal{P}_{1,2000,2015}$	$<$ 1m	$>$ 2000m	Jan 2015 - Dec 2015	140800
$\mathcal{P}_{1,2013-2015,user}$	$<$ 1m, same user	$<$ 1m	Jan 2013 - Dec 2015	57600
$\mathcal{P}_{1,2013-2015,tl}$	$<$ 1m	$<$ 1m, timelapse	Jan 2013 - Dec 2015	54400
$\mathcal{P}_{10,2013-2015,user}$	$<$ 10m, same user	$<$ 10m	Jan 2013 - Dec 2015	118400
$\mathcal{P}_{10,2013-2015,tl}$	$<$ 10m	$<$ 10m, timelapse	Jan 2013 - Dec 2015	-
$\mathcal{P}_{30,2013-2015,user}$	$<$ 30m, same user within 2 hours	$<$ 30m	Jan 2013 - Dec 2015	198400
$\mathcal{P}_{Middlebury}$	Middlebury pairs	random	N/A	32000
$\mathcal{P}_{1,2000,01\_2014}$	$<$ 1m	$>$ 2000m	Jan 2014	32000
$\mathcal{P}_{1,2000,02\_2014}$	$<$ 1m	$>$ 2000m	Feb 2014	32000
$\mathcal{P}_{1,2000,03\_2014}$	$<$ 1m	$>$ 2000m	Mar 2014	32000
$\mathcal{P}_{1,2000,04\_2014}$	$<$ 1m	$>$ 2000m	Apr 2014	32000
$\mathcal{P}_{1,2000,05\_62014}$	$<$ 1m	$>$ 2000m	May 2014	32000
$\mathcal{P}_{1,2000,06\_2014}$	$<$ 1m	$>$ 2000m	Jun 2014	32000
$\mathcal{P}_{1,2000,07\_2014}$	$<$ 1m	$>$ 2000m	Jul 2014	32000
$\mathcal{P}_{1,2000,08\_2014}$	$<$ 1m	$>$ 2000m	Aug 2014	32000
$\mathcal{P}_{1,2000,09\_2014}$	$<$ 1m	$>$ 2000m	Sep 2014	32000
$\mathcal{P}_{1,2000,10\_2014}$	$<$ 1m	$>$ 2000m	Oct 2014	32000
$\mathcal{P}_{1,2000,11\_2014}$	$<$ 1m	$>$ 2000m	Nov 2014	32000
$\mathcal{P}_{1,2000,12\_2014}$	$<$ 1m	$>$ 2000m	Dec 2014	32000
$\mathcal{P}_{1,2000,01\_2015}$	$<$ 1m	$>$ 2000m	Jan 2015	32000
$\mathcal{P}_{1,2000,02\_2015}$	$<$ 1m	$>$ 2000m	Feb 2015	32000
$\mathcal{P}_{1,2000,03\_2015}$	$<$ 1m	$>$ 2000m	Mar 2015	32000
$\mathcal{P}_{1,2000,04\_2015}$	$<$ 1m	$>$ 2000m	Apr 2015	32000
$\mathcal{P}_{1,2000,05\_2015}$	$<$ 1m	$>$ 2000m	May 2015	32000
$\mathcal{P}_{1,2000,06\_2015}$	$<$ 1m	$>$ 2000m	Jun 2015	32000
$\mathcal{P}_{1,2000,07\_2015}$	$<$ 1m	$>$ 2000m	Jul 2015	32000
$\mathcal{P}_{1,2000,08\_2015}$	$<$ 1m	$>$ 2000m	Aug 2015	32000
$\mathcal{P}_{1,2000,09\_2015}$	$<$ 1m	$>$ 2000m	Sep 2015	32000
$\mathcal{P}_{1,2000,10\_2015}$	$<$ 1m	$>$ 2000m	Oct 2015	32000
$\mathcal{P}_{1,2000,11\_2015}$	$<$ 1m	$>$ 2000m	Nov 2015	32000
$\mathcal{P}_{1,2000,12\_2015}$	$<$ 1m	$>$ 2000m	Dec 2015	32000

Table 5.1: Datasets



(a) Left image (b) Right image

(c) Right image, different exposure (d) Right image, different lighting

Figure 5.3: Images for motorcycle

Dataset	Positive Pairs	Negative Pairs	Size
$\mathcal{P}_{MiddleburyLR,diff\_user}$	Left, right pairs	Different users	
$\mathcal{P}_{MiddleburyLR,same\_user}$	Left, right pairs	Same users	
$\mathcal{P}_{MiddleburyLREL,diff\_user}$	Left, right, exposure, lighting pairs	Different users	
$\mathcal{P}_{MiddleburyLREL,same\_user}$	Left, right, exposure, lighting pairs	Same users	

Table 5.2: Datasets

5.3 Google Image Data

Because we will use the Flickr dataset to generate fuzzily labeled image pairs, we also require a manually labeled dataset for testing. For this, we use a dataset published by Wang et al., which we will refer to as the Wang set.(43). The Wang set consists of 5033 image triplets. The dataset was curated by sampling triplets of images, (Q, A, B) from the top 50 search results for 1000 popular text queries using the Google image search engine. Most text queries are thus represented multiple times. Human raters were given four choices in ranking the similarity of images in the triplets: 1) both A and B were similar to Q; 2) both A and B were dissimilar to Q; 3) A was more similar to Q than B; 4) B was more similar to Q than A. Each triplet was rated by three different humans. If all three ratings were the same, the triplet was included in the dataset.

The Wang set contains an extremely wide variety of images due to its creation through sampling popular Google image searches. A random sampling of the image categories returns `Lynda Carter, Paris skyline, Empire State building, brunette, Bob Marley, Angora Rabbit, Jeep Liberty, 2 Fast 2 Furious, Shemar Moore, soccer ball, motorbike racing, Brittany Murphy`. A plurality of classes refer to people, mostly celebrities.

We display in Table 5.3 a random sampling of triplets. In contrast to the Flickr data, which we expect to generate a non-trivial proportion of triplets (p, p_+, p_-), where all three images are relatively dissimilar, the Wang set has a high proportion of triplets where all three images are extremely similar. Despite the requirement of unanimous agreement by the three human raters in the creation of this dataset, we feel that some examples may be mislabeled or should not have been included in the dataset. In Table TODO, we show a few examples demonstrating the relatively narrow margin between similar and dissimilar pairs. , and in Table TODO a few examples demonstrating the necessity of learning a general image invariants.

Though Wang et al. took steps to ensure a fairly clean dataset by requiring unanimous rankings by three different rankers, we feel the dataset is not particularly clean and in fact contains a non-trivial number of ambiguous similarity rankings. As show in Table 5.4, the watermarked Monument Valley image seems less similar, as does the pink guitar, because of the presence of a magazine. There are many examples similar to the Nirvana triplet where the differences between p, p^+, and p^- are incredibly subtle. We do not remove these examples to "clean" the dataset, but mention them only to note that though the Wang set was created through three unanimous decisions, that does not mean we should expect it to be entirely pure.

This is not the say that the dataset is malformed. There are many instances of easily separable rankings, such as the one shown in Table 5.5.

In 6, rather than attempt to quantify our models' performances on what we consider to be easy or hard triplets, we will instead compare performance to general baseline techniques.

In Table 5.6, we see that the human rankers rely quite often on image variants. For New

Image Query	**Base Image** (p)	**Similar Image** (p_+)	**Dissimilar Image** (p_-)
New York City			
Bart Simpson			
Sonic boom			
Column1d	Column2d	Column3d	

Table 5.3: Random triplets from Wang Set

Image Query	Base Image (p)	Similar Image (p_+)	Dissimilar Image (p_-)
Monument Valley			
Guitar			
Nirvana			

Table 5.4: Hard triplets from Wang Set

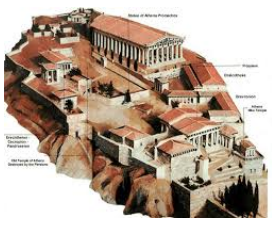
Image Query	Base Image (p)	Similar Image (p_+)	Dissimilar Image (p_-)
Parthenon			

Table 5.5: An easy triplet from the Wang Set

Image Query	**Base Image** (p)	**Similar Image** (p_+)	**Dissimilar Image** (p_-)
New York City			
Sydney Opera House			
Michelangelo			
Picasso			

Table 5.6: Variant reliant triplets from Wang Set

York City, zoom variance is important; for the Sydney Opera House and Michelangelo, illumination and stereoscopic variance comes into play; and for Picasso, the rankers seem to have picked up on some similarity in hue.

5.3.1 Quantifying the Amount of Fuzziness

We quantify the fuzziness in our Flickr dataset labels by extracting image embeddings using a model trained on ImageNet. We compute the mean and median distances for positive pairs and negative pairs as well as the percentage of triplets with ranking violations, where the positive distance is greater than the negative one.

We find that our sampling techniques do indeed split our data into roughly more similar and less similar pairs. We do see that our task is quite difficult. In contrast to Krause et al. study, which had roughly 16% fuzziness per class, we have an average fuzziness of TODO.(23)

Dataset	Median Positive Distance	Median Negative Distance	Mean Positive Distance	Mean Negative Distance	Ranking Violat
$\mathcal{P}_{1,2000,2013-2015}$	$<$ 1m	$>$ 2000m	Jan 2013 - Dec 2015	176000	
$\mathcal{P}_{1,2000,2015}$	$<$ 1m	$>$ 2000m	Jan 2015 - Dec 2015	140800	
$\mathcal{P}_{1,2013-2015,user}$	$<$ 1m, same user	$<$ 1m	Jan 2013 - Dec 2015	57600	
$\mathcal{P}_{1,2013-2015,tl}$	$<$ 1m	$<$ 1m, timelapse	Jan 2013 - Dec 2015	54400	
$\mathcal{P}_{10,2013-2015,user}$	$<$ 10m, same user	$<$ 10m	Jan 2013 - Dec 2015	118400	
$\mathcal{P}_{10,2013-2015,tl}$	$<$ 10m	$<$ 10m, timelapse	Jan 2013 - Dec 2015	-	
$\mathcal{P}_{30,2013-2015,user}$	$<$ 30m, same user within 2 hours	$<$ 30m	Jan 2013 - Dec 2015	198400	
$\mathcal{P}_{Middlebury}$	Middlebury pairs	random	N/A	32000	
$\mathcal{P}_{1,2000,01\_2014}$	$<$ 1m	$>$ 2000m	Jan 2014	32000	
$\mathcal{P}_{1,2000,02\_2014}$	$<$ 1m	$>$ 2000m	Feb 2014	32000	
$\mathcal{P}_{1,2000,03\_2014}$	$<$ 1m	$>$ 2000m	Mar 2014	32000	
$\mathcal{P}_{1,2000,04\_2014}$	$<$ 1m	$>$ 2000m	Apr 2014	32000	
$\mathcal{P}_{1,2000,05\_62014}$	$<$ 1m	$>$ 2000m	May 2014	32000	
$\mathcal{P}_{1,2000,06\_2014}$	$<$ 1m	$>$ 2000m	Jun 2014	32000	
$\mathcal{P}_{1,2000,07\_2014}$	$<$ 1m	$>$ 2000m	Jul 2014	32000	
$\mathcal{P}_{1,2000,08\_2014}$	$<$ 1m	$>$ 2000m	Aug 2014	32000	
$\mathcal{P}_{1,2000,09\_2014}$	$<$ 1m	$>$ 2000m	Sep 2014	32000	
$\mathcal{P}_{1,2000,10\_2014}$	$<$ 1m	$>$ 2000m	Oct 2014	32000	
$\mathcal{P}_{1,2000,11\_2014}$	$<$ 1m	$>$ 2000m	Nov 2014	32000	
$\mathcal{P}_{1,2000,12\_2014}$	$<$ 1m	$>$ 2000m	Dec 2014	32000	
$\mathcal{P}_{1,2000,01\_2015}$	$<$ 1m	$>$ 2000m	Jan 2015	32000	
$\mathcal{P}_{1,2000,02\_2015}$	$<$ 1m	$>$ 2000m	Feb 2015	32000	
$\mathcal{P}_{1,2000,03\_2015}$	$<$ 1m	$>$ 2000m	Mar 2015	32000	
$\mathcal{P}_{1,2000,04\_2015}$	$<$ 1m	$>$ 2000m	Apr 2015	32000	
$\mathcal{P}_{1,2000,05\_2015}$	$<$ 1m	$>$ 2000m	May 2015	32000	
$\mathcal{P}_{1,2000,06\_2015}$	$<$ 1m	$>$ 2000m	Jun 2015	32000	
$\mathcal{P}_{1,2000,07\_2015}$	$<$ 1m	$>$ 2000m	Jul 2015	32000	
$\mathcal{P}_{1,2000,08\_2015}$	$<$ 1m	$>$ 2000m	Aug 2015	32000	
$\mathcal{P}_{1,2000,09\_2015}$	$<$ 1m	$>$ 2000m	Sep 2015	32000	
$\mathcal{P}_{1,2000,10\_2015}$	$<$ 1m	$>$ 2000m	Oct 2015	32000	
$\mathcal{P}_{1,2000,11\_2015}$	$<$ 1m	$>$ 2000m	Nov 2015	32000	
$\mathcal{P}_{1,2000,12\_2015}$	$<$ 1m	$>$ 2000m	Dec 2015	32000	

Table 5.7: Dataset Fuzziness

Chapter 6

Experiments

We have three main experimental goals:

1. Prove it is possible to learn a useful image embedding function from publicly available and fuzzily labeled data
2. Investigate, and quantify if possible, the effects of data fuzziness on model learning rates and generalizability
3. Investigate how the presence of certain invariants in our data affects model generalizability

6.1 Weakly Supervised Image Embedding

Our main experiment, to find a network formulation which can learn a useful image embedding from fuzzily labeled data, is represented by the pipeline shown in Figure 4.1. As a recap, we compare two different methods used to train the networks that compose C: 1) concatenate the vector output from a frozen I with the vector output from V, and update V only; and 2) take only the vector output from V and update V. We designed V to arbitrarily produce an output vector of size 1024. We use an unmodified ResNet50 model pretrained on the ImageNet dataset as I, which has an output vector of size 2048. We will refer to the training of these methods as T_c^1 and T_c^2. We refer to the training of the blending network as T_b. We summarize some basic statistics about these training regimes in Table 6.1. We experiment with different layer widths for the small blending model, but on the whole, the number of training parameters is magnitudes smaller than the training of the C units.

Training is done on either a Tesla K20m GPU with 5GB of memory or on a Tesla P100 GPU with 16GB of memory. Do to memory constraints, any models run on the K20m GPU are trained with batch sizes of 6. Models trained on the P100 GPU are trained with batch sizes of 32. All training is single GPU.

Training Method	Units Trained	Trainable Parameters	Output Size
T_c^1	V (I frozen)	47,822,464	3072
T_c^2	V	34,587,776	1024
T_b	B (V and I frozen)	393,216 to 12,582,912	128 to 1024

Table 6.1: Training statistics

6.1.1 Optimizer Experiments

We considered several optimizers: stochastic gradient descent (SGD), RMSprop(**?**), Adagrad(**?**), Adadelta(**?**), Adam(**?**), and Nadam(**?**). Before beginning training, we run a small subsampling of our data to select our optimizer based on validation loss. While Nadam proved to be the most volatile, often causing exploding gradient updates, it also proved the best at finding successively deeper local minima. As we show in Figure **??**, the loss space of our problem is highly rough. Nadam is Adam with Nesterov momentum. The Adam optimizer essentially combines momentum, which averages the direction of gradient updates with an exponential decay parameter to find the proper update vector direction, with RMSprop, which determines the direction of the update vector. Nadam is an adaptive-learning method, meaning that a learning rate schedule is not necessary. We use standard hyperparameters for Nadam, with an initial learning rate of 0.002, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$.

6.1.2 Training Module C

Despite the extreme fuzziness of our labels as shown in Table 5.7, we find that our choices of architecture, optimizer, and loss function are able to quickly locate local minima. Validation loss plummets without delay, often reaching inflection points in the loss curve within three training epochs. Learning typically has enough momentum to escape local minima and find deeper minima.

As our data is weakly labeled, it is impossible to quantify model performance with any sort of validation or test accuracy. Instead we must use the Wang dataset for a hard accuracy quantification. We present validation losses in Table **??**. Because these validation losses are for different datasets, however, we do not use the normal approach of selecting a training method with the lowest validation loss and reporting its test accuracy

6.2 Investigation of Data Fuzziness

6.3 Image Invariants

Chapter 7

Results and Conclusion

7.1 Results

7.2 Conclusion

Bibliography

[1] A. Angelova, S. Zhu, and Y. Lin, "Image segmentation for large-scale subcategory flower recognition," in *Applications of Computer Vision (WACV), 2013 IEEE Workshop on*. IEEE, 2013, pp. 39–45.

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

[3] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 4, p. 98, 2015.

[4] T. Berg, J. Liu, S. Woo Lee, M. L. Alexander, D. W. Jacobs, and P. N. Belhumeur, "Birdsnap: Large-scale fine-grained visual categorization of birds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2011–2018.

[5] A. Bergamo and L. Torresani, "Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach," in *Advances in Neural Information Processing Systems*, 2010, pp. 181–189.

[6] K. W. Bowyer and P. J. Flynn, "The nd-iris-0405 iris image dataset," *arXiv preprint arXiv:1606.04853*, 2016.

[7] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," *IJPRAI*, vol. 7, no. 4, pp. 669–688, 1993.

[8] W.-T. Chu, X.-Y. Zheng, and D.-S. Ding, "Image2weather: A large-scale image dataset for weather property estimation," in *Multimedia Big Data (BigMM), 2016 IEEE Second International Conference on*. IEEE, 2016, pp. 137–144.

[9] C. Cortes, L. Kabongo, I. Macia, O. E. Ruiz, and J. Florez, "Ultrasound image dataset for image analysis algorithms evaluation," in *Innovation in Medicine and Healthcare 2015*. Springer, 2016, pp. 447–457.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[12] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman, "Learning object categories from internet image searches," *Proceedings of the IEEE*, vol. 98, no. 8, pp. 1453–1466, 2010.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[14] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward

neural networks." in *Aistats*, vol. 9, 2010, pp. 249–256.

[15] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2. IEEE, 2006, pp. 1735–1742.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[17] ——, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[18] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[19] A. K. Jain, N. K. Ratha, and S. Lakshmanan, "Object detection using gabor filters," *Pattern recognition*, vol. 30, no. 2, pp. 295–309, 1997.

[20] H. Jegou, F. Perronnin, M. Douze, J. Sánchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 9, pp. 1704–1716, 2012.

[21] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[22] G. Koch, "Siamese neural networks for one-shot image recognition," Ph.D. dissertation, University of Toronto, 2015.

[23] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei, "The unreasonable effectiveness of noisy data for fine-grained recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 301–320.

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

[25] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2. IEEE, 2006, pp. 2169–2178.

[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[27] L.-J. Li and L. Fei-Fei, "Optimol: automatic online picture collection via incremental model learning," *International journal of computer vision*, vol. 88, no. 2, pp. 147–168, 2010.

[28] T.-Y. Lin, Y. Cui, S. Belongie, and J. Hays, "Learning deep representations for ground-to-aerial geolocalization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5007–5015.

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[30] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[31] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[32] P. Paudyal, R. Olsson, M. Sjöström, F. Battisti, and M. Carli, "Smart: a light field image quality dataset," in *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016, p. 49.

[33] P. Pouladzadeh, A. Yassine, and S. Shirmohammadi, "Foodd: food detection dataset for calorie measurement using food images," in *International Conference on Image Analysis and Processing*. Springer, 2015, pp. 441–448.

[34] A. Rejeb Sfar, N. Boujemaa, and D. Geman, "Vantage feature frames for fine-grained categorization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 835–842.

[35] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu, "Unsupervised joint object discovery and segmentation in internet images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 1939–1946.

[36] O. Russakovsky, J. Deng, Z. Huang, A. C. Berg, and L. Fei-Fei, "Detecting avocados to zucchinis: what have we done, and where are we going?" in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2064–2071.

[37] F. Schroff, A. Criminisi, and A. Zisserman, "Harvesting image databases from the web," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 4, pp. 754–766, 2011.

[38] J. Shi, S. Zhou, X. Liu, Q. Zhang, M. Lu, and T. Wang, "Stacked deep polynomial network based representation learning for tumor classification with small ultrasound image dataset," *Neurocomputing*, vol. 194, pp. 87–94, 2016.

[39] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[40] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[41] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv:1602.07261*, 2016.

[42] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[43] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu, "Learning fine-grained image similarity with deep ranking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1386–1393.

[44] T. Weyand, I. Kostrikov, and J. Philbin, "Planet-photo geolocation with convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 37–55.

[45] J. Wu and J. M. Rehg, "Centrist: A visual descriptor for scene categorization," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 8, pp. 1489–1501, 2011.

[46] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 3485–3492.

[47] Z. Xu, S. Huang, Y. Zhang, and D. Tao, "Augmenting strong supervision using web data for fine-grained categorization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2524–2532.

[48] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1794–1801.

[49] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.

[50] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, 2014, pp. 487–495.