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 6, 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

This thesis would not have been possible without the love, kindness, and help of a great many people.

Firstly, I would like to thank my advisor, Szymon, for all of his guidance and his amazingly fast email response time.

Mom, Ann, and Dad, thank you for helping me become the person I am today. I will never be able to match the sacrifices you have made in helping me get to this point.

Thank you, Hannah, for the love. James and Dee, I would not have made it here without you. Lucia, Eddie, Davy, Katz, and Dominguez, you have made this place so wonderful. Finally, I'm so glad to have met you, Eliot.

Contents

Abstract

Learning an embedding useful for fine-grained image similarity on a human perception level requires detecting three general classes of attributes: 1) high level semantic characteristics, such as emotional qualities or aesthetic styles; 2) complex visual descriptors such as the objects that comprise the focus on an image; and 3) low level descriptors such as textures and colors which can influence human perception of the first two classes. In this paper, we present a deep Siamese network that uses multiple modules to allow for the learning and blending of different attribute classes. We train the network using weak supervision by sampling publicly available Flickr images for image pairs deemed similar and dissimilar by a set of constructed heuristics. Experiments show that this weakly supervised model learns a highly generalizable similarity embedding which surpasses the effectiveness of ImageNet trained models and appears to span a space mostly outside that described by ImageNet.

Chapter 1

Introduction

1.1 Image Similarity

What does it mean for two images to be similar? Can two images be similar if they are focused on different types of objects? Which two teapots in Figure 1.1 are the most similar?

(a) A teapot (b) A teapot (c) A teapot

Figure 1.1: Three teapots

Here we caption all three teapots with the same innocuous description, which hides the difficulties in categorizing objects and, by extension, images. Perhaps more descriptive captions would be `(a) A white teapot, (b) A Japanese teapot, (c) A teapot in a restaurant`. Even more verbose descriptions might note that the white teapot appears to be a digital rendering, that the middle teapot is lit as if in a museum display, and that the third teapot is partially out of frame and not the only object of importance in the image. Esteemed computer science faculty or even just middling computer science undergraduates might exclaim that the first teapot is the most remarkable of the three. The Utah teapot, as it is known, is perhaps the most famous teapot in the world due to its use as a stock computer graphics model.

We see that the task of understanding images is complex. There are an infinite number of facets through which we can describe any given image. Some of these can be quantified: presence of teapot, color, hue, contrast, main object, number of foreground/background

divisions, lighting, and so on. Other qualities are far more nebulous. `Famous` as a descriptor for the first image seems questionable–even if we agreed it was "famous," how would one quantify fame? Yet clearly there is a strong hidden attribute to the Utah teapot, because when considering the image of Lenna Soderberg in Figure 1.1, another famous image from computer graphics, some readers of this paper would agree that Lenna is in fact more similar to the Utah teapot than either of the actual teapots.

Utah teapot Lenna Soderberg

Figure 1.2: A similar pair

In the same vein, an image of the Empire State Building might be considered more similar to the Eiffel Tower than to a standard apartment complex in Brooklyn. Two red teapots might be more similar to each other than to a green teapot, unless one of the red teapots has the same type of long spout as the green one and the other doesn't.

In the past, the field of computer vision has considered images to be similar so long as both images focused on the same type of object. Gradually, our definition of what constitutes the same object has narrowed. Figure 1.1 shows two flowers of different species and two differently posed girls that were matched as similar in a 1998 study on perceptual similarity.[49] With some revolutionary new algorithms, it has started to become possible to train models to detect fine-grained similarity, which differentiates objects and images not only on their coarse physical characteristics but also on highly complex combinations of these characteristics, as well as soft concepts such as "Fame."

Similar girls Similar flowers

Figure 1.3: Similar pairs from 1998

Learning to differentiate images on such a scale has seen use in wide-ranging applications, such as in scientific research in classifying different species, of insects and birds; in facial recognition software; in helping consumers purchase products according to aesthetic preferences; and in building search-by-example databases.[29][11][4][64]

The task of detecting image similarity boils down to learning how to represent images. If we decide two teapots are similar if they are both red, we are choosing to represent our images with a field that denotes the presence of a certain color. Once we have a complex enough representation for an image, a formulation for similarity is easily computed as the distance between the two images in their embedded representations.

Learning image representations is important for almost every task in machine vision, including image classification, understanding scenes, semantic segmentation, object recognition, answering questions about images, captioning videos, and even geolocating images. The methods of extracting image representations have changed greatly over the years, moving from using manually defined features like histograms of oriented gradients (HOGs)[39] and scale-invariant feature transforms (SIFT)[14], to the current state-of-the-art technique of extracting feature vectors from convolutional neural networks, beginning with the seminal work of Krizhevsky et al.[30]. We will discuss these techniques and the ways in which image representations have changed in more depth in Chapter 2. In our view, there are two key negative characteristics associated with deep learning techniques as applied to machine vision: 1) Most deep learning is strongly supervised, which means that techniques are only as good as the magnitude or quality of their data; and 2) The ability of deep learning techniques to learn and enforce image invariants with incredible effectiveness has masked a detrimental shift of attention away from some features which are important to human-level perception.

1.2 Data Constraints

As methods for extracting image representations have shifted from manual definition to the more black box art of training deep convolutional networks, our reliance on large quantities of training data has increased. Though deep learning requires relatively little prior knowledge about a classification task in contrast with classical vision techniques, which require domain knowledge, or Bayesian techniques, which build generative models with carefully tuned priors, deep learning techniques can still require tens of thousands or millions of images per class. The flexibility that deep learning advocates tout in comparison to classical and Bayesian techniques really only disguises an extreme inflexibility of a different form. In initial years, deep learning's need for vast amounts of data was satisified by the popularization of large scale image datasets such as ImageNet, which contains 14,197,122 images belonging to 1000 classes[15]; the MIT Places dataset, which contains 7 million images for scene classification[72]; the SUN scene classification database[67]; and the Microsoft COCO dataset of 2.5 million images for common objects in context[37]. These and other datasets have allowed researchers to build models that are highly adept at basic image classification tasks[51]. In a review of deep learning models trained on these massive datasets with many basic image classes (ImageNet contains classes for many animals, plants, and basic 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, e.g., distinguishing dogs from wolves.[51] However, as the field has matured and begun turning its focus to more challenging tasks, such as fine-grained similarity rather than just similarity, it has become apparent that more expansive datasets, or at least more

robustly labeled datasets, are needed.

These datasets are often built by hiring mechanical Turkers, who are given instructions on how to label or annotate millions of images. A quick search for image datasets published in 2016 returns ones for irises, ultrasounds, weather property, tumors, light fields, and food calories.[9][13][12][56][44][46] Yet although we have been able to train more and more specific models for these specific types of image classification, this research still relies on the gathering of accurately labeled data. It is infeasible, both from manpower and expense standpoints, to gather large quantities of data for every possible image understanding task.

1.2.1 Alternatives to Strongly Supervised Learning

The field of machine learning can be roughly segmented into three types of categories:

1. Supervised learning: models are trained on example inputs and desired outputs, with the desired outputs known and labeled with a high degree of correctness
2. Unsupervised learning: models are trained without knowing the desired outputs; models are intended to detect structures or patterns in the input data without guidance
3. Reinforcement learning: models learn to interact with an environment with possible rewards and punishments

Semi-supervised learning, which provides partially labeled data for the model, and weakly supervised learning, which provides possibly incorrectly labeled data to the model, are other forms of learning. While reinforcement learning has limited applicability to agent-less computer vision problems like similarity, strong interest in unsupervised, semi-supervised, and weakly supervised learning has arisen in the field, both in response to the recognition that large-scale and fine-grained strong supervision is infeasible, and because of general theoretical interest. Rather than label tens of millions of images, semi-supervised learning might provide several tens of thousands of labeled images and attempt to self-label unlabeled images to increase the amount of available training data. Weakly supervised learning accepts that there will be some non-trivial level of error in the labeled data. Many researchers have begun using weakly labeled but publicly available, and thus vast, image data, such as images mined through Google image searches, to train models.

Such techniques can be surprisingly powerful. In training a classifier on 14,000 different classes, Kraus et al. generate a training set by pulling Google image data and exceed state-of-the-art classification accuracies on the CUB-200-2011 dataset, despite the fact that images returned by their image searches had 16% out-of-class error.[29] One type of weakly supervised learning, called distantly supervised learning, relies on labeling datasets using some heuristic. Xu et al.[68] use existing datasets to learn feature representations and part-based object classifiers. They then extract accurate part labels from fuzzy web image data. If the labeling heuristic used is accurate enough, this kind of weakly supervised learning can be nearly as good as strongly supervised techniques. The possible advantages of non-supervised learning are obvious–Kraus et al.'s landmark work is notable not only

because it exceeded state-of-the-art results with weak supervision, but also because they were able to scale their classifier to detect 14,000 different classes by expending only extra machine hours, and no man hours.

1.2.2 Image Invariants

The machine vision field's reliance on deep learning has also caused learning invariants to become an almost unquestioned tenant for building a vision model. Krizhevsky et al.'s 2012 submission to the ILSVRC notes that their use of data augmentation, which has become standard in training deep convolutional neural nets, "approximately captures an important property of natural images, namely, that object identity is invariant to changes in the intensity and color of the illumination."[30] Even before Krizhevsky, libraries for HOGs supported flip invariants, and the popular scale-invariant feature transformation technique, pioneered in 1999, obviously prioritizes scale-invariance.[63][42]

The proliferation of deep CNNs has been matched by work in augmenting data to learn these and other types of invariance. Cropping and zooming images, applying rotations, shears, translations, flips, and out-of-plane rotations are all common augmenting techniques used to learn the associated invariance. It seems that even without strong data augmentation techniques, the architecture of deep CNNs naturally lend themselves to learning image invariants. In a study on image invariants, Lenc and Vidaldi find that common invariants present in natural samplings of object classes, such as horizontal flips and rescaling transformations, will be learned by the third or fourth convolutional block in a deep network. The presence of these invariants has to do with the construction of the ImageNet dataset, which has millions of images for just 1000 classes, thus ensuring that a wide variety of each type of class is represented, with each example having its own lighting, rotation, scale, etc, and with all examples labeled as identical. Lenc and Vidaldi conclude that the representation of deep CNNs transform in predictable ways in response to image augmentations, and that these representations are largely interchangeable among different architectures, at least for the invariants learned in the shallower layers of a network. Deeper layers, which learn task-specific filters, are more specific to their architecture.[34] The learning of these invariants has been proven time and time again to be extremely useful across a spectrum of vision tasks, and this is intuitive. Our three teapots are all teapots despite differences in color or lighting conditions. Empirically, the learning of intensity and color invariance improved Krizhevsky et al.'s top-1 error rate by over 1%.[30]

Yet fine-grained similarity pushes a little against this tenant of machine vision. On a coarse level, invariants are useful for recognizing that two teapots are more similar to each other than they are to a dog, despite changes in rotation, lighting, color, etc. But when comparing three teapots, image variants must come back into play. We often compare vision models with human-level performance, and when it comes to fine-grained similarity, combinations of color and lighting and other traditionally ignored variants play important roles in our estimations of aesthetic quality. A good image similarity model thus must not only be able to detect the physical characteristics of objects, but must also be able to determine semantic information about the image.

1.3 Motivations

The direction and motivation for our research comes directly from considering these two key characteristics of deep vision models. We seek to

1. Learn an embedding function useful for comparing the similarity of images on a very fine-grained level
2. Learn this embedding using a weakly labeled and publicly available dataset
3. Prove that our results are generalizable to tasks beyond those that would normally be serviced by our dataset

Along the way, we investigate the importance of certain invariants in our learned similarity model and attempt to quantify the trade-offs in extra training time required for training with a weakly labeled dataset.

We use geo-tagged images uploaded to Flickr to learn our embedding by labeling pairs of images as similar or dissimilar using heuristics derived from the meta-data associated with each image. We explore how well these heuristics 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. As discussed, while we recognize the importance of detecting image invariants, we also attempt to train a model that can judge whether image variants should be more strongly considered in computing image similarity for certain examples than they would be by a more traditional deep architecture. To this end, we train a multi-scale and multi-module Siamese network with a pairwise loss function, with the objective of learning that pairs of images more likely to be considered similar by humans should be closer together in our image embedding space than are pairs likely to be considered relatively dissimilar. One module, which is relatively shallow, is trained to preserve the presence of variants in our image embedding, and the other, which is of standard depth, is trained to enforce the presence of invariants in the embedding.

1.4 Outline

This paper is organized as follows. In Chapter 2 we present a more in-depth discussion about the background of vision research, especially as pertaining to the techniques used to generate image embeddings and relating to image similarity. In Chapter 3, we discuss our publicly available dataset, the Flickr set, as well as a dataset used for evaluation and a dataset used for additional exploration into image similarity. In Chapter 4, we present our network formulations and training techniques. We report and discuss our experimental results in Chapter 5. We conclude with a summary and an outline of potential future work in Chapter 6.

Chapter 2

Background

2.1 A History of Image Similarity

In 1999, John Eakins summarized the hierarchy of detectable features in the field of content-based image retrieval, whose goals are similar to those of image similarity.[17] A level one content-based retrieval system supports the retrieval of images based on features such as color, texture, shape, and spatial locations of objects. Level two allows for queries of complex aggregations of features that are typically nameable by humans, such as "find images that contain dogs." Level three requires an image retrieval system capable of inferring a semantic quality about a scene, such as, "Is this restaurant romantic?" There are no hard boundaries between these levels, but building a system on par with human perception requires jumping from recognizing low level features like colors and edges to recognizing concepts associated with a particular visual stimulus. This jump has been dubbed "the semantic gap."

The semantic gap has still not been bridged. Work in the earliest days of machine vision was focused on extracting level one features and attempted to aggregate these into level two descriptors. Today, a deep learning revolution has brought the field to the cusp of level three systems, but as system have become more sophisticated, the difficulty of achieving incremental gains has also increased. We will briefly discuss low-level descriptors and some classical techniques used to extract them before formalizing some of the deep learning research that serves as the direct base for our work.

2.1.1 Low-Level Descriptors and Classical Techniques

Table 2.1 contains a general set of low-level feature descriptors. This overview is far from comprehensive but gives a general idea of the types of low-level features used in image similarity and other image understanding research.

Color-based features are perhaps the easiest to define. Features are represented in various color spaces that are thought to be close to human perception. These include RGB,

Color [43]	Dominant Color	Statistical properties of dominant colors
	Scalable Color	Color histogram in HSV space with fixed quantization
	Group of Frames	Extension of scalable color to groups of pictures
	Color Structure	Localized color distribution in Hue-Min-Max-Difference space
	Color Layout	Gridded layout of dominant colors
Texture	Homogenous Texture	Image Fourier transform statistics [48]
	Texture Browsing	Directionality, regularity, coarseness of textures [40]
	Edge Histogram	Frequency and directionality of brightness changes [66]
Shape	Region-based Shape	Pixel distribution within a 2D object region [7]
	Contour-based Shape	Distribution of pixels within object contour [7]
Location	Region Locator	Element location in image
	Spatio Temporal Locator	Element location in temporal domain

Table 2.1: Some general image descriptors

LAB, LUV, HSV, YCrCb, and HMMD spaces. For instance, the scalable color descriptor, which uses a Haar transform to allow it to scalably represent features, is computed in Hue-Saturation-Value (HSV) space and the color structure descriptor, which uses a small structuring window to localize color distributions, is computed in Hue-Min-Max-Difference (HMMD) space. Within these spaces, many other statistics such as color-covariance matrices and color histograms can be computed.

Texture features are often obtained through Gabor filters, wavelets, and local statistical methods. Gabor filters in particular prove useful for edge detection and texture segmentation.[65][41] Statistical methods like Gaussian random Markov fields are used to measure local pixel interdependencies as a way of identifying textures.[45] The Tamura features detect coarseness, contrast, directionality, linelikeness, regularity, and roughness by computing statistics like frequency distributions and moments of pixel gray values.[61]

Statistical approaches are also used for shape-based features. Fourier descriptors determine Fourier transformed boundaries and Delaunay triangulation is used for discovering image segments.

These descriptors for low-level features all require a manual definition. Using a variety of these techniques allows for the creation of a bag-of-visual-words (BOV) representing different characteristics about color, shape, texture, image segments, and feature locations, and from there, an image can be described on a global level using more complex and often spatially hierarchical algorithms, which take into consideration patch-level statistics about features, spatial relations between local features, and other aggregated computations. For instance, Bouchard and Triggs note that many objects have rigid structures on small local levels. Local rigidity can easily be detected using these low-level descriptors, but in order to capture the variance of the shape of objects on a larger scale, Bouchard and Triggs use an expectation maximization algorithm to recursively assign extracted local features to a hierarchy of parent classes.[8] In a similar vein, Grauman and Darrell extract local features and use spatial neighborhood constraints to compare the overall distribution of locally detected features between images.[21] Other hierarchical feature representations, such as spatial pyramids, which transform images into locally oriented segmentations proved effective in a variety of

image tasks as well[69][19][31]. Another method of moving from local descriptors to more complex global descriptors that does not rely on hierarchically aggregating or comparing BOVs, is the fisher vector. Rather than quantize local visual words, the Fisher vector assume features are generated from a Gaussian mixture model and describes local image areas by their deviation from an expected feature distribution.[53]

As some techniques became increasingly adept and effective as building global descriptors from local low-level features, other feature extractors gained widespread use because they preserved feature representations even under image transformations. Histograms of oriented gradients (HOGs), which compute local intensity gradients and edge directions across a tiled image, proved adept at detecting humans in photos.[14] HOGs operate on local tiles, so they are invariant to transformations of the global image except for relative orientations of the object to be detected, and this proved useful for humans, which are usually upright. Similarly scale-invariant feature transforms, which detects scale, rotation, and illumination invariant local features, has proven especially useful in object recognition, 3D modeling, and robotic navigation.[39] Further work has been done to extend SIFT to include other invariances, such as color (CSIFT) and geometric scale (GSIFT).[1][38]

Overall, between work done in aggregating local descriptors into global features and in detecting invariant features, by 2012, level 2 image understanding had been achieved with reasonable success. The ImageNet classification challenge asks participants to classify images into one of 1000 possible classes, and by 2012, a weighted prediction between SIFT, CSIFT, GSIFT, and Fisher vectors was able to do so with a top-5 error of just 26.2%, where top-5 error is defined as the percentage of samples for which none of a model's top five guesses match the true label.[52]

2.1.2 The Deep Learning Revolution

The fields of image similarity and image understanding were completely reset in 2012 by the submission of Krizhevsky et al.'s deep convolutional neural network to that same 2012 ILSVRC ImageNet classification competition. Krizhevsky et al.'s model scored a top-5 error rate of 15.3%, which was almost 70% better than the second place weighted prediction by SIFT, CSIFT, GSIFT, and Fisher vectors.[30].

Krizhevsky et al.'s performance demonstrated that convolutional neural networks, which had been applied by LeCunn et al. in 1989 for handwriting recogniztion, and which had been theorized as capable of learning low, medium, and high level features by using nonlinear transformations, were feasible for use in general image understanding problems.[33]

2.1.3 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.[26][30][60] Besides convolutional layers, CNNs typically consist of pooling layers, activation layers, fully connected

layers, and a loss layer.

Convolutional Layers

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. An example of a kernel is the edge detection kernel:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

If a 5×5 image is fed to a network that applies this kernel, a $3\times$ 3 feature map will be returned, assuming it is specified that the kernel can only be applied to to patches of the image where signals overlap completely. Many other useful kernels exist, and by applying hundreds of these across input images, convolutional layers can detect basic evidence of edges, textures, and other low-level features. A kernel operating in the red channel of an image, for instance, will detect certain low-level color descriptors. Deep convolutional networks typically have hundreds of filters.

Pooling Layers

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. They also transfer evidence of descriptors to neighboring patches, eventually allowing this information to be known globally, if the network is deep enough relative to the input image.

Activation Layers

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

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{2.1.1}$$

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.[25]

$$f(x) = \begin{cases} x & x > 0 \\ \alpha x & x \leq 0 \end{cases} \tag{2.1.2}$$

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. For general tasks, typically a 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 proven highly effective for neural network regularization by preventing networks from becoming overly reliant on certain neural paths.[58]

The term deep CNN refers to a CNN that has many layers. This definition is highly variable: the popular VGG16 model has 16 layers[57] and popular versions of ResNet contain 34, 50, 101, and 152 layers[25].

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.

2.1.4 Deep Networks Today

Recent research has shown that increasing the depth of deep neural networks increases their performance.[60] Much work on different activation functions has allowed CNNs to become sparser and thus take up less memory, despite their increasing depth.[57][60] In combination, these advancements have allowed networks that are hundreds of layers deep to become standard, and these extremely deep models have continued to excel. In recent years, some deep learning models have 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%[25]. It is however, as He notes, possible to build models that are too deep. Past a certain point, accuracy saturates, and the addition of layers beyond this point of saturation leads to an increase in training error.

The features learned by deep learning models capture enough semantic meaning about images that they can be used to transfer learning from ImageNet trained models to other tasks with some success. 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.[3]

Importantly, in a departure from classical image processing techniques, 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. It has been shown through visualizations of the weights learned by deep networks that these network learn many of the same low-level features as do classical techniques. Typically, the first few

layers of a network will learn various edge detection filters. Many of these filters will be replicated throughout the network in slightly different fashions, which show that networks learn invariance for basic descriptors. And similarly to how much research in the 2000s focused on aggregating counts of local feature descriptors into larger parent patches of the image, many deep networks include pooling and fully connected layers, which effectively spread knowledge about locally detected features to neighboring patches of the network.

2.1.5 Image Invariants

Ever since Krizhevsky et al. popularized the use of deep CNNs in 2012, researchers have focused on ways of incorporating various image invariants in their deep models. Invariance is learned when many examples of the same class are fed to a model, forcing the model to generalize its learned representation of that class to ignore minor differences in the image. In practice, this is often achieved through data augmentation –e.g., flipping, rotating, shearing, zooming, and changing color intensities in images. But in many cases, datasets either already contain enough images that capture certain invariants, or datasets are curated with the goal of including certain invariances. For example, Hadsell et al. learn an invariant mapping for dimensionality reduction by taking images of a plane from a full span of 360deg angles.[22] Regardless of how invariance finds its way into the dataset, it has become standard for many tasks to verify that invariance is being learned. In demonstrating that an unsupervised model could detect the presence of high level features such as cat faces, Le et al. also use an out-of-plane rotation dataset to confirm that their model learned rotational invariance, which was particularly important since cat faces are often turned to the side.[32]

It is said that CNNs have translational invariance baked into their architecture. As CNNs apply kernels independently of location across input images, is it impossible for CNNs not to learn translational invariance of kernel activations. Lenc and Vedaldi's study on invariants in deep networks concluded that classification datasets contain enough examples with naturally occurring horizontal flips and rescaling for these invariants to be learned as well.[63] Rotational and vertical flip invariance is less commonly seen and are not usually naturally enforced.

2.2 Deep Learning for Image Similarity

2.2.1 Siamese Neural Networks

As noted by Wang et al.[64], the network structures that are effective at classifying images into object classes are not necessarily well-designed for detecting image similarity, especially when similarity detection must be fine enough not just to identify whether two objects are in the same class, but also to rank similarities within classes. The Siamese architecture, first proposed by Bromley et al. in 1994[10] 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 by Lin[36], matching visual similarity for product

design by Bell and Bala[4], comparing image patches[70], and one-shot image classification by Koch[28].

A Siamese net is a formulation of at least 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 network weights 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[4], 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 outputs from the CNN layer, multiple loss layers can be used, as well.

For the purposes of learning image similarity, only a simple formulation is required. The embeddings produced by each half of the Siamese network must be compared in some manner, and a standard Euclidean distance is usually used. Formally, we define the similarity of two images, $S(A, B)$, as the Euclidean distance of their feature embedded vectors, $f(A)$ and $f(B)$:

$$S(A, B) = ||f(A) - f(B)||_2^2 \tag{2.2.1}$$

A cosine similarity $\dfrac{f(A) \cdot f(B)}{||f(A)||||f(B)||}$ can also be used but this is less common. We note that though in the simple Siamese formula discussed here, $f(\cdot)$ is taken as the weight representation of the final convolutional block of a single CNN, we can construct a Siamese net that runs each image through mutiple CNNs or other feature extraction modules. In this case, $f(\cdot)$ would refer to the aggregate feature embedding found by all the modules. Embeddings might be aggregated via concatenation or another, more complicated technique.

We base our network design, which we discuss in 4, off of a fairly complex Siamese network with two different feature extracting modules and a blending model for their aggregation.

2.3 Weakly Supervised Learning

Deep learning usually requires massive amounts of labeled data and the problems with relying on supervised learning at a large scale 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 type of task and do not necessarily extend to novel concepts. As Russakowsky et al. discussed[51], models pretrained on general datsets like ImageNet were only good at dividing 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 have allowed deep learning techniques to tackle tasks such as differentiating species of flowers[2], leaves[47], and birds[5], 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 highly 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 data.[6][18][35][55]. 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 screens. 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.[50]

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[29]. They quantify the fuzziness (incorrectly labeled images) in their images classes to be an average of 16%, demonstrating that well-trained CNNs are still able to overcome significant fuzziness. It is this success that motivates the use of Flickr images, which we show in the next chapter, are even more difficult to label.

Chapter 3

Data

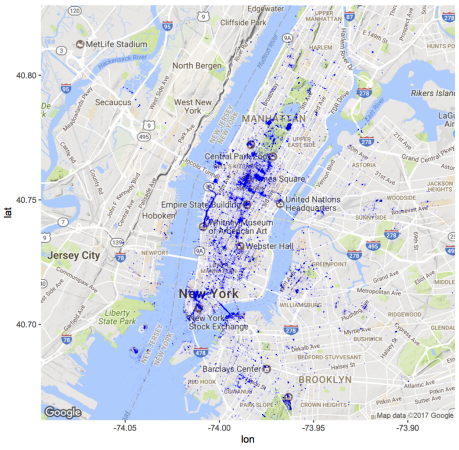
The success of Kraus et al.'s models despite the fuzziness of their data motivates our decision to build a corpus for image similarity and label it using our own heuristics. ImageNet is perhaps unable to advance beyond general classification and understanding because it is limited by its finite number of classes. Rather than replicate Kraus' approach and attempt to increase this number by a magnitude, we instead hope to learn an embedding function from classless data, which will better approximate the continuous nature of fine-grained image similarity.

3.1 Flickr Data

Our corpus is built by pulling 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. The data is quite dense in certain areas. 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.

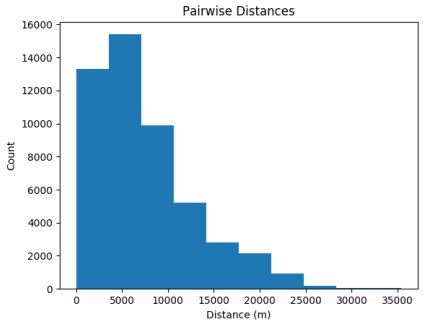
With 6 million images, there are roughly 1.8×10^{13} possible image pairs, which is infeasible to train a CNN on. To maximize the information learned from this corpus requires the development of smart heuristics for sampling positive and negative image pairs. The heuristics used in this paper are covered in Section 3.1.1.

(a) All images

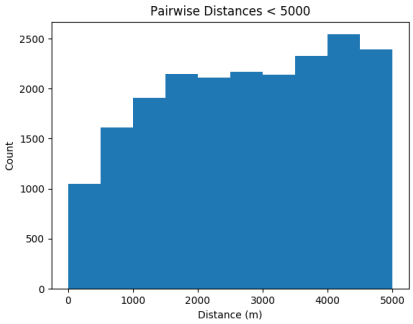


(b) Closer view of highest image density

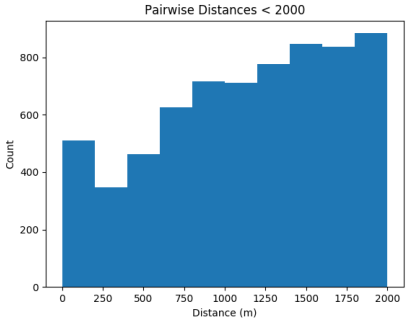
Roughly 0.6% of our image pairs have an image distance of one meter or less.



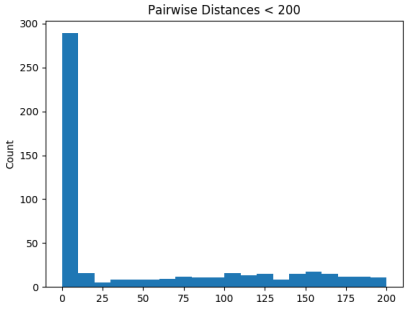
(a) All sampled pairwise distances



(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. Though the existence of semantic clusters provides the basis for believing semantically similar pairs can be created for model training, these clusters are never explicitely identified, nor are any other similar methods used which would more strongly supervise training pair creation.

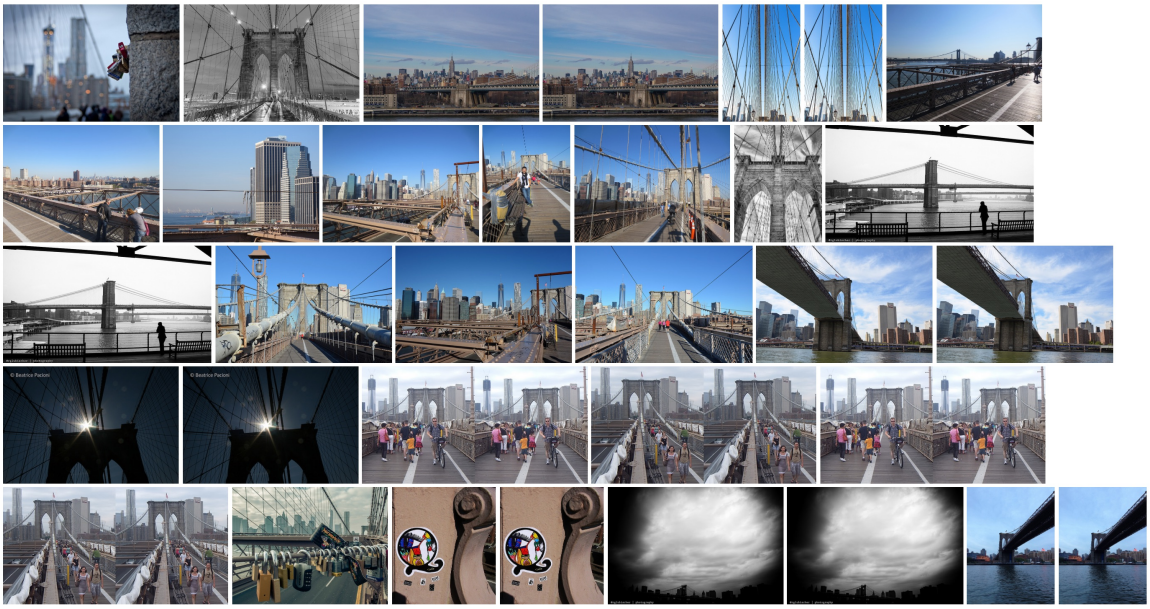


Figure 3.1: Photos from the Brooklyn Bridge (40.706 -73.996)

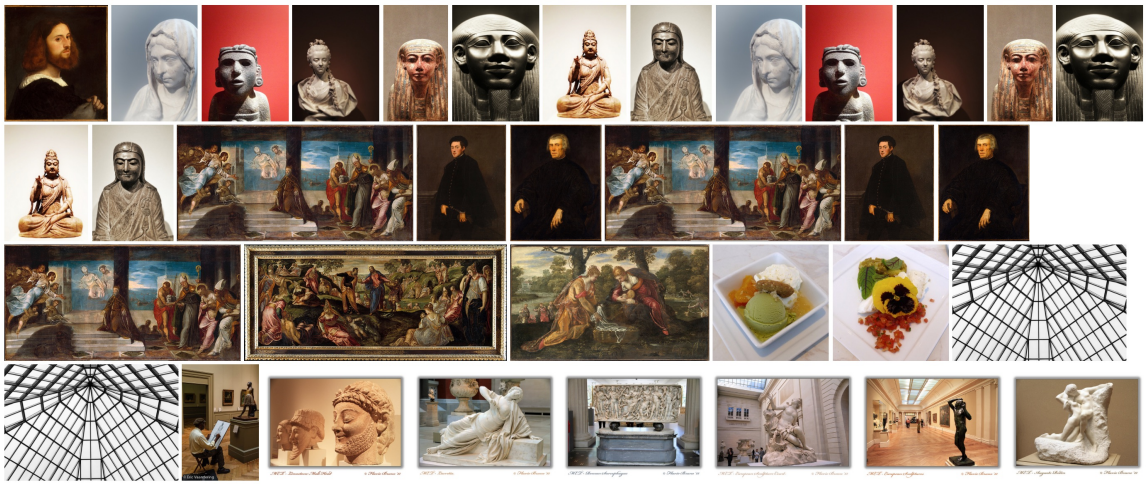


Figure 3.2: Photos from the Metropolitan Museum of Art (40.779 -73.963)

A non-trivial number of duplicate images, as seen in Figure 3.1, indicates aggressive deduplication is required when selecting positive image pairs for training, no matter the heuristic.

3.1.1 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

The use of distance heuristics as a proxy for image similarity is intuitive. Images taken at the same location are more likely to be of the same object or include similar aesthetic styles than are images that are far apart. 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.

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, an inability to precisely quantify this statistic is not overly concerning. This means two images with a Euclidean distance, d, between their coordinates can be, at worst, up to $d + 20$ meters away.

$\mathcal{P}_{21,2000}$ is formed 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 the maximum geolocation error, this means p, p^+ are within 11 meters of each other. From these positive pairs, images p^- that are farther than 2020 meters from p are sampled. The distance of 2020 meters is chosen to be longer than the maximum diameter of any eyeballed cluster in Figure 3.1. 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 21 meters and a minimum distance for p, p^- of 2000 meters. Many other distance-based datasets are created in the same fashion.

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 because of memory hashing issues. Otherwise we would have explored results for sets like $\mathcal{P}_{30,[50-100]}$, where negative samples lie farther than 50 meters but within 100

meters from the base image.

Temporal Heuristics

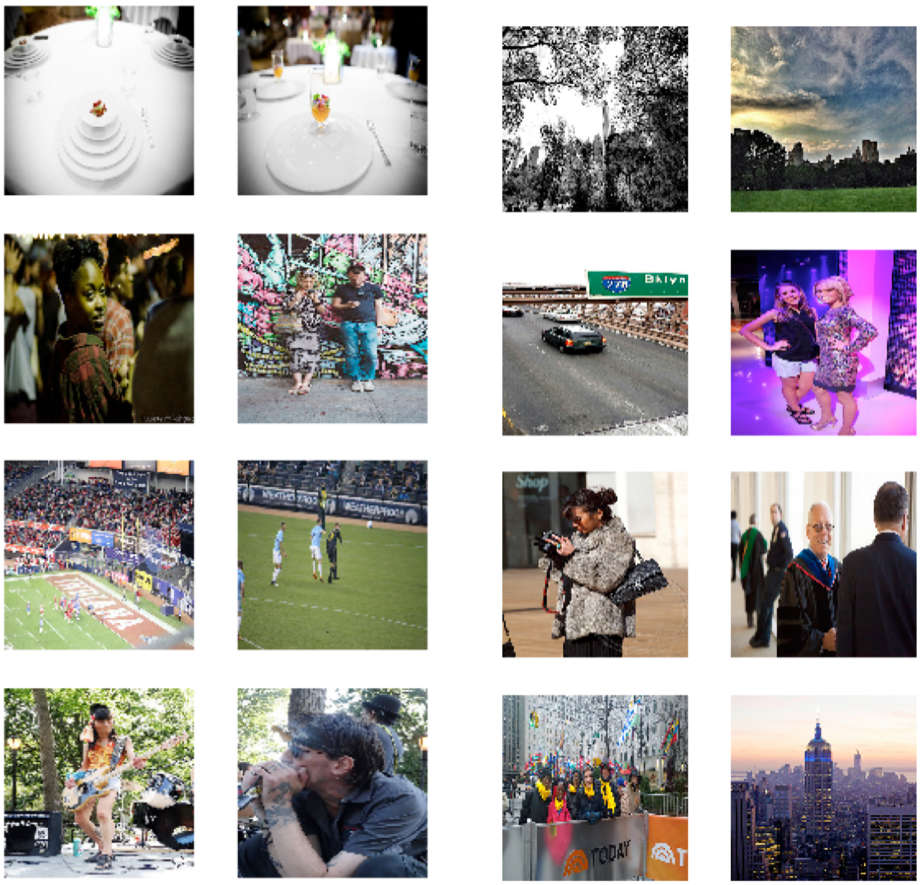
The Flickr corpus can be split into subsets divided by year and month. Applying distance heuristics across time divisions creates sets like $\mathcal{P}_{30,2000,June2013}$) and $\mathcal{P}_{10,2000,2013-2015}$), which refer to the distance set described intersected with the set of all possible image pairs from June 2013 and from 2013, 2014, and 2015 respectively. Datasets limited to one month are more likely 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, the 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. Less fuzziness can be enforced by requiring similar pairs to have been taken by the same user. We can denote this as $\mathcal{P}_{10,2000,June2013,user}$). Dataset are also created which 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 $\mathcal{P}_{10,2013,user}$ if the bounding ball is 10 meters and the images are from 2013. As seen in Figure 3.3, there are many reasons why photos taken by the same user would be more similar on average. There are similarities due to the style of the photographer (second pair), because of user interest (sporting events, third pair), and because a user has been to same places for the same reasons as himself (dinner, first pair).

Other

One user 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. These images are used to construct a datset where similar images are pairs sampled entirely from this time lapse, and dissimilar images follow a minimum distance heuristic. We refer to this as $\mathcal{P}_{timelapse,2000,June2013}$. Examples of time lapse phots are shown in Figure 3.4. As can be seen, the time lapse dataset will have very similar images as its positive examples, except with lots of noise from different lighting and weather conditions



(a) Photos taken by the same users (b) Photos taken by different users

Figure 3.3:

and the possibility of occlusions to the baseline image in the form of passing vehicles and other transient changes to the scene.

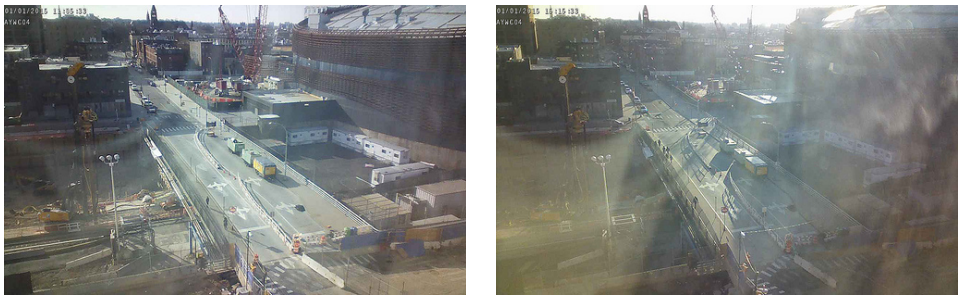


Figure 3.4: Two timelapse photos of the Barclays Center

Summary

In total, 35 main datasets are created with each designed to have various degrees of label fuzziness and to require our model to focus on different features. These are summarized in Table 3.1. There is an incredible amount of available data, so focus is placed on generating sets from the years 2013, 2014, and 2015. For brevity, we summarize only a subset of the datasets that we ran experiments on, since many of the results will be redundant.

3.2 Middlebury Stereo Data

While we can augment images to create datasets that include color, rotational, translational, and magnitude invariants, especially with the discovery of time lapse data in the Flickr set, it is more difficult to artificially construct a dataset that includes out of plane or 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 study of stereo algorithms.[54]

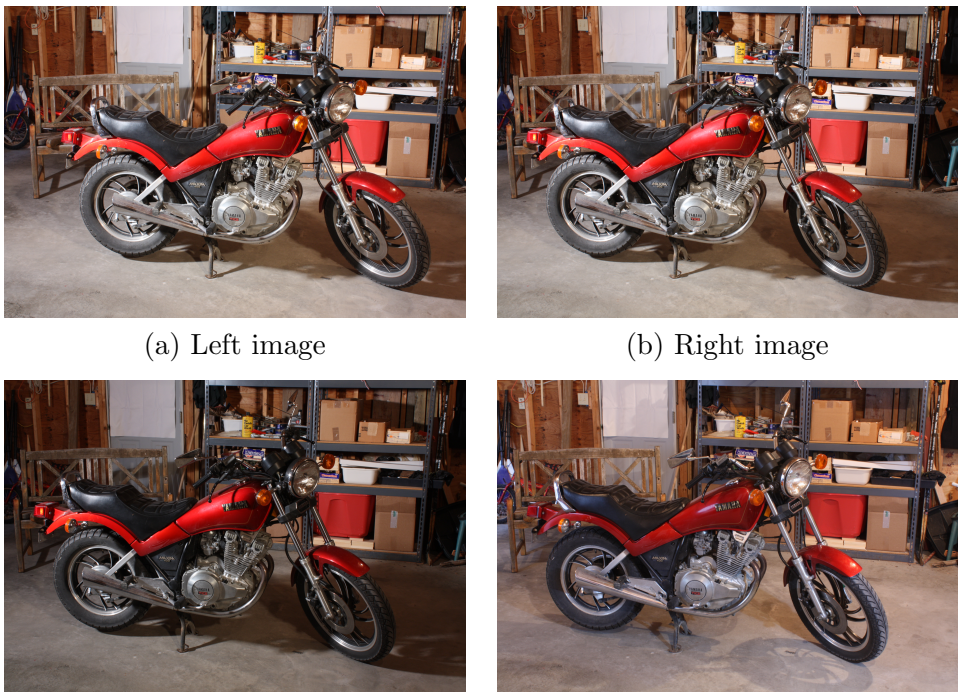
The dataset is very small, containing data for just 33 objects such as a motorbike, plants, and umbrella. There are four pictures per object–default left and right stereo images, a right image under different exposure, and a right image under different lighting conditions.

Examples are shown in Figure 3.2.

Several datasets are created with pairs from the Middlebury data taken as positive pairs, and negative pairs formed by sampling the Flickr corpus. 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, images are sampled from Flickr that are within 10 meters of each other and either taken by the same or different users.

Dataset	Positive Heuristic	Negative Heuristic	Time Frame	Size
$\mathcal{P}_{21,2000,2013-2015}$	$<$ 1m	$>$ 2000m	Jan 2013 - Dec 2015	176,000
$\mathcal{P}_{21,2000,2015}$	$<$ 1m	$>$ 2000m	Jan 2015 - Dec 2015	140,800
$\mathcal{P}_{21,2013-2015,user}$	$<$ 1m, same user	$<$ 1m	Jan 2013 - Dec 2015	57600
$\mathcal{P}_{21,2013-2015,tl}$	$<$ 1m, timelapse	$<$ 1m	Jan 2013 - Dec 2015	54400
$\mathcal{P}_{10,2013-2015,user}$	$<$ 10m, same user	$<$ 10m	Jan 2013 - Dec 2015	118,400
$\mathcal{P}_{10,2013-2015,tl}$	$<$ 10m, timelapse	$<$ 10m	Jan 2013 - Dec 2015	110,400
$\mathcal{P}_{30,2013-2015,user,2h}$	$<$ 30m, same user within 2 hours	$<$ 30m	Jan 2013 - Dec 2015	198,400
$\mathcal{P}_{MiddleburyLR,diff\_user}$	Left, right pairs	Different users	NA	32000
$\mathcal{P}_{MiddleburyLR,same\_user}$	Left, right pairs	Same users	NA	32000
$\mathcal{P}_{MiddleburyLREL,diff\_user}$	Left, right, exposure, lighting pairs	Different users	NA	32000
$\mathcal{P}_{MiddleburyLREL,same\_user}$	Left, right, exposure, lighting pairs	Same users	NA	32000
$\mathcal{P}_{21,2000,01\_2014}$	$<$ 1m	$>$ 2000m	Jan 2014	32000
$\mathcal{P}_{21,2000,02\_2014}$	$<$ 1m	$>$ 2000m	Feb 2014	32000
$\mathcal{P}_{21,2000,03\_2014}$	$<$ 1m	$>$ 2000m	Mar 2014	32000
$\mathcal{P}_{21,2000,04\_2014}$	$<$ 1m	$>$ 2000m	Apr 2014	32000
$\mathcal{P}_{21,2000,05\_62014}$	$<$ 1m	$>$ 2000m	May 2014	32000
$\mathcal{P}_{21,2000,06\_2014}$	$<$ 1m	$>$ 2000m	Jun 2014	32000
$\mathcal{P}_{21,2000,07\_2014}$	$<$ 1m	$>$ 2000m	Jul 2014	32000
$\mathcal{P}_{21,2000,08\_2014}$	$<$ 1m	$>$ 2000m	Aug 2014	32000
$\mathcal{P}_{21,2000,09\_2014}$	$<$ 1m	$>$ 2000m	Sep 2014	32000
$\mathcal{P}_{21,2000,10\_2014}$	$<$ 1m	$>$ 2000m	Oct 2014	32000
$\mathcal{P}_{21,2000,11\_2014}$	$<$ 1m	$>$ 2000m	Nov 2014	32000
$\mathcal{P}_{21,2000,12\_2014}$	$<$ 1m	$>$ 2000m	Dec 2014	32000
$\mathcal{P}_{21,2000,01\_2015}$	$<$ 1m	$>$ 2000m	Jan 2015	32000
$\mathcal{P}_{21,2000,02\_2015}$	$<$ 1m	$>$ 2000m	Feb 2015	32000
$\mathcal{P}_{21,2000,03\_2015}$	$<$ 1m	$>$ 2000m	Mar 2015	32000
$\mathcal{P}_{21,2000,04\_2015}$	$<$ 1m	$>$ 2000m	Apr 2015	32000
$\mathcal{P}_{21,2000,05\_2015}$	$<$ 1m	$>$ 2000m	May 2015	32000
$\mathcal{P}_{21,2000,06\_2015}$	$<$ 1m	$>$ 2000m	Jun 2015	32000
$\mathcal{P}_{21,2000,07\_2015}$	$<$ 1m	$>$ 2000m	Jul 2015	32000
$\mathcal{P}_{21,2000,08\_2015}$	$<$ 1m	$>$ 2000m	Aug 2015	32000
$\mathcal{P}_{21,2000,09\_2015}$	$<$ 1m	$>$ 2000m	Sep 2015	32000
$\mathcal{P}_{21,2000,10\_2015}$	$<$ 1m	$>$ 2000m	Oct 2015	32000
$\mathcal{P}_{21,2000,11\_2015}$	$<$ 1m	$>$ 2000m	Nov 2015	32000
$\mathcal{P}_{21,2000,12\_2015}$	$<$ 1m	$>$ 2000m	Dec 2015	32000

Table 3.1: Datasets



(a) Left image (b) Right image

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

Figure 3.5: Images for motorcycle

Dataset	**Positive Heuristic**	**Negative Heuristic**	**Size**
$\mathcal{P}_{MiddleburyLR,diff\_user}$	Left, right pairs	Different users	32000
$\mathcal{P}_{MiddleburyLR,same\_user}$	Left, right pairs	Same users	32000
$\mathcal{P}_{MiddleburyLREL,diff\_user}$	Left, right, exposure, lighting pairs	Different users	32000
$\mathcal{P}_{MiddleburyLREL,same\_user}$	Left, right, exposure, lighting pairs	Same users	32000

Table 3.2: Middlebury data

3.3 Google Image Data

A manually labeled dataset is still needed for testing. We use a dataset published by Wang et al., which will be referred to as the Wang set.[64]. 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.

Table 3.3 displays a random sampling of triplets. In contrast to the Flickr data, for which a non-trivial proportion of triplets (p, p^+, p^-), will have all three images be 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, there are some examples we feel may be mislabeled or should not have been included in the dataset. In Table 3.4, we show a few examples demonstrating the relatively narrow margin between similar and dissimilar pairs, and in Table 3.6 a few examples demonstrating how image variants appear to have played a role in the labeling of the Wang set.

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 3.4, 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 3.5.

Table 3.6 shows that human rankers quite often rely on image variants in their judgements of similarity. For "New York City", zoom variance is important; for the "Sydney Opera House" and "Michelangelo", illumination and out-of-plane rotation variance comes into play; and for "Picasso", the rankers seem to have picked up on some similarity in hue.

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

Table 3.3: Random triplets from Wang Set

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

Table 3.4: Hard triplets from Wang Set

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

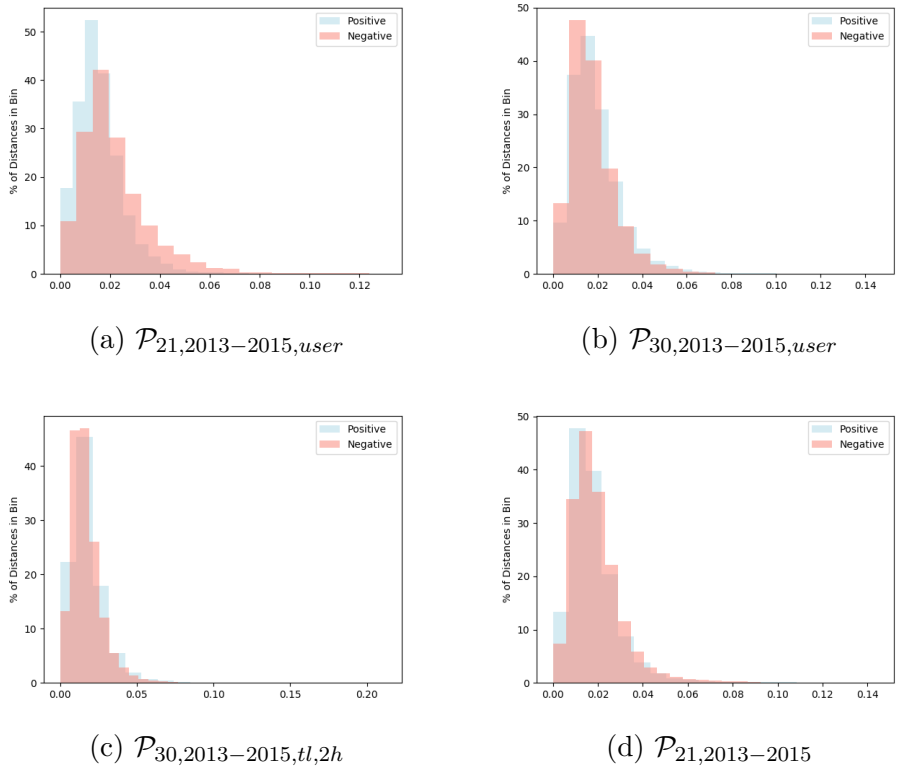
Table 3.5: An easy triplet from the Wang Set

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

Table 3.6: Variant influenced triplets from Wang Set

3.3.1 Quantifying the Amount of Fuzziness

One way of quantifying the fuzziness of the sampled Flickr dataset labels is by extracting image embeddings for each image using a model trained on ImageNet andcompute the mean and median distances for positive pairs and negative pairs. Another useful statistic is the percentage of triplets with ranking violations, where the positive distance is greater than the negative one. Figure 3.3.1 shows that the distribution of distances for positive and negatives pairs is highly similar, with very little separation between means of the two distributions. In fact, $\mathcal{P}_{20,2013-2015,user}$, has an inversion rate of 52%, mean that dissimilar pairs are actually closer on average in the ImageNet embedding space. The average inversion rate for all datasets is 36.6%, which is much higher than the rate of out-of-class errors of 16% in Krause et al.'s study.[29] A full table of distances and inversion is listed in Appendix 6. Not including the time lapse or Middlebury datasets, the three lowest rates of inversion are 0.1034, 0.2098 and 0.3022, and the three highest are 0.5257, 0.4481 and 0.4032.



(a) $\mathcal{P}_{21,2013-2015,user}$ (b) $\mathcal{P}_{30,2013-2015,user}$

(c) $\mathcal{P}_{30,2013-2015,tl,2h}$ (d) $\mathcal{P}_{21,2013-2015}$

Figure 3.6: Positive and negative pair distances

3.4 Differences with ImageNet

ImageNet is a dataset built on top of WordNet, which is a graph of words and phrases with similar semantic concepts connected to each other. There are 100,000 synonym sets, which are strongly connected local clusters, in WordNet. ImageNet aims to provide 1000 example

images for each synonym set, with all images quality-controlled and human annotated. Notably, ImageNet currently only provides examples for nouns, meaning each image has an object as its central focus. With at least 1000 example images per object, it is impossible for a deep CNN trained on ImageNet not to learn certain types of image invariants.

In contrast, the Flickr set contains no class segmentations, so it is unlikely for the same types of generalized image invariants to be learned. A upside-down image, which might force an ImageNet model to learn rotational invariance, will not be tied to a label in the Flickr set. The vast diversity of the Flickr images also contributes to its effectiveness in learning the importance of variants that ImageNet might not prioritize. It is likely that if human annotators labeled all the Flickr photos from New York, there would be at least a magnitude more classes than in ImageNet. Figure 3.4 some randomly selected Flickr images which display the diversity of image space the data spans, as well as a lack of quality-control, meaning many images will not be focused on a particular object or scene to the degree ImageNet images are.

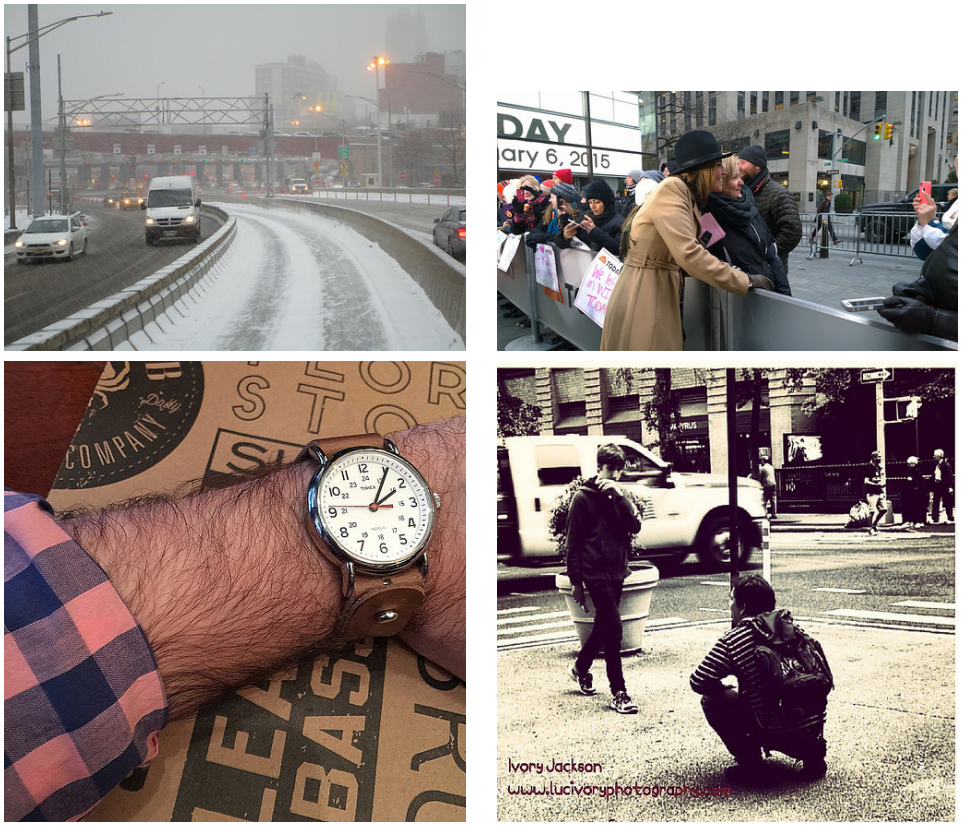


Figure 3.7: Random Flickr images

Chapter 4

Network Architecture and Training

4.1 Model Design

We build off of the general Siamese network formulation proposed by Bromley et al. and used with deep CNNs by Hadsell et al. and Lin et al.[10][22][36] The typical Siamese configuration is modified by the addition of a second pair of CNNs. The networks which comprise the two pairs will be referred to as **I** and **V**. We intend to use **I** to extract image invariant features that are typically learned by deep networks and **V** to preserve other variant features in the final embedding. To this end, **I** is not trained, but is instead loaded with pre-trained ImageNet weights. **V** is trained using the sampled datasets described in Chapter 3. Because of the data sampling method, it is expected that the level and types of invariances learnable by **V** will be quite different from those encapsulated in ImageNet module. Nevertheless, the ability of **V** to learn the same invariants as **I** is handicapped by making it much shallower. This modified Siamese architecture is therefore both multi-module and multi-scaled, in terms of the module depths. During prediction, **I** and **V** take copies of the same image as input and their output feature vectors are concatenated and used as input for a third module, a shallow CNN referred to as **B**. **B** is trained to learn a proper weighting for blending the invariant and variant features produced by **I** and **V**. The blended output produced by **B** will be the model's final image embedding.

First, we discuss our architecture choices for $\mathbf{I}, \mathbf{V}$, and **B** and then we discuss the training process for learning weights for **V** and **B**

4.1.1 Architecture of I and V

We do very little exploration of different network configurations, opting to use ResNet architecture with light modifications as our base for both **I** and **V**. The ResNet architectures use residual layers to optimize the training of extremely deep networks., 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.[23] The residual blocks used in ResNet contain skip layers, which allow image residuals to skip past weight layers. The general architecture of ResNet can be broken down into identity and convolutional blocks.

Identity blocks are composed of three 2D convolutional layers, with each convolutional layer followed by a batch normalization layer, which mitigates shifting distributions of network weights by normalizing means and variances of layer inputs, and a ReLU activation layer. Before output from the last batch normalization layer is inputted to the ReLU activation layer, it is joined by a skip connection that connects the input to the first convolutional layer to this last activation layer, thus providing a way of shallowing the network if learning desires to strongly activate that path. The output from the ReLU activation is then batch normalized.

Convolutional blocks are the same as identity blocks, except the skip connection also passes through one 2D convolutional layer.

A 50 layer ResNet model is used for **I**. This contains one initial 2D convolutional layer, ten identity blocks, four convolutional blocks, and two max pooling layers, one at the beginning of the network and one at the end. For **V**, in order to avoid the invariances that Vidaldi notes inevitably build up by a third or fourth convolutional block, two of the four convolutional blocks and all the identity blocks in between them are removed, thus reducing the network to a relatively shallow architecture containing one initial 2D convolutional layer, two convolutional blocks, five identity blocks, and two max pooling layers.

Beyond this, the ResNet architecture in modified in the same way for both **I** and **V** by changing the standard weight initialization to use the He Normal initialization and the activation function from ReLU to a parametric ReLU. The softmax layer for the output of **I** is removed, and the last pooling layer is flattened to create an output layer of size 2048. The final output layer of **V** is flattened as well, and a fully connected layer of size 1024 is appended to serve as its output. The choice of 1024 is somewhat arbitrary but not without reason. It makes sense to keep the output dimension around the same magnitude as the outputs for ImageNet and Places365, 1000 and 365 respectively, and 1024 is a convenient factor of 2048.

He Normal weight initialization

The He Normal weight initialization 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 it is desirable for the variance of a layer l_{n+1} to be equivalent to the variance of the output of the previous layer l_n so that a weights neither shrink to 0 nor explode as a series of inputs is passed through it.[20] 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 $Var(W_i) = \frac{2}{n_{in}}$.[24]

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.[25]

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

The Flickr datasets are fairly large and very noisy, so mitigating the possibility of "dying" ReLU activations by using parametric ReLU activations is sensible. 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. The Flickr sets contain a wide variety of images, so the possibility of a large percentage of our networy dying is larger than it would be were ImageNet used as the training set. It is almost assured that at least one Flickr user has uploaded an all-black photo, perhaps taken inside a pocket, or an all-white photo taken directly at the sun, and these types of extreme image examples are likely to kill more ReLU units than normal. While parametric ReLU should help guard against this, norm clipping is also applied to the model's weight gradients to ensure they can be updated with a maximum value of one.

4.1.2 Loss Function

Two different loss functions are used. One loss function, Equation 4.1.3, follows the contrastive loss proposed by Hadsell et al.[22], which assigns a high loss to pairs whose embeddings are far apart and a low loss to pairs whose embeddings are close together. The contrastive loss function takes three parameters: l, a binary variable which indicates whether a pair is similar or dissimilar, and a base image p_1 and a query image p_2. g is a gap parameter which is set to be 1. The contrastive loss is used for training $\mathbf{V}$. Wang et al.[64] extend this loss function to a triplet loss, given in Equation 4.1.4, which takes triplets of images. Again, g is a gap parameter. When a positive pair distance is smaller than a

negative pair distance by at least g, the loss function takes the value of 0. When the positive pair distance is larger, however, the loss becomes as large as the distance difference, plus the value of g. The Wang loss is only used when training $\mathbf{B}$ because it requires passing in a triplet of images. For the $\mathbf{V}$ training phase, it is computationally prohibitive to extend our pair sampling heuristics to be able to sample valid triplets so training on pairs with contrastive loss suffices.

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

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

L2 regularization is added for both of these loss functions. The constrastive loss function becomes

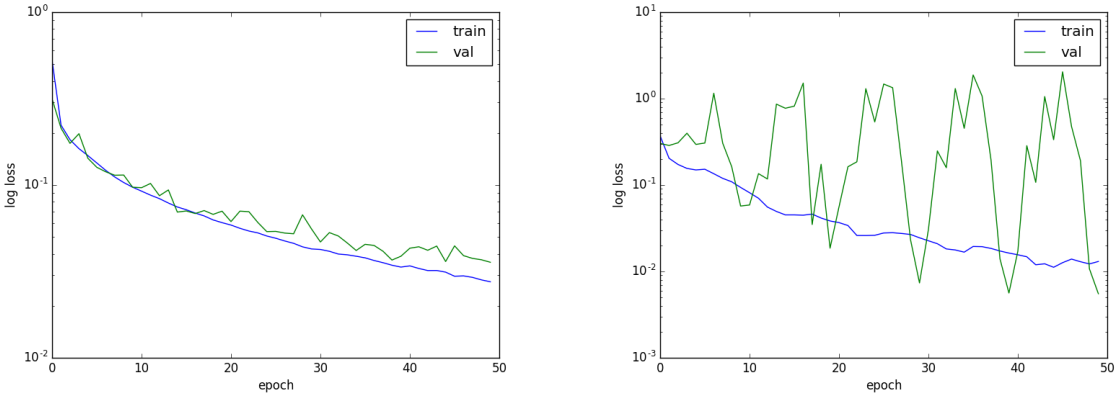
$$\frac{\lambda}{2}||\mathbf{W}||_2^2 \max\{0, g + ||f(p_i) - f(p_i^+)||_2^2 - ||f(p_i) - f(p_i^-)||_2^2\} \tag{4.1.5}$$

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

4.1.3 Optimizer

Several different optimizers are considered: stochastic gradient descent (SGD), RMSprop[62], Adagrad [16], Adadelta[71], Adam[27], and Nadam[27]. Before beginning training, a small experiment on a subset of data is run to select an optimizer based on validation loss. While Nadam, which is Adam with Nesterov momentum, proved to be the most volatile, often causing exploding gradient updates, it also proved the best at finding successively deeper local minima. 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 $\epsilon = 10^{-8}$.

The results for the majority of our datasets is similar to Figure 4.1 (a), with an extremely smooth log-loss curve. As can be seen, loss drops off very quickly in the initial epochs. The initial gradients of Nadam were much steeper in practice than any of the other optimizers. For $\mathcal{P}_{10,2013-2015,user}$, which had a greater than 50% inversion rate, the loss space is much rougher. Still, Nadam is able to successively escape local minima, as seen by the periodic dropoffs in training loss, and bring validation loss back in line with training loss. Since models are checkpointed at the end of each epoch where validation loss decreased, this is almost equivalent to optimal behavior.



(a) $\mathcal{P}_{1,2000,01\_2014}$ (b) $\mathcal{P}_{10,2013-2015,user}$

Figure 4.1: Nonsmooth training

4.1.4 Architecture of B

A shallow CNN should suffice for blending the variant and invariant outputs. CNNs with one to three fully connected layers and varying layer widths are tested. Layer widths are allowed to vary between 128 and 3072 neurons. From experimental results, the parameters for the number of layers and the width of layers appear to have a negligible effect on the model, so a two layer configuration with widths of 1024 and 256 is chosen. Dropout layers with a dropout rate of 0.3 are included between the fully connected layers in order to improve generalizability. Again, layers are initialized with He Normal weight initialization and pReLU is used to activate the fully connected layers.

4.2 Training Pipeline

4.2.1 Preprocessing

All images are downsampled to size (224,224), the size used by ResNet architectures.[23] Images are not grayscaled nor are any other color manipulations performed besides rescaling to [0,1] because the time of day a photo is taken is likely to have a significant effect on its representation in an image embedding. Photos already in gray scale are removed using an entropy metric. A Laplacian filter is applied to each image, and if the Shannon entropy of the result is below a threshold, the image is pruned. Mean pixel subtraction is not applied since it is relatively redundant with the use of batch normalization layers.

4.2.2 Data Augmentation

Data augmentation is applied after preprocessing for some, but not all, experiments. Images are randomly flipped, both horizontally and vertically, rotated up to 20 degrees, and cropped on zooms of up to 5%. We do not apply any augmentation to color channels.

4.2.3 Training

Only $\mathbf{V}$ and $\mathbf{B}$ are trained. Since the purpose of $\mathbf{I}$ is to extract image invariant features, $\mathbf{I}$ is loaded with weights from a pre-trained ImageNet model. The weights of $\mathbf{I}$ are kept frozen throughout the training of $\mathbf{V}$ and $\mathbf{B}$.

Training V

$\mathbf{V}$ is trained using two methods, neither of which involve $\mathbf{B}$, on our sampled Flickr pairs. For the first method, we run pairs of images through our Siamese network, feeding base images p to $\mathbf{I_p}$ and $\mathbf{V_p}$. We concatenate their output vectors into an embedding E_p. The query image, which is either similar or dissimilar to p, is fed through the other half of the Siamese network, through $\mathbf{I_q}$ and $\mathbf{V_q}$. Their output vectors are concatenated into the embedding E_q. The Euclidean distance of $||E_p - E_q||$ is passed to our contrastive loss layer, which takes a fuzzy binary label indicating the similarity/dissimilarity of the pair (p, q), 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 illustrated in Figure 4.2.

In the second method, we feed input pairs only to $\mathbf{V}$ and not to $\mathbf{I}$. The outputs of the pair of $\mathbf{V}$ networks are passed to the Euclidean distance layer, which passes a distance to the loss layer, which updates the weights of $\mathbf{V}$.

Training B

Regardless of how $\mathbf{V}$ is trained, the training of $\mathbf{B}$ remains the same. $\mathbf{B}$ is trained using triplets from the Wang set. This training requires a triplet Siamese formulation, where there are three instances of $\mathbf{I}, \mathbf{V}$, and $\mathbf{B}$, as shown in Figure 4.4. Triplet images are inputted to both $\mathbf{I}$ and $\mathbf{V}$. $\mathbf{B}$ takes as input the concatenated outputs of $\mathbf{V}$ and $\mathbf{I}$. The Euclidean distances between the outputs of $\mathbf{B_p}$ and $\mathbf{B_p^+}$ and the outputs of $\mathbf{B_p}$ and $\mathbf{B_p^-}$ are passed to the triplet loss layer, which tries to enforce that the positive pair distance should be less than the negative pair distance. The computed gradients are used to update the weights of $\mathbf{B}$ only; the weights of $\mathbf{V}$ and $\mathbf{I}$ are frozen. Since training occurs on the Wang set which only has 5033 triplets, 10-fold cross validation is used rather than a permanently held-out validation set. Training does not occur on Flickr datasets because those datasets are intended to contain less information that would cause the learning of image invariants.

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

Table 4.1: Training statistics

T_b refers to the training of the blending network. Some basic statistics about the three training regimes are summarized in Table 4.1.

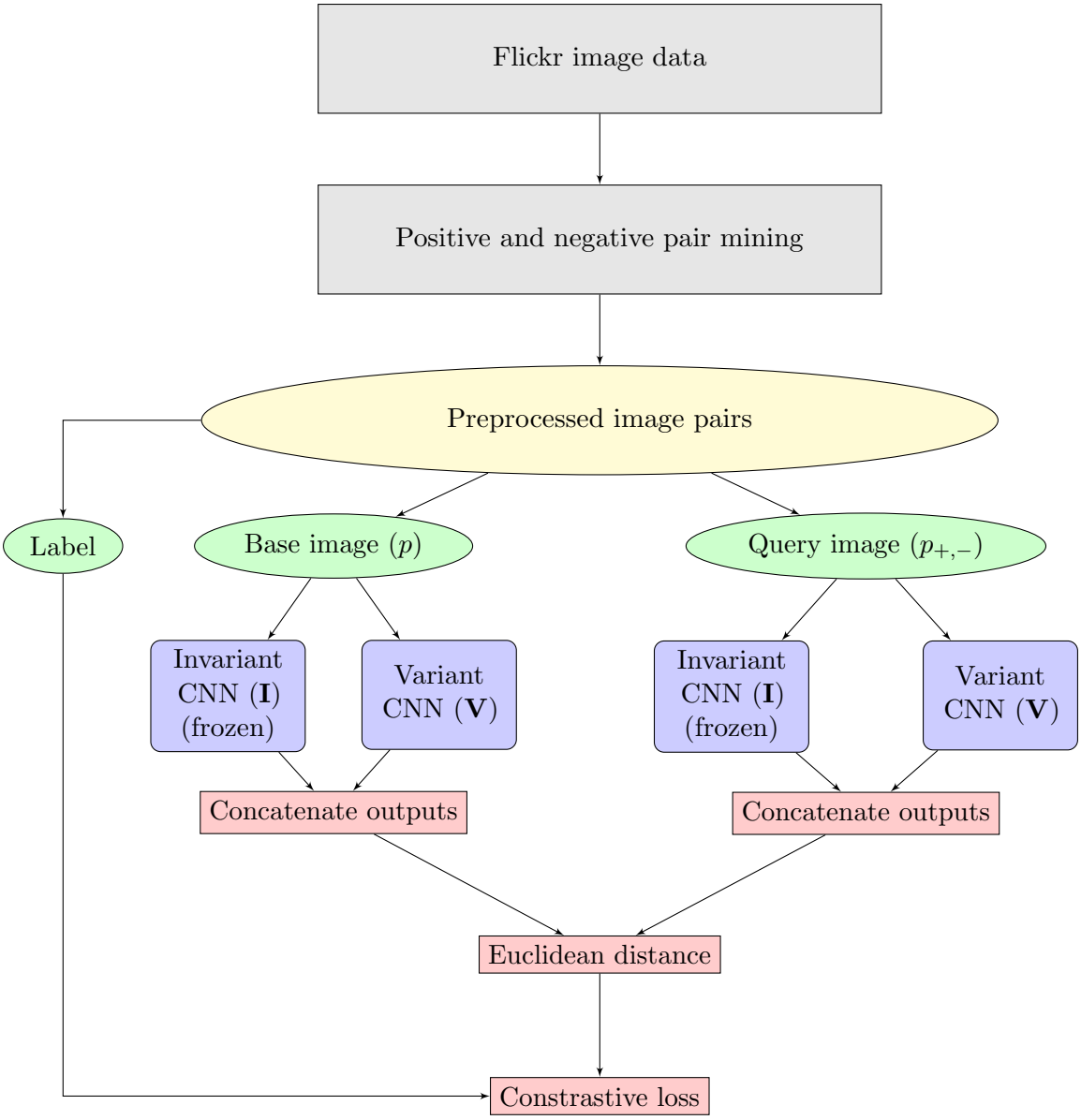


Figure 4.2: Method 1 for training the variant network, **V**

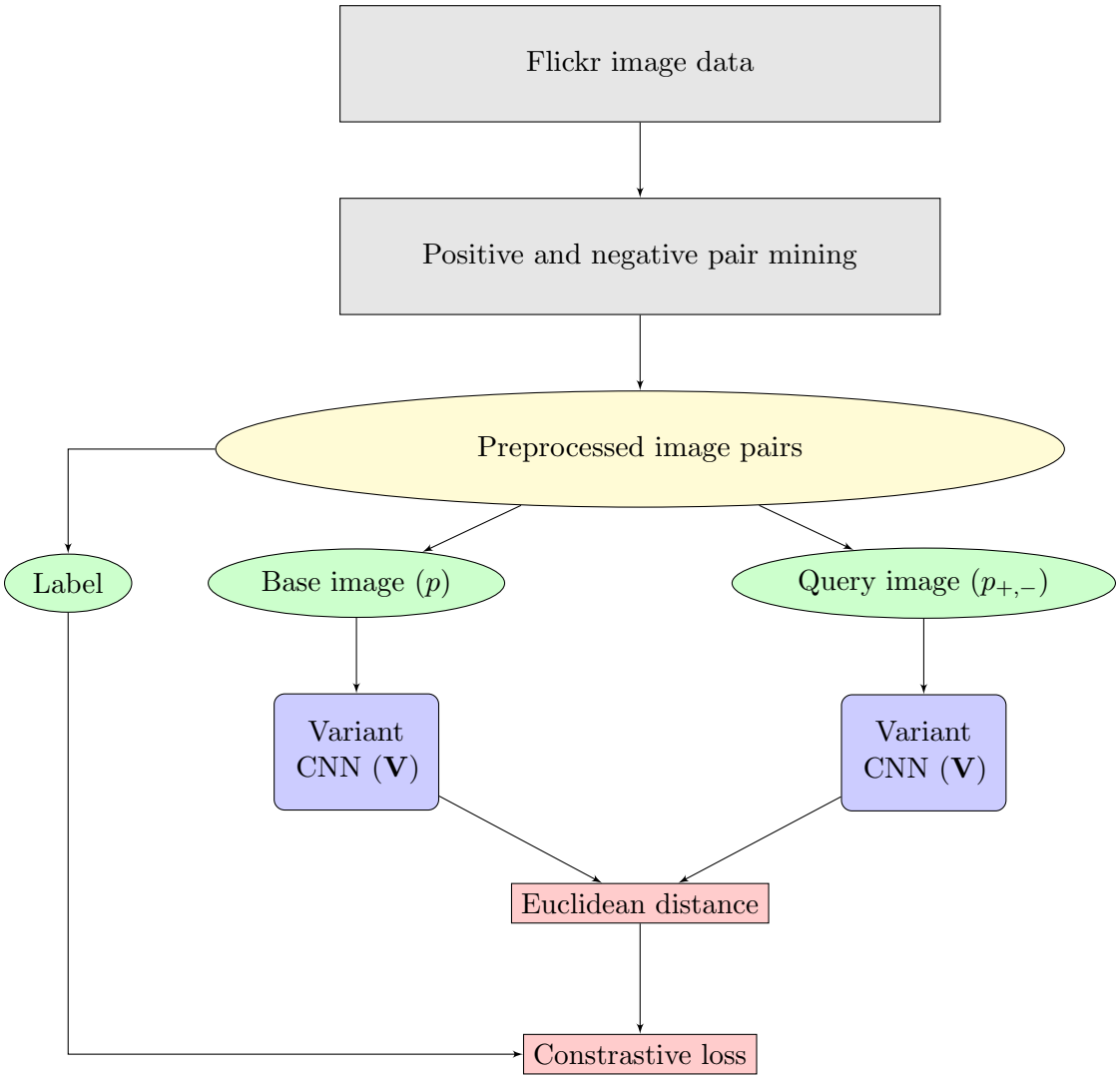


Figure 4.3: Method 2 for training the variant network, $\mathbf{V}$

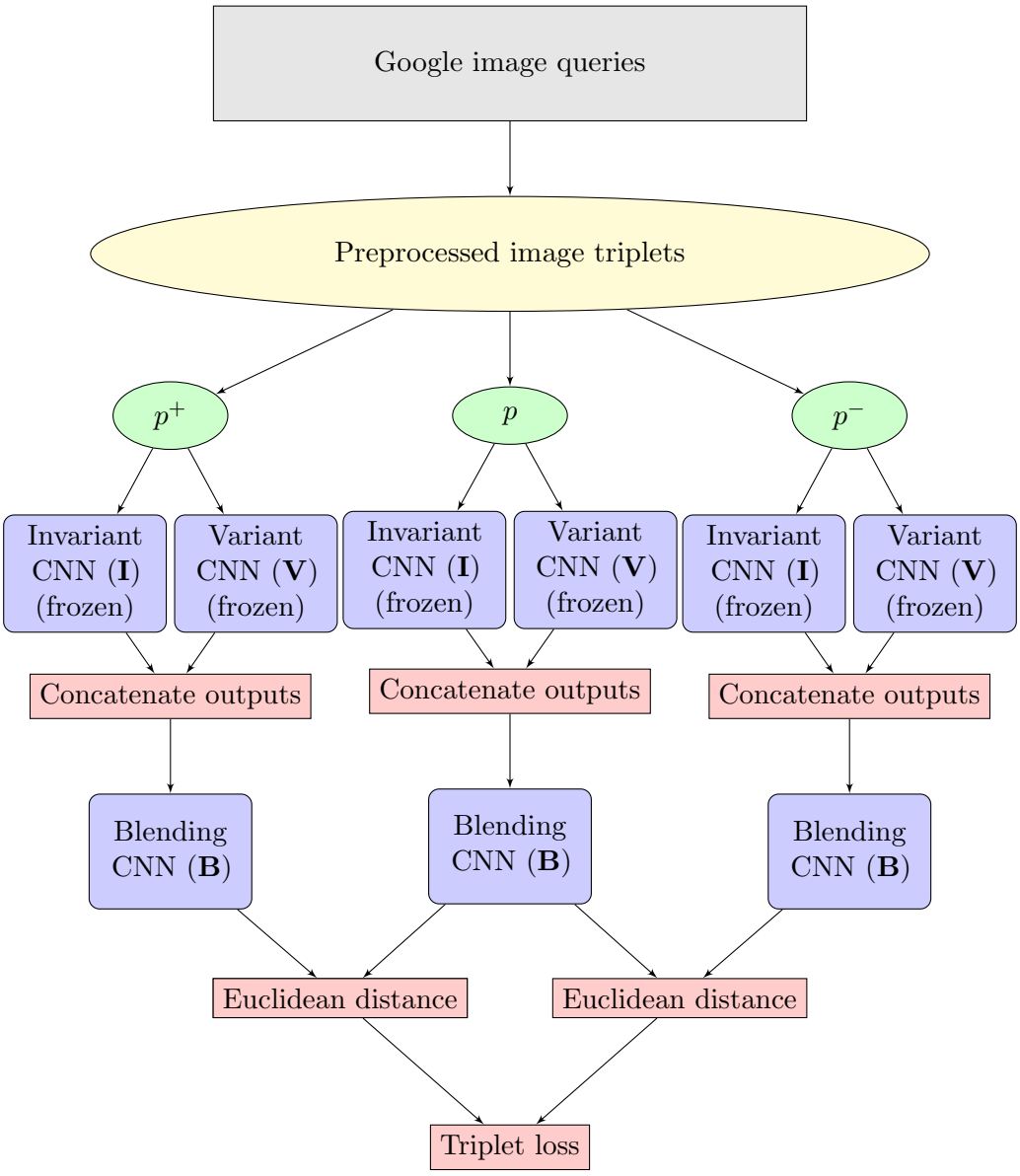


Figure 4.4: Pipeline for training the blending network, **B**

Chapter 5

Experiments and Discussion

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, if possible, quantify the effects of data fuzziness on model learning rates and generalizability
3. Investigate the importance of invariants

We will first discuss our training of **V**. Then we evaluate the image embeddings produced by our full blended model and explore the embedding power of **V**. Lastly, we discuss what our Middlebury and time lapse datasets imply about the importance of image invariants.

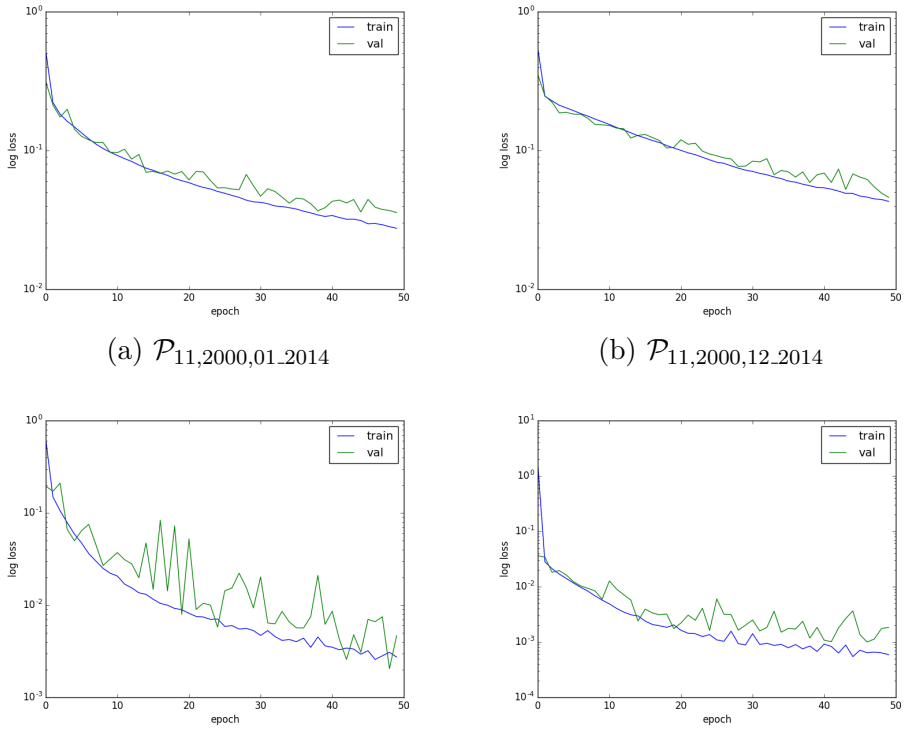
Network training was done on either a Tesla K20m GPU with 5GB of memory or on a Tesla P100 GPU with 16GB of memory. Due to memory constraints, all models that run on the K20m GPU are trained with batch sizes of 6. Models trained on the P100 GPU are trained with batch sizes of 32. All training is single GPU. Generally, because the Flickr datasets contained at minimum 32000 pairs and often around 100,000 pairs, the training of **V** took approximately 2-4 days for 50 epochs on the Tesla P100 GPU. The training of **B**, which done using 10-fold validation over the Wang set, required less than a day on the Tesla P100 GPU. Training done on the K20m GPU was roughly four times slower.

5.1 Weakly Supervised Image Embedding

5.1.1 Training Module V

Despite the extreme fuzziness of the labels as shown in Table 1, that the choices of architecture, optimizer, and loss function prove to be able to quickly locate local minima. Validation

loss plummets without delay, often reaching inflection points in the loss curve within three training epochs, and learning typically has enough momentum to escape local minima and find deeper minima. Representative training and validation loss curves are displayed in Figure 5.1. Due to constraints of time and shared GPU resources, training was limited to 50 epochs. For most datasets, this resulted in a training time of approximately 3 days. As seen in Figure 5.1, additional training time would likely lower validation loss by a non-trivial amount, as the loss curves do not reach a fully flattened state, nor is there a conclusive split between training and validation loss.



(a) $\mathcal{P}_{11,2000,01\_2014}$ (b) $\mathcal{P}_{11,2000,12\_2014}$

(c) $\mathcal{P}_{11,2013-2015,user}$ (d) $\mathcal{P}_{11,2000,2015}$

Figure 5.1: Log loss plots for training and validation

5.1.2 Training Module B

In constructing **B**, it is assumed that any blending function should be relatively simple, so **B** is limited to be at maximum three layers deep. To determine the proper widths of these layers, L_1 and L_2, a rough grid search is performed. Our grid search finds no meaningful differences between different layer widths, so the widths of L_1 and L_2 are arbitrarily chosen to be 1024 and 256 with a dropout layer with dropout rate of 0.3 between them. Because of the limited size of the Wang dataset used for training, experiments with ad-hoc manual architectures are conducted as well, such as including a residual layer for either the variant output or invariant output, or for both. None of these manual formulas showed any promise, so the model was reverted back to its simple two layer state for the computation of ranking accuracy.

5.1.3 Ranking Accuracy of Full Blended Model

For reporting the accuracy of our full model, it would be normal to select the training method and dataset for $\mathbf{V}$ with the lowest validation loss and compute its blended test accuracy on the Wang set. But differences in validation loss are really due to the different datasets use for training, rather than as a product of different network architectures, so these validation losses are not useful for directly comparing model effectivenesses. Instead, each trained model is used in an ensemble prediction. Note that though many datasets have been mentioned, some, like the Middlebury datasets, were curated to explore the effect of image invariants, while others were intended for exploring the effect of an increase in the number of data samples on training. The final ensemble consists of the models trained on $\mathcal{P}_{11,2013-2015,user}, \mathcal{P}_{11,2000,2015}, \mathcal{P}_{11,2000,2013-2015}, \mathcal{P}_{20,2013-2015,user}$, and the monthly datasets from 2014, $\mathcal{P}_{11,2000,01\_2014}$ through $\mathcal{P}_{11,2000,12\_2014}$.

As a reminder, a triplet $t = (p, p^+, p^-)$ is considered to be properly ranked by an embedding function if and only if $f(\dot{)}$ if $||f(p) - f(p^+)|| < ||f(p) - f(p^-)||$.

Final prediction begins by taking the $\mathbf{V}$ networks trained on each of the datasets in the ensemble for each of the two training methods. $\mathbf{V}, \mathbf{I}$, and $\mathbf{B}$ are triplicated, and Wang set input triplets are passed into for feature extraction. 10-fold cross validation is used. $\mathbf{B}$ begins with uninitialized weights and is trained on nine folds of the data before saving its predicted embeddings for the last fold. The ranking accuracy for this prediction fold is *not* computed at this time–only the embeddings for each image in the validation fold are computed and saved. After saving predictions for the entire dataset for each of our different models, an overall ranking accuracy for triplets is computed using the average predicted embedding for each image. We achieve a final ranking accuracy of 64.20%.

For comparison, embeddings produced by an ImageNet-trained model can serve as a baseline classifier. The baseline ranking accuracy is 60.8%, which is a fairly impressive result given the granularity required by a large portion of the Wang set in upholding correct triplet rankings. This is another testament to the well-proven generalizability of features learned by ImageNet trained models. Our model surpasses this by a small but non-trivial margin. This is a substantial result given the context of our training methods. The distribution and variety of images present in the Flickr dataset is far different than the Google image queries which comprise the Wang set. Image queries are almost always object focused, and we had chosen the Flickr datasets based on ideas of maximizing stylistic similarities in photos. Wang et al. had achieved a ranking accuracy of 85%, but this result is not applicable for comparison to our discussed technique because Wang et al. trained a model based on thousands of examples for each of the triplets in their test set.

One worry we have in interpreting our results is wondering if it is the invariant module $\mathbf{I}$ that is actually responsible for the successfully ranked embeddings. This can be proven to not be the case, however, when one considers that we have surpassed an ImageNet-only classifier, meaning that our variant module, in combination with the blending, contributes at least some amount to the successful embeddings. We attempt to quantify the division between the invariant and variant modules by computing the ranking accuracies of embeddings produced by $\mathbf{V}$ only. These are shown in Table 5.1. The average ranking accuracy is 57.33%, which

Dataset	Epochs	Samples	Validation Loss	Ranking Accuracy
$\mathcal{P}_{11,2000,2013-2015}$	50	176,000	0.0034	0.5692
$\mathcal{P}_{11,2000,2015}$	50	140,800	0.0010	0.5682
$\mathcal{P}_{11,2013-2015,user}$	50	57600	0.3467	0.5889
$\mathcal{P}_{11,2013-2015,tl}$	50	54400	0.3037	0.5782
$\mathcal{P}_{20,2013-2015,user}$	50	118,400	0.3327	0.5585
$\mathcal{P}_{20,2013-2015,tl}$	50	110,400	0.0878	0.5736
$\mathcal{P}_{30,2013-2015,user,2h}$	50	198,400	0.0099	0.5973
$\mathcal{P}_{11,2000,01\_2014}$	50	32000	–	0.5625
$\mathcal{P}_{11,2000,02\_2014}$	50	32000	–	0.5629
$\mathcal{P}_{11,2000,03\_2014}$	50	32000	–	0.5830
$\mathcal{P}_{11,2000,04\_2014}$	50	32000	–	0.5686
$\mathcal{P}_{11,2000,05\_62014}$	50	32000	–	0.5710
$\mathcal{P}_{11,2000,06\_2014}$	50	32000	–	0.5758
$\mathcal{P}_{11,2000,07\_2014}$	50	32000	–	0.5694
$\mathcal{P}_{11,2000,08\_2014}$	50	32000	–	0.5810
$\mathcal{P}_{11,2000,09\_2014}$	50	32000	–	0.5730
$\mathcal{P}_{11,2000,10\_2014}$	50	32000	–	0.5651
$\mathcal{P}_{11,2000,11\_2014}$	50	32000	–	0.5788
$\mathcal{P}_{11,2000,12\_2014}$	50	32000	–	0.5673

Table 5.1: Validation losses

is comfortably above a random ranking and quite close to the ImageNet baseline. This is perhaps a more impressive result than the overall accuracy, since at no point in its training does **V** see a single example from the Wang set. Taking into consideration the distance between the average Wang image and the average Flickr image implies that this result is also highly generalizable, so long as some transfer learning, such as occurred with the training of **B** directly on the Wang set, occurs. **V** has learned an image embedding function that is nearly as good as ImageNet's and with only a fraction of the cumulative training power. ImageNet is, of course, trained on over ten million strongly supervised images, and **V** is trained on at most 140,800 weakly labeled examples for 50 epochs.

To confirm that **V** has learned a useful image embedding function that is not merely a subset of **I**/ImageNet, we compare the overlap in their ranking predictions.

As shown in Figure 5.2, on average, our models correctly rank 61.3% of the same triplets that the baseline classifier does. In total, we rank 57.5% of triplets in the same fashion, counting jointly incorrectly ranked triplets as well. The relatively low overlap between our models, and the fact that this overlaps remains the same range without exception, indicates that the learnable embedding from our dataset differs to a significant degree from the one learnable from ImageNet. Given that the baseline classifier correctly ranks 60% of triplets, our correct ranking overlaps indicates that roughly 37% of triplets are correctly ranked by both our models and the baseline. An additional 20-25% are ranked correctly by either the baseline model or one of our models, but not by both.

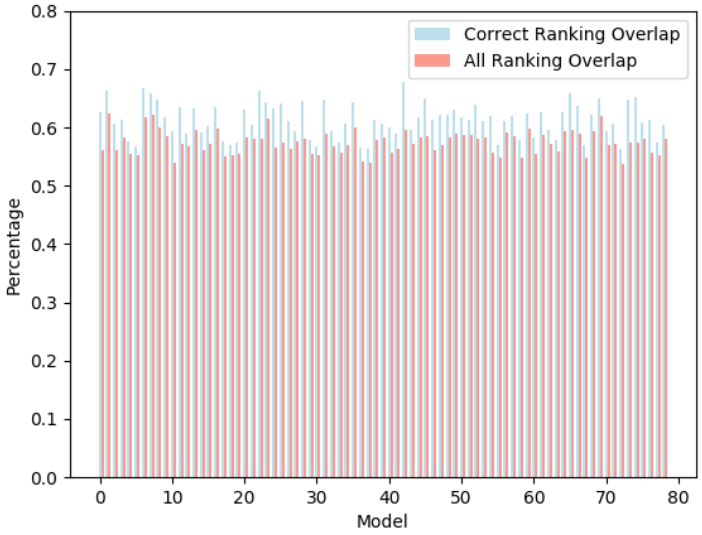


Figure 5.2: Prediction overlaps with baseline

5.1.4 Comparison of Training Methods for V

Recall that we train **V** using two methods, T_c^1, where the concatenated outputs from **I** and **V** are used to generate the contrastive loss and update **V** only, and T_c^2, where only the output from **V** is sent to the loss function.

In our discussion, let the intermediate embedding vector used as input to **B** be referred to as E_c and let the final embedding vector outputted by **B** be referred to as E_b. In the case where we use the training T_c^1, E_c is the concatenation of the invariant embedding, E_i, and the variant embedding, E_v. When discussing embedding distances among pairs or triplets, embedding with a superscript E^+ or E^- will refer to the positive or negative example, and embeddings without superscripts will refer to the base image.

We initially expected embeddings E_b produced by models trained using T_c^1 to outperform models trained using T_c^2. Either images will be similar because they contain the same type of soft style, in which case both T_c^1 and T_c^2 will pick up on this, or they will be similar despite differences in soft style because the physical composition of the images will be similar in a way approximately learnable by an ImageNet model. In the latter case, only T_c^1 trained embeddings will capture this. Because of the way we sample images from our Flickr set, which is entirely without a notion of class, we do not expect to learn any notion of invariance. The ImageNet training process, as discussed, strongly enforces the learning of invariance. In other words, we expect T_c^1 to outperform T_c^2 because the set of features it trains a model to detect should approximate a superset of the features T_c^2 will train a model to detect. We also think T_c^1 will force **V** to learn a set of features farther from ImageNet learnable features. Since the output of **I** is sent to the loss function, when training on two images with ImageNet-invariant features, the overall Euclidean distance of the outputted vectors to be smaller on average and thus not activate a large update. Then **V** is more likely to receive

large gradients when **I** is unable to detect any common salient features.

In actuality, as seen in Table 5.1, it appears that T_c^1 struggles to optimize on the loss space of our datasets. In theory, the addition of features should never decrease a model's effectiveness, since any weighing of the smaller set of features will be a subset of the possible weightings of the larger set, with optimization solutions being equivalent when the extra features are given weights of 0. Reaching this solution, however, might take a prohibitively long time. Perhaps more importantly, there is potential for large errors in the magnitude of our weight updates to the variant model. Should the invariant portions of our output vector be dramatically different, this will cause the overall Euclidean distance of our prediction to be large as well. But since we are only updating the variant model, we will change its weights in an unwarranted manner because of this large difference. This would be the opposite of what we had intuited would happen, with the training of **V** being driven by large gradients received in the absence of **I** detecting useful features.

Despite our intuition, it appears that the training of **I** and **V** is best left separate. These empirical results support the idea that a third network is needed to blend their outputs.

5.2 Importance of Low-Level Descriptors

Lastly, we investigated the importance of image invariants through our time lapse and Middlebury datasets. The Middlebury dataset contains just 31 unique objects, and with only four images per object, it not suited to teaching a network a wide variety of image invariants. Likewise, the time lapse dataset identifies only images of the same location, with images mainly differing in the temporal domain, as similar. Though lighting conditions change, since there is only one "class" of image in this dataset, any invariants learned are not general image invariants, such as rotational invariance for salient objects, but are unique to the structure of the time lapse base image.

Given that these datasets are so close to the limit case of training a network with identical images labeled as similar and random images labeled as different–which we would not expect to learn any meaningful information–it is surprising to see in Table 5.2 that ranking accuracies for models trained on these datasets seems comfortably above 50% random guessing. The positive examples in these datasets do contain very similar distributions and locations of low-level local descriptors. While this area requires further research, it is likely that the majority of the discriminative power learned by these models can be attributed to the importance of these descriptors. This would mean that the 60% baseline accuracy achieved by ImageNet models likely owes a substantial portion of its performance to these descriptors as well.

Dataset	**Epochs**	**Samples**	**Validation Loss**	**Ranking Accuracy**
$\mathcal{P}_{11,2013-2015,tl}$	50	54400	0.3037	0.5782
$\mathcal{P}_{20,2013-2015,tl}$	50	110,400	0.0878	0.5736
$\mathcal{P}_{MiddleburyLR,diff\_user}$	50	32000	0.0000	0.5535
$\mathcal{P}_{MiddleburyLR,same\_user}$	50	32000	0.0878	0.5611
$\mathcal{P}_{MiddleburyLREL,diff\_user}$	50	32000	0.0897	0.5851
$\mathcal{P}_{MiddleburyLREL,same\_user}$	50	32000	0.0039	0.5390

Table 5.2: Validation losses

Chapter 6

Conclusion

6.1 Conclusion

This paper presents a deep Siamese network designed and trained for the purpose of preserving the extraction of both invariant and variant features from images. Most importantly, we find that this preservation allows us to learn an image embedding function which appears to describe a space mostly separate from the embedding space learned by ImageNet-trained models. Multiple steps are taken to increase the likelihood of this preservation. Our multi-module Siamese network separates the weights used to extract variant and invariant features. We use weakly supervised learning on class-less data, and this likely prevents the learning naturally occurring image invariants. We empirically show that the use of a shallow CNN can blend together our invariant and variant features in a way that outperforms ImageNet on an image similarity task, and this contributes to the evidence that alternatives to strongly supervised learning are not only viable, but also effective.

6.2 Future Work

The most important future work would include verifying and exploring to a greater degree the type and quality of features learned by $\mathbf{V}$. This is possibly best done by examining predictions for curated pairs of images that share certain stylistic or semantic characteristics.

While the predictive power of $\mathbf{V}$ has enough generalizability to extend to the Wang set, the Wang set was still constructed for use as an image similarity task. It would be interesting to see if $\mathbf{V}$ can be used to transfer learning to other machine vision tasks, or how well it would be able to learn for an ImageNet classification task.

Lastly, this project exploited only the tiniest fraction of the data availble in the Flickr set, and this Flickr set itself only contains images geotagged from New York City. It is possible that truly large scale training, for longer than 50 epochs, and with better sampling heuristics

could improve the empirical results in this study as well as open the doors to even more weakly supervised applications.

A: Data

Dataset	Mean Positive Distance	Median Positive Distance	Mean Negative Distance	Median Negative Distance	Ranking Violations
$\mathcal{P}_{1,2000,2013-2015}$	0.0180	0.0156	0.0193	0.0170	0.4482
$\mathcal{P}_{1,2000,2015}$	0.0079	0.0077	0.0196	0.0168	0.1034
$\mathcal{P}_{1,2013-2015,user}$	0.0159	0.0144	0.0219	0.0186	0.3310
$\mathcal{P}_{1,2013-2015,tl}$	0.0075	0.0074	0.0220	0.0193	0.0543
$\mathcal{P}_{10,2013-2015,user}$	0.0193	0.0169	0.0178	0.0157	0.5257
$\mathcal{P}_{10,2013-2015,tl}$	0.008	0.0077	0.0194	0.0165	0.1487
$\mathcal{P}_{30,2013-2015,user,2h}$	0.0179	0.0157	0.0170	0.0150	0.5190
$\mathcal{P}_{MiddleburyLR,diff\_user}$	0.0077	0.0063	0.0197	0.0170	0.1169
$\mathcal{P}_{MiddleburyLR,same\_user}$	0.0077	0.0064	0.0178	0.0155	0.1550
$\mathcal{P}_{MiddleburyLREL,diff\_user}$	0.0064	0.0049	0.0202	0.0175	0.0818
$\mathcal{P}_{MiddleburyLREL,same\_user}$	0.0064	0.0049	0.0178	0.0156	0.1139
$\mathcal{P}_{1,2000,01\_2014}$	0.0160	0.0147	0.0197	0.0174	0.3806
$\mathcal{P}_{1,2000,02\_2014}$	0.0145	0.0128	0.0183	0.0156	0.3733
$\mathcal{P}_{1,2000,03\_2014}$	0.0148	0.0132	0.0223	0.0194	0.3022
$\mathcal{P}_{1,2000,04\_2014}$	0.0161	0.0142	0.0205	0.0177	0.3586
$\mathcal{P}_{1,2000,05\_62014}$	0.0152	0.0139	0.0199	0.0178	0.3391
$\mathcal{P}_{1,2000,06\_2014}$	0.0169	0.0149	0.0208	0.0184	0.3672
$\mathcal{P}_{1,2000,07\_2014}$	0.0151	0.0138	0.0199	0.0178	0.3228
$\mathcal{P}_{1,2000,08\_2014}$	0.0190	0.0171	0.0212	0.0188	0.4192
$\mathcal{P}_{1,2000,09\_2014}$	0.0149	0.0135	0.0192	0.0174	0.3303
$\mathcal{P}_{1,2000,10\_2014}$	0.0149	0.0134	0.0192	0.0173	0.3368
$\mathcal{P}_{1,2000,11\_2014}$	0.0171	0.0151	0.0201	0.0176	0.3792
$\mathcal{P}_{1,2000,12\_2014}$	0.0155	0.0137	0.0179	0.0155	0.4005
$\mathcal{P}_{1,2000,01\_2015}$	0.0172	0.0154	0.0200	0.0175	0.4093
$\mathcal{P}_{1,2000,02\_2015}$	0.0156	0.0139	0.0200	0.0174	0.3632
$\mathcal{P}_{1,2000,03\_2015}$	0.0165	0.0148	0.0201	0.0181	0.3590
$\mathcal{P}_{1,2000,04\_2015}$	0.0198	0.0175	0.0219	0.0194	0.4032
$\mathcal{P}_{1,2000,05\_2015}$	0.0180	0.0161	0.0214	0.0192	0.3753
$\mathcal{P}_{1,2000,06\_2015}$	0.0162	0.0153	0.0198	0.0184	0.3417
$\mathcal{P}_{1,2000,07\_2015}$	0.0123	0.0109	0.0177	0.0156	0.3252
$\mathcal{P}_{1,2000,08\_2015}$	0.0110	0.0093	0.0187	0.0162	0.2098
$\mathcal{P}_{1,2000,09\_2015}$	0.0158	0.0143	0.0182	0.0160	0.4152
$\mathcal{P}_{1,2000,10\_2015}$	0.0183	0.0173	0.0202	0.0184	0.4320
$\mathcal{P}_{1,2000,11\_2015}$	0.0150	0.0128	0.0191	0.0167	0.3443
$\mathcal{P}_{1,2000,12\_2015}$	0.0154	0.0142	0.0185	0.0165	0.3897

Table 1: Dataset Fuzziness

Experiments

Bibliography

[1] A. E. Abdel-Hakim and A. A. Farag, "Csift: A sift descriptor with color invariant characteristics," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 1978–1983.

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] M. Bober, "Mpeg-7 visual shape descriptors," *IEEE Transactions on circuits and systems for video technology*, vol. 11, no. 6, pp. 716–719, 2001.

[8] G. Bouchard and B. Triggs, "Hierarchical part-based visual object categorization," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 710–715.

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

[10] 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.

[11] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 539–546.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[17] J. Eakins, M. Graham, and C.-b. I. Retrieval, "University of northumbria at newcastle," *Content-based Image Retrieval*, 1999.

[18] 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.

[19] 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.

[20] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *Aistats*, vol. 9, 2010, pp. 249–256.

[21] K. Grauman and T. Darrell, "Efficient image matching with distributions of local invariant features," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 627–634.

[22] 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.

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

[24] ——, "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.

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

[26] 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.

[27] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

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

[29] 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.

[30] 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.

[31] 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.

[32] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8595–8598.

[33] 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.

[34] K. Lenc and A. Vedaldi, "Understanding image representations by measuring their equivariance and equivalence," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 991–999.

[35] 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.

[36] 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.

[37] 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.

[38] S. K. Lodha and Y. Xiao, "Gsift: Geometric scale invariant feature transform for data registration," in *SPIE Conference on Vision Geometry XIV*, 2005.

[39] 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.

[40] B. S. Manjunath, J.-R. Ohm, V. V. Vasudevan, and A. Yamada, "Color and texture descriptors," *IEEE Transactions on circuits and systems for video technology*, vol. 11, no. 6, pp. 703–715, 2001.

[41] R. Mehrotra, K. R. Namuduri, and N. Ranganathan, "Gabor filter-based edge detection," *Pattern recognition*, vol. 25, no. 12, pp. 1479–1494, 1992.

[42] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[43] J.-R. Ohm, L. Cieplinski, H. J. Kim, S. Krishnamachari, B. Manjunath, D. S. Messing, and A. Yamada, "The mpeg-7 color descriptors."

[44] 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.

[45] R. Porter and N. Canagarajah, "Robust rotation-invariant texture classification: wavelet, gabor filter and gmrf based schemes," *IEE Proceedings-Vision, Image and Signal Processing*, vol. 144, no. 3, pp. 180–188, 1997.

[46] 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.

[47] 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.

[48] Y. M. Ro, M. Kim, H. K. Kang, B. Manjunath, and J. Kim, "Mpeg-7 homogeneous texture descriptor," *ETRI journal*, vol. 23, no. 2, pp. 41–51, 2001.

[49] B. E. Rogowitz, T. Frese, J. R. Smith, C. A. Bouman, and E. B. Kalin, "Perceptual image similarity experiments," in *Photonics West'98 Electronic Imaging*. International Society for Optics and Photonics, 1998, pp. 576–590.

[50] 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.

[51] 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.

[52] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[53] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International journal of computer vision*, vol. 105, no. 3, pp. 222–245, 2013.

[54] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, "High-resolution stereo datasets with subpixel-accurate ground truth," in *German Conference on Pattern Recognition*. Springer, 2014, pp. 31–42.

[55] 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.

[56] 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.

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

[58] 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.

[59] 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.

[60] 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.

[61] H. Tamura, S. Mori, and T. Yamawaki, "Textural features corresponding to visual perception," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 8, no. 6, pp. 460–473, 1978.

[62] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, 2012.

[63] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," in *Proceedings of the 18th ACM international conference on Multimedia*. ACM, 2010, pp. 1469–1472.

[64] 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.

[65] T. P. Weldon, W. E. Higgins, and D. F. Dunn, "Efficient gabor filter design for texture segmentation," *Pattern recognition*, vol. 29, no. 12, pp. 2005–2015, 1996.

[66] C. S. Won, D. K. Park, and S.-J. Park, "Efficient use of mpeg-7 edge histogram descriptor," *ETRI journal*, vol. 24, no. 1, pp. 23–30, 2002.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[72] 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.