INDIANA UNIVERSITY BLOOMINGTON

Convolutional Neural Networks for Infrared, Fine-Grained, and Egocentric Scene Classification

by

Sumit Gupta

A thesis submitted to the faculty of the
School of Informatics and Computing
in partial fulfillment of the requirements for the degree of
Master of Science

June 2016

Accepted by the faculty of the School of Informatics and Computing, in partial fulfillment of the requirements for the degree of Master of Science.

Thesis Committee

Dr. David J. Crandall

Dr. Michael S. Ryoo

Dr. Saúl A. Blanco Rodríguez

June 2016

Abstract

In recent years, Convolutional Neural Networks (CNNs) have seen significant rise in popularity because of their remarkable performance in image classification, object recognition, and object localization. They are being applied on a wide variety of problems in computer vision. However, they are still quite new, and we do not have a clear understanding how well they perform on less traditional computer vision tasks, for example, object recognition in infrared images. We use CNNs to generate models on different datasets with varying level of difficulty, and record their performance. Despite being relatively new and achieving state-of-the-art results in various domains of computer vision, we try to answer the extent up to which CNNs perform well on less traditional tasks in the domains of infrared spectrum, fine-grained recognition, and egocentric vision. We manually collect infrared images using Seek Thermal Infrared camera mounted on an Android phone for human recognition, use NOAA Right Whale dataset for fine-grained recognition of Right Whales, and manually collect egocentric dataset for scene classification. We find that their performance on these non-traditional domains is mixed, with state-of-the-art performance on fine-grained recognition and egocentric vision, but relatively poor performance on thermal imagery. We also discuss potential factors that affect CNN performance in different domains and on these less traditional tasks.

Contents

List of Figures

List of Tables

Abbreviations

CNN	Convolutional Neural Networks
SVM	Support Vector Machines
IR	Infrared
HOG	Histogram of Oriented Gradients
NOAA	National Oceanic and Atmospheric Administration

Chapter 1

Introduction

A huge number of images are captured from cameras daily. A large fraction of them are captured for recreational purposes using mobile phones and handheld digital cameras. On many of those images, intelligent systems extract various types of information in real-time. For example, cameras detect faces so that they can adjust focus appropriately [7], phones detect smiles to click a photograph automatically [8], or an image captured from a wearable camera can describe the scene [9]. To recognize the type of images and objects within those images, classification is required to put images and objects into different categories. Image classification is at the core of computer vision with many important applications. For example, Google photos [10] has a search capability for automatically recognizing objects and returning relevant results.

On the backend, there are several image classification techniques used to achieve this. For performing classification, image features are required to train a classifier. Features are a representation of an image in a specific form that help in solving a computational task. Image features are analogous to machine learning features. However, image features have to contend with a significant variation in scale, size, rotation, etc. For example, local features like SIFT [11] are based on small regions of an image and do not represent an image as a whole, while edge features [12] and color histograms [13] are more global.

Image features are very general and require human knowledge to design and extract such information from images.

Similar to learning an image classifier, one can learn features as well, as in the case Convolutional Neural Networks (CNNs) [14, 15]. Over the last few years, CNNs have taken over a large part of research for designing and extracting good features from images [16–19]. They learn features automatically without human intervention and achieve state-of-the-art results in many tasks in the computer vision domain including image classification. Most of the remarkable results in CNNs are achieved for deliberately composed images in visible spectrum and coarse grain classification tasks. For example, a huge amount of research has studied visible spectrum image datasets like ImageNet [20], CIFAR-10 [21], MNIST [22], etc. Even though CNNs were known to work well for certain problems like handwritten digit recognition [23] since the 1990s, their usage declined because of the increasing popularity of SVM [24] until recently when they regained attention [16].

However, CNNs are still quite new, and we do not have a complete understanding of how well they perform on less traditional tasks. For instance, the infrared spectrum is an important domain where instead of capturing visible light, the image is composed by temperature maps of objects. Currently, we do not have a clear understanding of how well CNNs would perform for infrared images. John et al. [25] apply CNNs to recognize humans on infrared images. However, their application and dataset is limited to humans with standing pose. Another important domain is fine-grained recognition where the task is to identify sub-ordinate categories like types of aircraft [26] or species of birds [27] or dogs [28]. Finally, work on CNNs has largely focused on deliberately composed images. What if the images are not deliberately composed, such as egocentric images from a wearable camera [29]? In the following subsections, we explain these different less-traditional computer vision problems and their importance.

1.1 Infrared Imagery

Most research in the computer vision community has focused on visible spectrum images and videos. Recently, development of low cost infrared cameras has made it easy to explore infrared images using mobile phones, especially in case of night vision [30, 31]. Infrared spectrum images are quite different from visible spectrum. In visible spectrum images, the temperature of the objects does not contribute anything to the image, and visual features are based on actual visual elements that humans see. On the other hand, infrared spectrum images are temperature maps, where visual elements correspond to temperature, and edges corresponds to changes in temperature between different objects. Figure 1.1 shows an image of the same scene captured in the visible and infrared spectra. These IR images introduce new opportunities and challenges for computer vision systems. For example, recognizing humans in a visible spectrum image might be difficult because of clutter and occlusions. However, humans are clearly exposed in infrared spectrum because of their body temperature relative to surrounding environment (assuming they are not standing in front of other hot objects). This makes evaluation of CNNs for both spectra worth exploring.

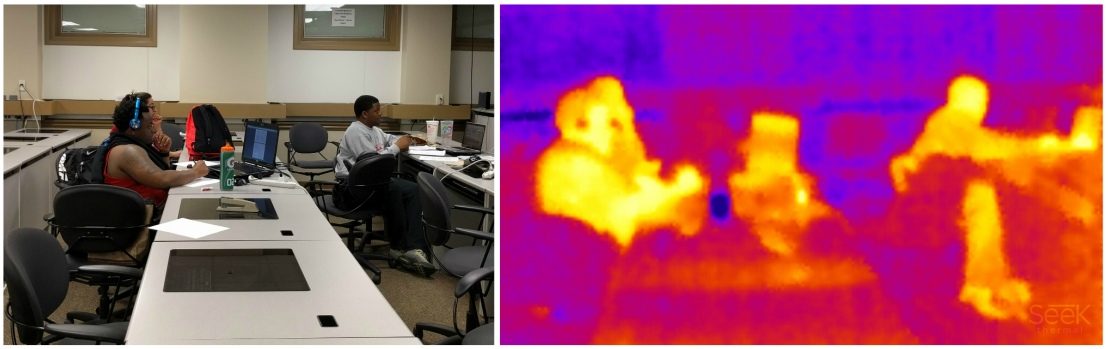


FIGURE 1.1: Same scene in visible and infrared spectrum.

The visible spectrum, as the name suggests, is the range of electromagnetic spectrum with wavelengths from 390nm to 700nm, which is visible to human eye [32]. The infrared spectrum has a longer wavelength range than the visible spectrum of light, and extends

from 700nm to 1mm [33]. The spectrum of light is shown in Figure 1.2.

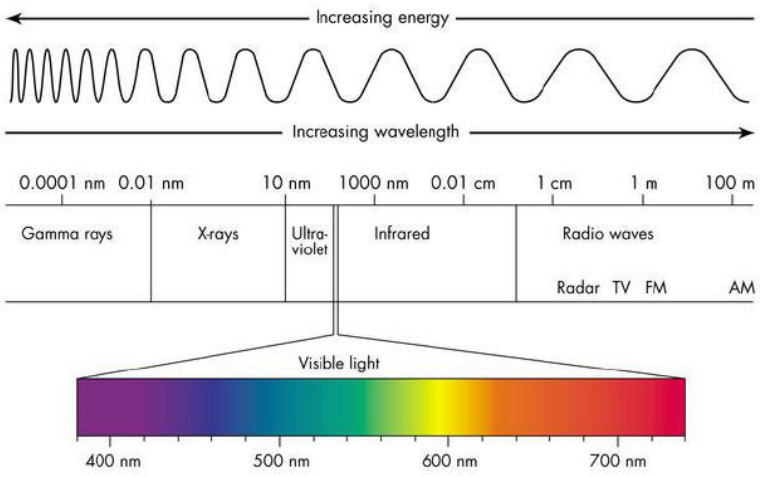


FIGURE 1.2: Spectrum of Light. Source [1]

Objects at room temperature emit thermal radiation having wavelength in the infrared spectrum range. With the development of small infrared imaging cameras, it is possible to capture thermal images easily. The concept of infrared cameras is based on capturing thermal radiation emitted by objects and mapping them to color values in the visible spectrum. Like visible spectrum images, the quality of objects in infrared images can vary significantly depending on location, activities and the surrounding environment. Images can be completely clutter free or extremely noisy, depending on the temperatures of surrounding objects. An infrared image is like performing a segmentation of a visual spectrum image based on temperature. It combines objects of the same temperature and adds many artifacts which can be invisible in visible spectrum (like hot air coming out from a laptop). Infrared can be very useful in detecting objects which are known to maintain their temperature in a certain range, for example living objects like animals. Since they map temperatures instead of visible light, thermal imaging is invariant to visible light, i.e. useful infrared images can be captured during the night time.

Even with the rise of CNNs, to our knowledge, there is no work comparing performance of CNNs on problems in infrared and visual spectra. The infrared spectrum has received

less attention from the research community than the visible spectrum. Most of the research for human detection in infrared images is on pedestrians. Bertozzi et al. [34] propose human detection by localizing warm symmetrical objects having fixed aspect ratio, size, shape and thermal characteristics. It makes assumptions that pedestrians are not occluded by other objects or people. Suard et al. [35] and Zhang et al. [36] apply HOG features on infrared images to detect pedestrians, which is more robust than using aspect ratio, size and shape [34]. Qi et al. [37] apply scattered difference of directional gradients on thermal images which works better than HOG features. However, their datasets of pedestrians contain standing pedestrians in a very limited set of poses. Nanda and David [38] propose a probabilistic template based approach which captures the variety in human poses and occluded body parts and applies it on low quality infrared videos. Recently, Portmann et al. [39] study finding humans in aerial images and use a background subtraction method and particle filter guided detector. However, their dataset only contains images taken from nine outdoor sequences. Davis and Keck [40] also use background subtraction and template classification on a dataset collected from a stationary camera in various scenes. John et al. [25] use CNNs to recognize humans after performing segmentation with an adaptive fuzzy C-means algorithm. As mentioned above, almost all work in the area of infrared images is related to detecting pedestrians with straight-up pose. To our knowledge, there has been no work related to human detection in various poses in infrared images using CNNs.

1.2 Fine-Grained Recognition

Another important problem is fine-grained recognition, which means recognizing sub-categories of objects, for example, car models [41], breeds of dogs [28], airplane models [26], etc. This task is particularly challenging because of the minute details in objects, requirements of domain knowledge, and similarity in appearance of different objects.

For example, distinguishing a dog from a human is less challenging than distinguishing between different breeds of dogs. The problem is important because it deals with recognition at a much lower level scale involving finer details. For example, automatic identification of bird species from photographs can help researchers in conservation efforts, while automatic identification of aircraft and car models can help authorities to maintain records and statistics.

Compared to infrared human recognition, fine-grained recognition and image classification in general has received more attention from the research community. In image classification, Wan et al. [42] and Ciresan et al. [43] have achieved state-of the art results for the MNIST dataset, Graham [44] and Springenberg [45] for CIFAR-10, Snoek [46] for CIFAR-100, and Szegedy [18] for ImageNet. There has been work on variety of fine-grained classification datasets including flowers [47], larva [48], birds [27], dogs [28], etc. Before Krizhevsky et al. [16], researchers mostly used hand crafted techniques to extract features. For example, Csurka et al. [49], Wang et al. [50] and Yao et al. [51] use codebook based approaches like bag of key points, while Branson et al. [52] and Farrell et al. [53] use part based annotations. Yao et al. [54] propose codebook and annotation free recognition by running a random template matching process.

Annotating different parts of an object is challenging, which is often required for good features to identify individual category types having subtle differences. Branson at al. [55] and Zhang et al. [56] showed improvement over handcrafted features by learning them automatically using CNNs [53, 57]. We consider the problem of whale recognition for which, to our knowledge, there is no previous work specifically for whales in fine-grained recognition from aerial photographs using computer vision techniques.

1.3 Egocentric Images

Both problems we discussed above, infrared recognition and fine-grained recognition, involve deliberately captured images. What if the images are not deliberately composed? For example, images taken from wearable, egocentric camera are not deliberately composed [29]. It is an important domain of computer vision in which images are captured from a wearable camera from the perspective of a person or a robot. These days, more and more intelligent systems like wearable cameras and robots are becoming parts of our daily lives. These systems interact with our surroundings using many types of sensors. For example, wearable cameras like Narrative Clip, GoPro and Google Glass are becoming popular [58]. The problem of classification of egocentric images is important because it captures the objects and actions of daily living. An intelligent system like Google Glass can alert a human about potential dangerous situations by analyzing a scene, or can select few images to create a summary of the day. Such systems require image classification at all levels. Because of the popularity of wearable cameras, egocentric images have received attention from the research community. There are many challenges in classifying images captured from an egocentric context. Because of the egomotion, the images from a wearable camera are mostly blurred and poorly composed compared to deliberately composed images.

Most of the work in egocentric images and videos is on activity recognition and contextual awareness. The earliest known works in egocentric and wearable computing are Starner [59, 60], in which authors discuss about the adoption of wearable technology. Initial research to apply computer vision algorithms on egocentric images was focused on hands and faces [61, 62]. Starner et. al. [63] use computer vision techniques for place recognition to play a game. Torralba et. al. [64] recognize places and objects using wavelet image decomposition features and Hidden Markov Models (HMMs). Mayol and Murray [65] recognize hand activities based on skin color segmentation and color

histogram. Kang et. al. [66] proposes an image matching algorithm for scene understanding in indoor environment using bag of words model and TF-IDF. Kameda and Ohta [67] propose an approach to locate pedestrians using SURF. Nakayama et. al. [68] propose algorithm for describing generic objects and multi-labeling and retrieval of unconstrained real-world images. Kitani et. al. [69] propose unsupervised learning of action categories on sports videos based on learning a motion histogram codebook and action categories using stacked Dirichlet process mixture model. Fathi et. al. [70] propose learning object models based on bottom-up segmentation, while Falthi et. al. [71] propose approach to recognize daily activities using semantic relationship between activities, actions and objects. Pirsiavash and Ramanan [72] present recognizing activities of daily life using temporal pyramid and active object models. Ryoo and Matthies [73] analyze interactions in egocentric vision by integrating global and local motion information. All the approaches mentioned above use handcrafted features. Compare to the research in normal image classification tasks, CNNs have gained less attention for egocentric vision. Though, there have been few works applying CNNs to egocentric vision. Bambach et. al. [74] use CNNs to detect hands after probabilistically generating window proposals. Ma et. al. [75] apply CNNs in two-stream fashion to analyze appearance and motion for activity recognition. Castro et. al. [76] use CNNs with Random Decision Forests to recognize activities from 19 classes. Ryoo et. al. [77] propose pooled feature representation and use CNN as appearance feature extractors. Singh et. al. [78] use egocentric cues like hand pose, head motion and saliency maps as network input to CNN for activity recognition.

1.4 Contribution and Summary of Thesis

Our goal in this work is to apply, evaluate, and reason about performance of CNNs on these less traditional tasks. We use CNNs to generate models on different datasets with varying level of difficulty, and record their performance. Despite being relatively new

and achieving state-of-the-art results in various domains of computer vision, the research question to be answered is the extent to which CNNs perform well on less traditional tasks in the domains of infrared spectrum, fine-grained recognition, and egocentric vision. We manually collect data for infrared spectrum and egocentric images, and use the NOAA Right Whale dataset [79] for fine-grained classification.

Chapter 2

Approaches

In this chapter we introduce the approaches we have tried for human recognition, fine-grained classification, and egocentric scene classification. We experiment with different approaches including Color Histograms, Histogram of Oriented Gradients and CNNs.

2.1 Color Histogram

One of the simplest approach to represent features is histogram of pixel intensities, in which individual pixel values are collected into a vector. A color histogram is a distribution of color intensities in an image, i.e. the number of pixels for each color range in the color space. Histograms can be constructed for various color spaces including RGB, HSV, and with different number and sizes of discrete bins. Because of the large number of possible values of the color intensities, we create 256 bins and count the number of pixels into each bin. The histogram layouts the intensities from 0 (pure black) to 255 (pure white). Pixels of the same intensities are stacked vertically to create the histogram. The image intensities for an image are considered as random variables with a probability density function. After computing the histogram values for each color, we normalize it to estimate the probability density function, which outputs the probability

of each intensity occurring at any random pixel. We create a color histogram for every image and store the values into a normalized vector and use them as training data. Figure 2.1 gives sample color histograms of an IR image from our dataset. Since color histograms ignores spatial information in the image, we also explored more advanced features for better performance.

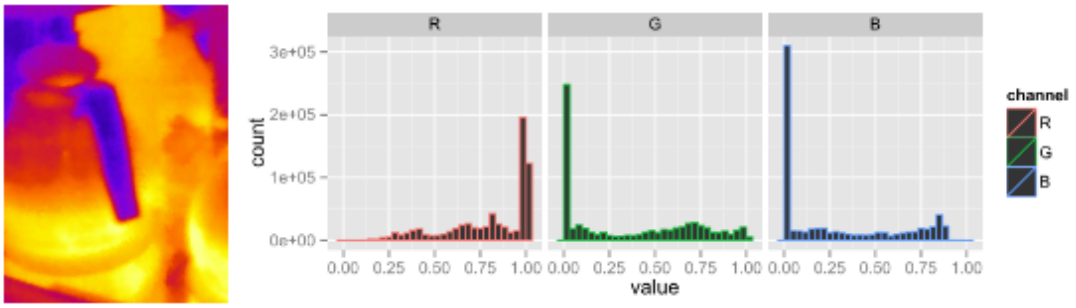


FIGURE 2.1: Color Intensity Histogram for an image in IR dataset.

2.2 Histogram of Oriented Gradients

Histogram of Oriented Gradients (HOG) descriptors [80] are used widely for generating features for image classification and are known to work well for pedestrian detection. They give importance to the gradient orientation in local parts of an image. The idea behind HOG is that local object appearance and shape can be well described by the distribution of local intensity gradients. To compute the HOG descriptor of an image, the first step is to compute the histogram of gradient directions for each pixel in equal sized small contiguous regions called *cells*. Following important steps describe computation of HOG descriptors as shown in Figure 2.2.

Gamma/Color Normalization: this step is generally performed to compensate for large variation in illumination and luminance in the images. Dalal and Triggs [80] pointed out that gamma and color normalization have little effect on the results likely because of the descriptor normalization in subsequent steps.

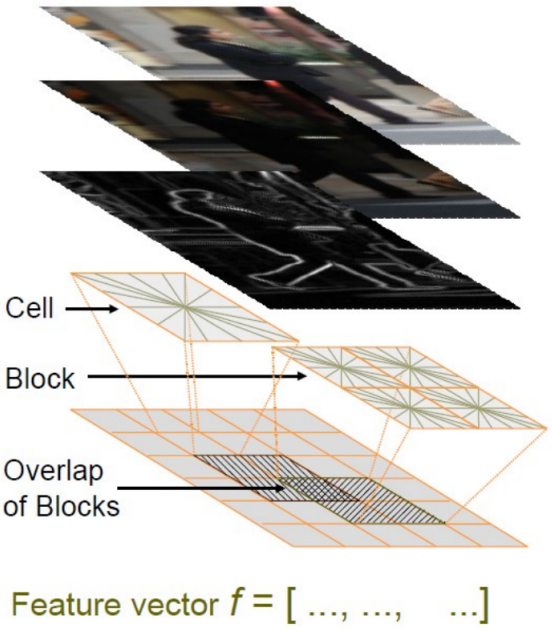


FIGURE 2.2: Steps for HOG computation. Source [2]

Gradient Computation: computation of image gradients is done by applying 1-D gradient filters over horizontal and vertical directions. Instead of using more complex masks like 3x3 Sobel filter, 1-D gradient filters were found to perform better. For color images, the authors compute gradients for each color channel and take the one with largest norm.

Orientation Binning: for calculating the cell histograms, each pixel in a cell casts a weighted vote for the orientation bin that it belongs to. Cells can be rectangular, or radial in shape in which histogram bins are evenly spaced over 0 to 180 degrees (0 to 360 in case of signed gradient). The weight in the vote can be the gradient value itself or a function of gradient.

Descriptor Blocks: For invariance to illumination and contrast, gradients are locally normalized by calculating a measure of intensity values over larger region called *blocks* and using the value to normalize the cells of that block. Blocks are overlapped so that cells can vote multiple times in orientation distribution. Blocks can be of two types (Figure 2.3): rectangular (R-HOG) and circular (C-HOG). R-HOG blocks, which is

more commonly used, are rectangular with optimal sizes of 2x2 cells, with each cell having 8x8 pixels.

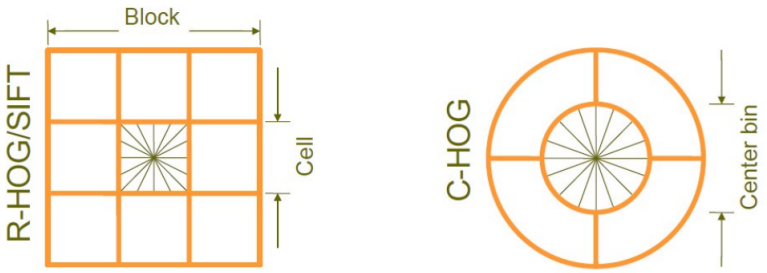


FIGURE 2.3: R-HOG and C-HOG. Source [2]

Block Normalization: There are four different ways to normalize the blocks. Let v be the unnormalized feature vector having all cell histograms of a block and $\|v\|_k$ denotes the k-norm and ϵ be a small constant. The normalization schemes are:

L2 Norm:

$$\hat{v} = \frac{v}{\sqrt{\|v\|_2^2 + \epsilon^2}}$$

L1 Norm:

$$\hat{v} = \frac{v}{\|v\|_1 + \epsilon}$$

L1 Sqrt:

$$\hat{v} = \sqrt{\frac{v}{\|v\|_1 + \epsilon}}$$

L2-Hys is calculated by re-normalizing L2-norm after clipping. The authors observed that L2-Hys, L2-norm, and L1-sqrt all perform equally well, while unnormalized blocks significantly reduces the performance.

After these steps, the HOG descriptor is a vector with normalized cell histogram values of all the cells in the image. HOG has some advantage over other techniques, as it is more invariant towards illumination and shadows. Also since it is computed for local cells, it is more invariant to geometric transformations.

2.3 Convolutional Neural Networks (CNNs)

With the rise of deep learning in recent years, we explain the concepts behind CNNs and Artificial Neural Networks (ANNs). A neural network is a biological network of neuron cells in brains of animals which communicate with each other using electrical signals. When a neuron receives a strong signal through synapses, it activates itself and emits a signal forward to other connected neurons which can further activate those neurons. An Artificial Neural Network is a biologically inspired network which tries to imitate the workings of the human mind by creating a network of artificial neurons. The complexity of biological neurons is simplified in ANNs, which contain a set of inputs with weights (strength of inputs). The inputs with weights determine the activation of the neurons based on an activation function. A single neuron of an ANN is called a Perceptron, as shown in Figure 2.4.

A perceptron is a binary classifier which upon taking many inputs produces only one output with two possible values (say, 0 and 1). It is the fundamental unit of an ANN. To determine whether a perceptron will get activated or not, it computes the dot product of inputs and weights, and fed the result into an activation function. It determines the extent to which there is a potential for firing a perceptron. There are many types of linear and non-linear activation functions, including Sigmoid, Tanh and Rectified Linear Unit (ReLU). The main problem with perceptrons is that they assume linear separability of the input data which makes it difficult to learn many types of functions including an XOR. To solve the issue with a single perceptron, multiple perceptrons can be combined in a feed-forward fashion to learn different functions including XOR.

A feed-forward neural network is an ANN where the connection between perceptrons do not make a cycle [81]. A multilayer perceptron is a feed-forward ANN as shown in Figure 2.5.

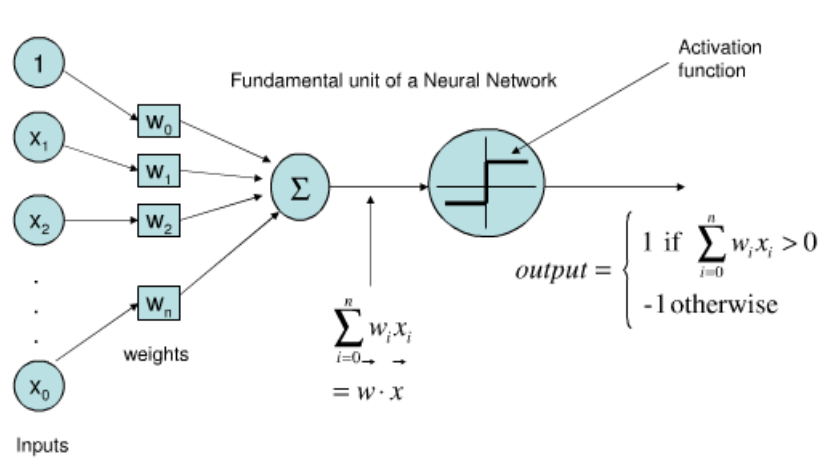


FIGURE 2.4: A perceptron. Source: [3]

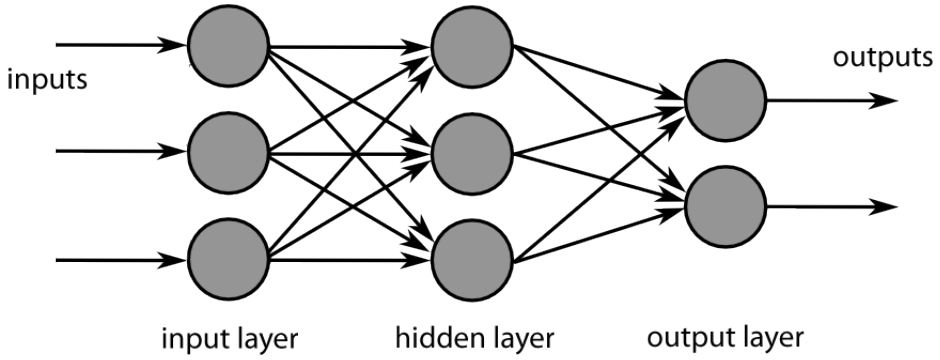


FIGURE 2.5: A multilayer perceptron. Source: [4]

2.3.1 The Back-propagation Algorithm

The Back-propagation Algorithm [82] is generally used to train artificial neural networks. It works in two steps. Step 1 involves *propagation*. Forward propagation of the input from first layer to output layer through hidden layers and backward propagation of errors, which are calculated by the difference between the output values and expected values. Step 2 involves *weight update*. During the backward propagation of errors, each perceptron updates its weights according to its error contribution to the overall error.

2.3.2 CNNs

CNNs are a type of feed-forward artificial neural network in which neurons perform image convolution operation together at a large scale. An example of a CNN is given in

Figure 2.6, in which the network is used to identify handwritten digits [5].

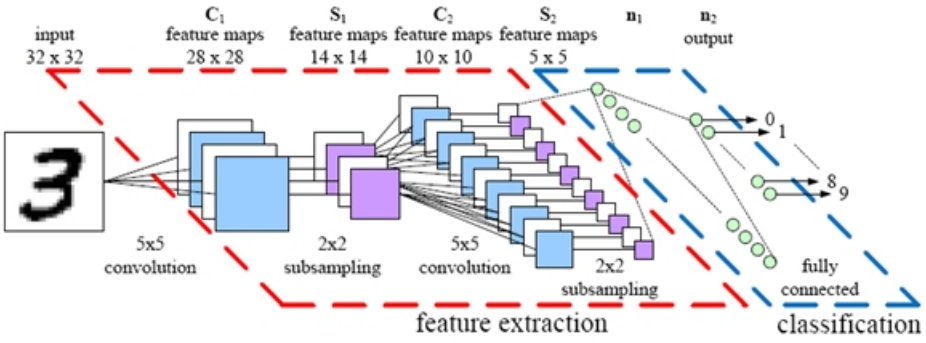


FIGURE 2.6: Convolutional Neural Network. Source: [5]

For classification tasks, we need good features which describe the image category well for a general image. When a CNN is trained using training data, the weights within the network are updated during a backward pass using the back propagation algorithm. Once these weights achieve a satisfactory level of accuracy, they can be re-used to test on other data or to retrain an almost similar network. These weights are the features learned from the training data. Unlike traditional computer vision techniques, CNNs learn the features automatically. A CNN is composed of many layers, some of which are described below:

Input Layer: this is the first data layer which takes the image input. It starts the forward propagation.

Pooling Layer: It subsamples the images based on kernel size and stride parameters which are decided by the network designer. Its function is to reduce the size of image for the next layers to reduce computation and overfitting.

Convolution Layer: the weights of this layer are a set of learnable filters. When an image passes through a convolution layer during a forward pass, each filter convolves with the image thus producing activation maps. Convolution layer learns filters which activate when they receive patterns they are trained for.

Fully Connected Layer: this layer has pair-wise connections to all the activation of the previous layer.

We use the Caffe Deep Learning framework [83] for our CNN tasks. Caffe is a C++/CUDA library for CNN with command line, Python and MATLAB interfaces.

In the following chapters, we apply these techniques to IR, Right Whale, and Egocentric datasets. We apply color histograms and HOG to IR dataset for computing a baseline. Then we experiment with CNNs as a classifier and feature extractors on each of the datasets. We present observations and conclusion in chapter 4.

Chapter 3

Experiments

To perform experiments on the problems mentioned in the previous sections, we used different datasets on different problems. Using the methods mentioned above, we ran experiments on three datasets including one dataset with images in the infrared spectrum (IR Dataset), and two with images in the visible spectrum: NOAA Right Whale Dataset [79] and Egocentric scene dataset [84]. Out of these three datasets, we manually collect infrared and egocentric dataset [84]. We run similar experiments on each dataset as described in the following section and then provide observations in Chapter 4.

3.1 Experiment Protocol

We perform three experiments on each of the datasets: (1) Classification of images based on CNN fine-tuning of pre-trained models, (2) Feature extraction from fully connected layers of pre-trained CNN and classification using SVMs, and (3) Features extraction from fully connected layers of fine-tuned model and classification using linear SVMs. We used the Caffe Deep Learning framework [83] for performing these experiments. Following sections provides details about the experiments.

3.1.1 Classification based on CNN fine-tuning of pre-trained models

CNNs contain millions of parameters. Deep networks containing several layers require a huge number of images to train. In the case we do not have a large training set, we can use pre-trained networks and fine-tune them on our dataset. Fine-tuning a CNN is based on the concept of transfer learning. In Machine Learning, transfer learning [85] is the concept of storing knowledge (a model and its parameters) by solving a problem and exploiting that knowledge to generate useful results on a different but related problem. We exploit transfer learning in CNNs by fine-tuning them to a new problem. While training a CNN, the edge weights between layers are constantly being updated by the back-propagation algorithm [82]. As the CNN sees more images and the number of iterations increases, the weights keeps changing. The weights and network configuration then becomes a model which can be used to test new images. In general, the transferability of CNN models is subjective and depends on a number of scenarios, like new dataset size and similarity with the original dataset.

The model thus generated is trained on a particular set of training images (called a pre-trained model), for example a training set of vehicles (cars, trucks, motorcycle and bikes). All the weights in the network have some value which is good enough for classifying vehicles. We can now remove the last layer of the network and replace it with out own layer with the desired number of outputs. We can then reuse the well initialized weights and dataset-specific features for vehicles to generate a new model by training it with different datasets and continuing the back-propagation. This procedure is called fine-tuning a CNN [86]. In general, fine-tuning means to retrain an already trained network so that it performs better for a new dataset. It borrows the network weights from an already trained network and changes them to suit new data by retraining it.

Adjusting Learning Rates: During back-propagation, the learning rate of a layer determines the rate at which weights of a layer are changed. Since we are fine-tuning a pre-trained network by replacing the last layer, all other previous layers are already well

initialized (but still require fine-tuning). So, it is important to keep the learning rate of the replaced layer high but the remaining layers low, because we do not want to change the previously learned good weights too quickly or too much.

CNNs have proved to be very effective in image classification. For this particular task, we take well known AlexNet [16] based derivatives and fine-tune them on our datasets. Since we already have a pre-trained model on over a million images which works well for image classification, we take Caffenet which is similar to AlexNet except that the order of pooling and normalization layer is switched. Our model consists of an input data layer which resizes each input image to 227x227 pixels and subtracts the mean image file from each input image, followed by 5 convolution layers. Each of the convolution layers is followed by a max pooling layer and normalization layer. We use rectified linear units (ReLUs) as activation units. Convolution layers are followed by three fully connected layers (fc6, fc7 and fc8) and a softmax layer. We also have dropout layers on fc6 and fc7 with a dropout ratio of 0.5. Since the learning has already been done for all layers except fc8, we set the learning rate of fc8 high compared to all other layers.

3.1.2 Feature extraction from pre-trained CNN and classification using SVMs

Features are characteristics of data which are general enough to be fed into a classifier. The task of classification is heavily based on extracting good features. In the case of CNNs, the features are contained in the outputs which attain values after training. Once a CNN is trained, the weights of the network determines several interesting properties. The first layer of the CNN contains high level generic features like edges and blobs [87]. As the network becomes deeper, the deep layers contain features which become more specific to the individual classes in the dataset. A CNN feature is a multi-dimensional vector containing numbers extracted from any of the layers. These numbers in the feature

are the outputs corresponding to that particular layer from which it is extracted. After extracting the features, we can train a classifier based on those feature set.

As described above, CNNs generate features based on the training data. We extract a 4096 dimensional feature vector from the fc7 layer of CaffeNet using the pycaffe interface of Caffe. Since CaffeNet has 1000 classes in fc8, we disregard this output layer and take features from fc7. For the fc7 layer, it always outputs a 4096 dimensional vector irrespective of the number of classes. The data in the Caffe framework flows as blobs. We extract the blobs from the fc7 layer and convert them to human readable format. The features are extracted by inputting the mean subtracted infrared images into the network, 227x227 image size, changing the channel mode to BGR and starting the forward propagation. Since we use pycaffe interface for the forward propagation, we do the initial pre-processing of the image in python as required by the Caffe. We input the image in a batch size of one to merge the extracted feature into a matrix using numpy library [88]. Once the features are extracted, we do L2 normalization of the feature and label matrices, and then use the Linear SVM in the scikit-learn python library to train an SVM. A similar process is repeated for the test set thus generating test features and label vectors, which are normalized and tested using the SVM model.

3.1.3 Features extraction from fine-tuned model and classification using SVMs

We combine both of the above steps: fine-tuning a pre-trained network and training an SVM using features extracted from pre-trained network. We first fine-tune a pre-trained network and create a model which gives the maximum accuracy on the test set. We then extract features from that model and train an SVM to do the classification. Once both of the above approaches are done, it is simple to implement this approach. Instead of the pre-trained model in approach 2, we just replace it with one of the fine-tuned models snapshots.

System Configuration: We train/fine-tune and test the CNN on a Dell PowerEdge T630 server with NVidia Tesla K40 GPU boards and two Intel E5-2680 v3 2.5GHz, 30M Cache, 9.60GT/s QPI, Turbo, HT, 12C/24T, Max Mem 2133MHz and 128 GB memory. For training and testing the SVM, we use a Macbook Pro with Intel i5 2.7 GHz processor, Intel Iris 6000 GPU and 8 GB of memory.

3.2 IR Based Human Recognition

For the human recognition problem, we used a Seek Thermal Imaging camera [30] on an Android phone to capture infrared images. According to [89], the normal human body temperature at rest is around 37° Celsius. We observed over a large number of electronic items that, on average, charging adapters, monitor screens, recently used mobile phones, active printer surfaces, etc. are very near to at-rest human body temperature. We also observed that cooked food, which gradually goes from very hot temperatures to normal room temperature, also comes in nearly the same range after some time. Note that these factors depend on many conditions, like for how long a charging adapter is in continuous use, the air conditioning inside the buildings, etc. We also noted that the at-rest human body temperature of 37° Celsius best matches with the face of a human and there are variations in temperature of different parts of body. For example, hair is much cooler than other body surfaces. Most warm is exposed skin, the face in particular.

3.2.1 Dataset

We manually collected 600 images having people in them in a variety of poses. This set includes images from video sequences (15 frames per second), which were converted to images during pre-processing. We manually removed blurry and duplicate images from the video sequences. We also manually clicked 600 images of general objects without people in them. Our dataset contains a total of 1200 infrared images, each with a

resolution of 624x832 pixels. Sample images of humans are shown in Figure 3.1. For the non-human images, we randomly captured visible hot spots like computer screens, kitchen stove, dining plates, furniture, etc. We did not use video sequences to gather more data for non-human images because there are no moving parts. Sample non-human images are shown in Figure 3.2. We manually labeled each image as to weather it contain humans or non-humans.

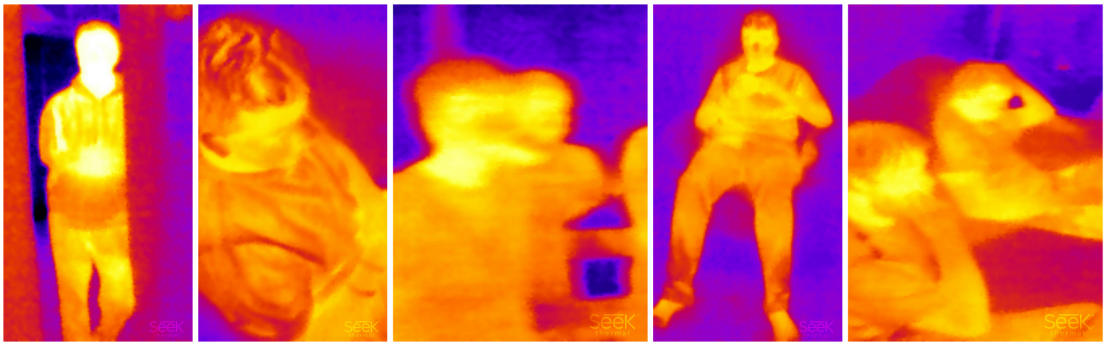


FIGURE 3.1: Infrared images of humans in various poses in our dataset.

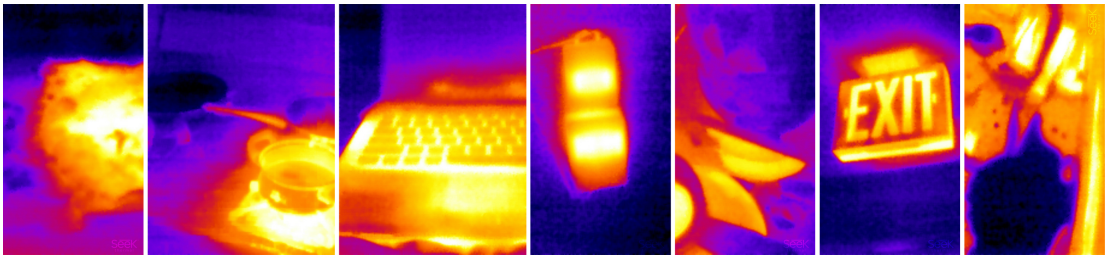


FIGURE 3.2: Infrared images of general objects like food, stove, laptop, etc.

We collected data in a variety of environments and conditions to make it general enough. The dataset was collected in short increments in several different combinations of time of the day. Evenings are much colder than day time, causing a clear difference between non-human surfaces like walls and windows. Electronic items like laptops, bulbs were not affected by the time of day. Turning air conditioning on/off affects non-human surfaces, while opening/closing windows had the same effect as the air conditioning, and started affecting the temperature of surrounding areas after some time. We observed that even with large variations in surrounding environment like day/night, AC on/off, the temperature of an at-rest human body remains in almost the same range of 35°C to 38°C, with an exception of an active human who is cooking, eating, exercising, etc.

Our dataset contains human(s) in various poses. We categorized all images containing humans into six categories as shown in Figure 3.3:

1. Single standing: image covering full or half body of a single person in standing pose, with or without other hot objects, with possible some overlap between the person and objects.

2. Multiple standing: image covering full or half bodies of multiple people in standing poses, with or without other hot objects, with some overlap.

3. Single sitting only: a single person sitting on a chair or couch in variety of poses (folded legs, etc.) with or without other hot objects, with possible overlap and occlusion.

4. Standing and sitting: multiple people, some sitting and some standing, with possible overlap and occlusion.

5. Multiple sitting only: multiple people, all of them sitting on chairs or couches with possible overlap and occlusion.

6. Mixed category: multiple people with a mixed variety of poses, with overlap and occlusion, and from a variety of angles and directions.

Many images in the categories contains other non-human hot objects like laptop screens, mobile phones, food plates, partial body parts of other people, etc. Our dataset contains images in both portrait and landscape mode.

3.2.2 Experiment

We created a text file with a (key, value) mapping for image paths and labels. We perform data augmentation to artificially increase the dataset size by generating horizontally flipped images. Doing this increases the dataset by a factor of two, thus producing a total of 2400 images. We do the augmentation before randomly splitting them into 1600 training and 800 testing images. We then converted the images into LMDB format as

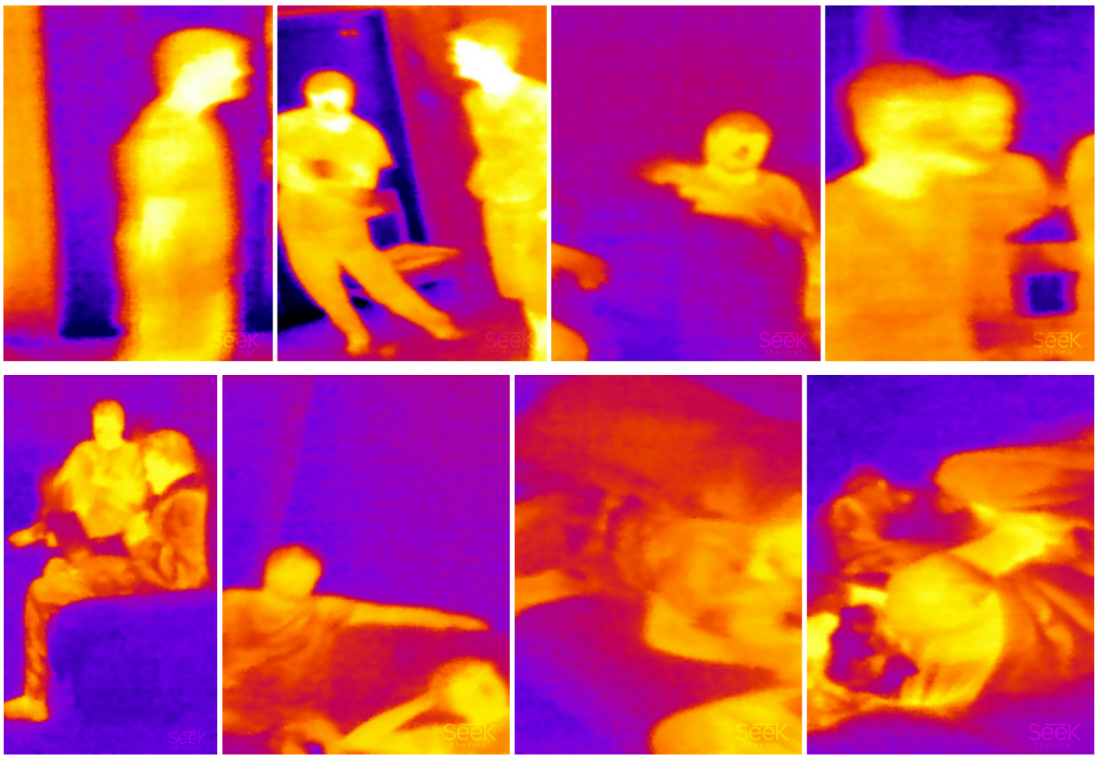


FIGURE 3.3: Different Categories of human poses. From top left clockwise order: single standing, multiple standing, single sitting, sitting and standing, multiple sitting (2), single sitting, mix.

an input to Caffe. During the process, we snapshot the model at every 100 iterations. We create a 1600 x 4096 dimensional ndmatrix using the python NumPy library [88] and input the images in a batch size of one to merge the extracted feature into 1600 x 4096 dimensional matrix. Table 3.1 shows the results of using the three approaches on the IR dataset.

Dataset	**Approach**	**Accuracy**
IR Augmented	Caffenet Fine tuned	72.4%
IR Augmented	Caffenet + SVM	69.03%
IR Augmented	Caffenet Fine tuned + SVM	76.75%

TABLE 3.1: Results of CNN on IR dataset.

3.2.3 Baselines

For a comparison of above results, we need a baseline. As mentioned above, infrared images have prominent colors, and thus it makes sense as a natural baseline as explained below.

Color Histogram: To classify the images we used color histograms in RGB space as discussed in Chapter 3. We downsampled each image to a 225x300 resolution using the CImg library [90] irrespective of orientation of the image. We created 256 bins for color histogram values for red, blue and green, and stored numerical values in a multi-dimensional array using CImg library [90] in C++. We wrote the normalized histogram values for each image in a file. For training a classifier, we use scikit-learn library [91] to read the histogram and corresponding labels. We divided the dataset into 800 training and 400 testing images by randomly assigning each image to either train or test set. We train an SVM and achieved about 62% accuracy on the test set compared to about 50% random guess. Some of the incorrectly classified images are shown in Figure 3.4.

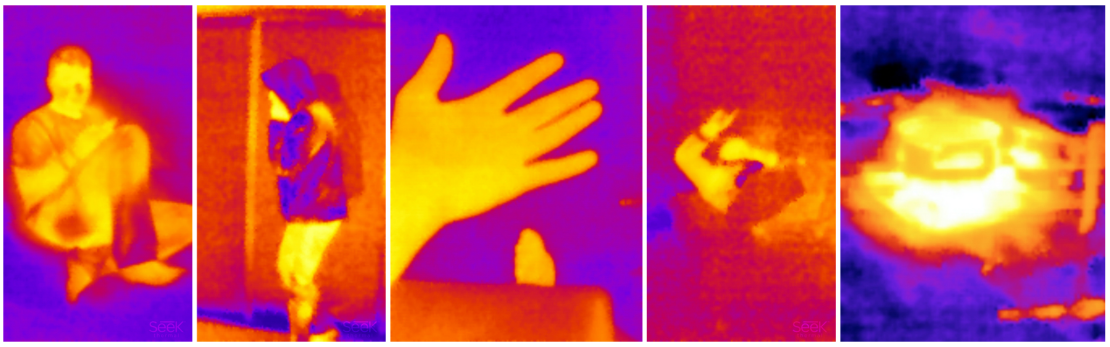


FIGURE 3.4: Incorrectly classified images using color histogram.

Even though infrared images have a small number of unique colors distributed over the range, it is clear by looking at the results that color intensities for different objects can be in a similar range as that of humans. To recognize humans, we need more advanced methods as discussed below.

Another reasonable baseline is based on Histogram of Oriented Gradients [80] as these feature descriptors are known to work well for pedestrian recognition in visible spectrum images. To compute the baseline, we downsampled each image to a 300x400 resolution, irrespective of the orientation. We used both OpenCV [92] and MATLAB [93] to compute HOG features for each image and write them in a separate file. We then used scikit-learn to train an SVM by reading the data from the file and dividing it into training and testing sets. We randomly separated 800 images for training set and 400 images for testing set. We achieved about 72% accuracy over all testing images. We separated images based on their categories and randomly selected equal numbers of non-human images from the dataset. We then divided the new smaller dataset in 4:1 training testing ratio and trained different per-category based SVMs. As expected, the accuracy of single standing (Category 1) significantly exceeds other categories, at about 94%. Table 3.2 mentions performance over different categories and the complete test set. It is clear that HOG works quite well for clutter free human recognition in infrared images, as in case of category 1. However, in the case of random and varying sitting and standing poses with occlusion, the results are not as promising. Few images contained reflections on glass window panes but no humans. Those images are classified correctly. Even though there were no humans in such images, we counted them as correctly classified in the Standing only category.

Dataset type	**Accuracy**
Standing only	94.67%
Multiple Standing	81.3%
Single Sitting	76.19%
Standing and Sitting	65.1%
Multiple Sitting	68.4%
Mix	64.32%
Overall	71.8%

TABLE 3.2: HOG Performance on IR dataset and its subsets.

3.3 Fine-grained classification problem

We train CNN models to recognize 447 individual Right Whales which are left in the oceans. For aiding the conservation efforts, and to save the Right Whale species from extinction, it is necessary for NOAA to maintain and observe health status of each of the remaining right whales. Through the background study of the problem [6], we found that whale faces are an important part of the whale body from which experienced researchers identify individual whales. As explained in Figure 3.5, as whales grow older, callosities (white color patches) begin to form, which remain for a long time and are distinctive to individual whales. So we cropped the whale images into a 256x256 pixel size using Sloth [94] annotations [95]. A few examples of whale faces are shown in the bottom row of Figure 3.6.

3.3.1 Dataset and Experiments

For whale recognition, we use the NOAA Right Whale Dataset [79]. The whale dataset contains about 4500 training images and 7000 testing images (without testing labels) and a total of 447 classes. These images are taken over a span of about 10 years by researchers and NOAA employees. The 447 classes belong to the 447 individual whales which are left in the worlds oceans. The dataset is quite challenging considering the variations in the appearance of same whale, i.e. whales have different appearance in images based on the camera angle and the activity they are doing. The variation in the appearance is shown in Figure 3.6. On average we have about 10 images per class, but the data is quite unevenly distributed; i.e. more than 50% of the classes have 9 images or less and about 30% have 5 images or less, thus making the recognition task challenging.

Data Pre-processing: We take the data from the train labels file and create a (key, value) mapping to numbers from 1 to 447. Since we do not have labels for the 7000

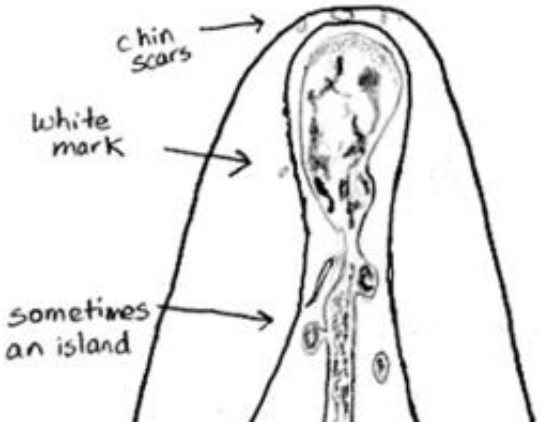


FIGURE 3.5: Whale faces have important characteristic to classify individual whales. Image source: NOAA[6].

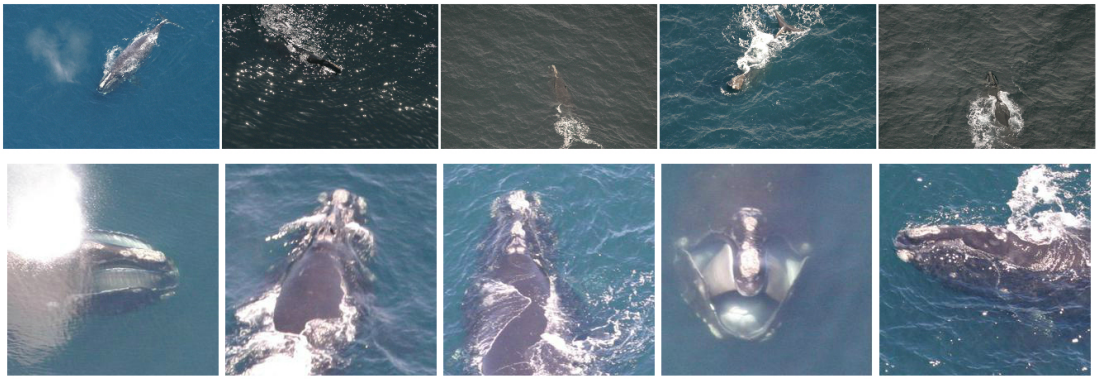


FIGURE 3.6: Sample images from NOAA Right Whale Dataset. First row: first three photos are of the same whale while last two are of another whale. Second row: whale faces all belonging to the same whale.

testing images, we randomly choose about 10% of the training data to use as our testing set to create three different datasets. In other words, we create 3 partitions of training and testing data from the original training data by randomly choosing the testing set. One such set contains exactly 4090 training images and about 400 testing images. Later, we will average the result of all 3 datasets to calculate an accuracy figure. Once the datasets are generated, we convert them to an LMDB file format database, which is required by Caffe to process images efficiently.

Data Augmentation: Since the data size is quite small for training a CNN from scratch, we used a data augmentation technique to prevent over-fitting. We used horizontal flips, vertical flips, 90, 180 and 270 degree rotations to increase our dataset size by a factor of

six. For the augmented dataset, we have a total of about 27000 training images. From this, we again randomly select 10% of the data as testing and create two such datasets as described above. Again, we average the results of both datasets to come up with an accuracy figure.

During the fine-tuning of CaffeNet, we snapshot the model at every 500 iterations and use that for extracting features in subsequent steps. We repeated the above steps for another pre-trained model on Flickr style images. Next section reports the result we obtained.

3.3.2 Result

We achieved the highest accuracy of about 21% on the whale faces augmented dataset using an SVM as a classifier on features extracted from further fine tuning a Flickr-Finetuned Caffe model. The best results occurred at the 7000 iteration snapshot of the fine-tuning. It provided about 20% accuracy on the faces dataset, and about 14% accuracy on the whale only dataset compared to a random guess accuracy of 0.22%. We believe that this accuracy is better than that of unexperienced human accuracy. Since whales look almost same to unexperienced humans, it is difficult to identify individual whales. Overall results are shown in Table 3.3. We use the same system configuration as we used for the human recognition task.

3.4 Egocentric Scene Classification Problem

In this section, we discuss the egocentric dataset and experiments performed on it. Because of the nature of the problem (multi-label classification, section 3.4.2), the experiments are slightly different than previous experiments, which were multi-class classification.

Dataset	Approach	Accuracy
Whale only	Caffenet + SVM	12.44%
Whale only	Caffenet Finetuned	13.5%
Whale only	Caffenet Finetuned + SVM	18.06%
Whale only	Flickr Finetuned + SVM	13.69%
Whale faces + augmentation	Caffenet + SVM	12.63%
Whale faces + augmentation	Caffenet Finetuned	14.86%
Whale faces + augmentation	Caffenet Funetuned + SVM	18.52%
Whale faces + augmentation	Flickr Finetuned + SVM	21.8%
Whale faces + augmentation	Flickr Finetuned	19.94%

TABLE 3.3: CNN results on NOAA Right Whale dataset.

3.4.1 Dataset

We collected the data manually [84] using a Narrative Clip wearable camera [96]. Two users wore it through out the day over a period of 5 months on their shirts capturing daily activities. The photographs were manually labeled by several users using a web based interface. The users are asked to provide natural language English sentences describing what activities they are doing if they assume they are the one wearing the camera, what is happening in the photographs from a third persons perspective assuming the photograph is not from a wearable camera, the location where the photograph is taken, actions and activities happening in the photographs and important objects in the photograph. These inputs are taken from multiple users over several months. Overall, the dataset contains more than 6000 photographs out of which we manually labeled a total of 900 photographs with relevant tags. Table 3.4 shows sample images and their tags by several users (duplicates removed).

Since the tags are user created, they are ambiguous. For example, users can describe a room as *indoors*, *indoor*, *inside*, *inside a room* and a person as *man*, *people*, *human*, *person*, etc. To increase the labeled images with tags, we manually created a mapping between words and phrases to combine similar tags and scan for relevant words and

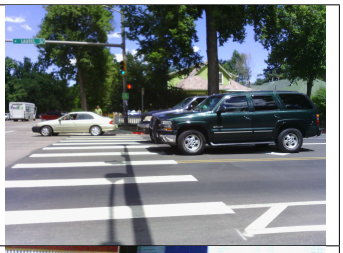
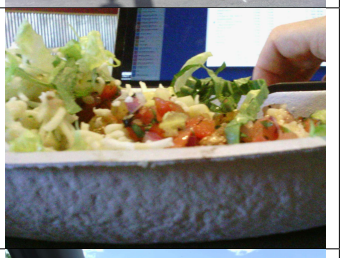
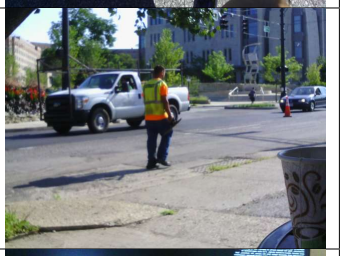
outside, car, road, street, walking, signal, traffic, traffic light, zebra crossing, vehicle, trees, shadow, outdoors	
food, meal, finger, hand, eating, typing, laptop, laptop screen, lettuce, working, inside, restaurant, indoors, vegetables, bowl, salad	
car, toyota logo, hand, steering wheel, wrist watch, driving, trees, outside, inside car, dashboard, travel, outdoors	
inside, elevator, instructions, button, indoors, looking	
bike, biking, hand, road, outside	
road, street, car, truck, building, walking, looking, trees, man, construction worker, outdoor	
monitor, table chair, paper, markers, sitting, working, furniture, indoor, screen, programming, table, chair, inside a room	

TABLE 3.4: Egocentric dataset images and their corresponding user generated tags.

phrases in the sentence descriptions. For example, *inside refrigerator* photos are tagged as *inside* and *home* because that is where a refrigerator is most likely be found. Overall, we processed about 1100 images with an average of 6.58 tags per image. To increase the dataset size, we flipped each image horizontally, which increased the dataset size by a factor of two. We created three top level tag categories based on Location, Objects and Activity, and selected the most frequent tags falling into these three categories:

Location: *inside*, *outside*, *restaurant*, *home*, *street*, *office*.

Objects: *trees*, *laptop*, *hand*, *table*, *man*.

Activity: *walking*, *sitting*, *eating*, *working*, *typing*, *driving*.

3.4.2 Experiments

Predicting tags for egocentric images is a multi-label classification problem, since image can have multiple tags. We used similar methods to train our multi-label classification models with small differences. Caffe supports multi-label classification through SigmoidCrossEntropyLoss layer [97]. Unlike the previous two problems, we use pycaffe to fine-tune Caffenet on our dataset and used Hamming distance as our accuracy measure [98]. We use scikit-learns OneVsRestClassifier, which supports multi-label classification, instead of SVMs. We order the tags based on their relative frequency in each category and assign an index to each of them from 0 to 16. We then create an array of 17-bit vectors to store the corresponding classes as 0 or 1 for each image by using MultiLabelBinarizer class and training the multi-label classifier.

3.4.3 Results

For each of the steps above, we randomly divided our data in a 4:1 ratio for training and testing. We achieved the highest accuracy of about 89% by using features from the fine-tuned model. Since the number of possible classes is large (17) and the average number

of tags is around 6 per image, naively predicting zeroes would produce an accuracy of about 64%. To test our model more thoroughly, we labeled 1000 random unlabeled images from the pool of egocentric dataset. To speedup the labeling process, we use Clarifai API [99], which is an online service for generating tags based on an image, and then manually verified and replaced them to fit our top 17 labels. For example, Clarifai always outputs indoors for an indoor scene, whereas we used inside. We replaced all other tags like road and sidewalk to street, etc. This new test set produced about 86% accuracy.

We also separated data based on mutually exclusive tags like *inside* & *outside*, *walking* & *sitting*, etc. and trained separate classifiers for each of them. We achieved accuracies ranging from 95% to 98% in most cases. For example, simple binary SVM classifiers on features from Caffenet on tags inside, outside; sitting, walking etc. give near perfect accuracy on the newly generated test data. Table 3.5 gives some of the images and their generated labels.

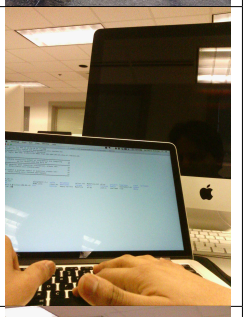
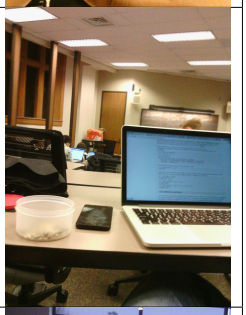
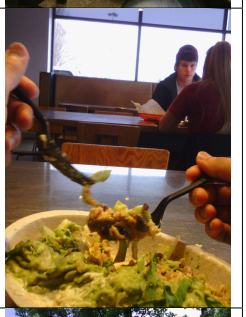
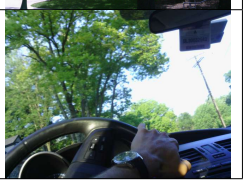
User generated: outside, trees, walkway, walking, road, car, snow, looking, shadow Ground truth: outside, street, trees, walking Prediction: outside, street, trees, walking	
User generated: inside, office, working, laptop, hand, programming, typing, mac, apple logo, macintosh, screen, reflection, keyboard, inside a room, lab, sitting Ground truth: inside, office, laptop, hand, sitting, working, typing Prediction: inside, office, laptop, hand, table, working, typing	
User generated: inside, class, classroom, box, phone, working, laptop, macbook, screen, keyboard, sitting, door Ground truth: inside, laptop, table, sitting, working Prediction: inside, laptop, table, sitting, working	
User generated: inside, indoor, restaurant, people, man, woman, table, hand, fork, chipotle, food, meal, eating, sitting Ground truth: inside, restaurant, hand, table, man, eating, sitting Prediction: inside, restaurant, hand, table, man, eating, working	
User generated: trees, university, outdoor, walking, looking Ground truth: outside, street, trees, walking Prediction: outside, street, trees, walking	
User generated: trees, watch, wrist watch, outside, driving, steering wheel, car, inside a car, inside, dashboard, hand, traveling Ground truth: inside, outside, trees, hand, sitting, driving Prediction: outside, trees, driving	

TABLE 3.5: Egocentric dataset images and their corresponding user generated tags.

Chapter 4

Conclusion and Future Work

We ran CNNs on three different non-traditional computer vision problems and evaluated their performance. We observed that even though CNNs are good classifiers for general image classification tasks [16, 18], they do not work as well in the case of our infrared dataset, while they worked well in case of fine-grained recognition problem and egocentric images.

We suspect that CaffeNet did not produce good results on IR dataset because it was trained on ImageNet data, which contains only visible spectrum images. A network trained specifically on infrared images might produce better results. Also, there are huge variation in poses, occlusions and thermal noise in the images in IR dataset as shown in Figure 4.1. For these variety of poses, temperature information of different body parts (like face, hair, hands, etc.), and parts annotations would be useful for a better accuracy. For example, different poses can be modeled and separate classifiers could be trained for different parts of human body based on shape, temperature, pose, orientation, etc., and then fed into a classifier.

It is difficult for CNN to learn distinctive shape, color and parts of humans. We can say that humans, in clear standing pose, are clearly identifiable because of their unique

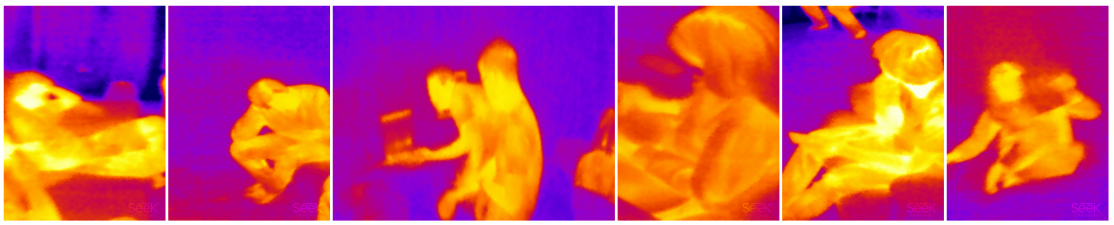


FIGURE 4.1: Some images of humans in IR Dataset.

thermal signature, characteristics and clearly visible body parts and limbs. However, varying occlusions and poses significantly impact the recognition accuracy. Also, considering the number of possible poses a human can take, the dataset size is relatively small (600 images) to train a network from scratch specifically for infrared images. We suspect that a larger infrared images dataset would produce better results.

For the whale classification task, CNNs worked better compared to the baseline accuracy of 0.23%, achieving about 21% which is better than unexperienced human accuracy. We also observed that domain knowledge (for example, whale faces) and data augmentation helps CNNs learn better features. The accuracy could potentially be improved by incorporating more domain knowledge. In case of egocentric images, we see that CNNs performed remarkably well. We reason that is because of a well defined clean dataset having specific characteristics and distinctive classes. Even while using similar architectures, the significant difference in accuracies shows that, like any other machine learning model, performance of CNN depends on the architectural design, quality of dataset, and parameter tuning. A dataset with clearly distinctive classes perform better than the one having subtle differences. To learn the subtle differences, CNNs needs to learn at a very lower level of abstraction to differentiate between different classes.

Future Work: there is a lot of scope of improvement in our work. We can evaluate additional less traditional computer vision problems. In the case of IR Dataset, a larger dataset would help in producing better results. Our dataset contains only 600 human images, that too in many different poses. As mentioned above, modeling different poses based on human body parts annotations and segmentation would definitely

help. Also, to use the more information about temperature, hot patches, highest/lowest temperature values, Seek Thermal API can be used whenever it becomes available. This additional information will not only help generate better results in recognition, but also in localization. R-CNN [17] based detection can be employed to accurately locate humans.

As mentioned in section 3.1.1, effectiveness of fine-tuning significantly depends on similarity with the original dataset. In this case, we used the networks which were trained on visible spectrum images, and fine-tuned them on infrared spectrum images. If enough data becomes available, one can train a new network from scratch. For the fine-grained recognition problem, we can include more domain knowledge, like distinctive shapes of callosities, body shape, etc. along with more data augmentation techniques like random crops and color jittering. We can also experiment and validate results on other fine-grained datasets like flowers [47], larva [48], birds [27], and dogs [28]. In the case of egocentric images, we had only a few type of each category. A person interacts with much more objects and does a lot more types of activities, which can be included. Also, the dataset we used had images from only two users; we can include images from more users as well to build a more general dataset.

Bibliography

[1] Electro magnetic spectrum and light. https://9-4fordham.wikispaces.com/Electro+Magnetic+Spectrum+and+light.

[2] Navneet Dalal. *Finding people in images and videos*. PhD thesis, Institut National Polytechnique de Grenoble-INPG, 2006.

[3] Perceptron. http://i.stack.imgur.com/KUvpQ.png.

[4] Multilayer perceptron. http://technobium.com/wordpress/wp-content/uploads/2015/04/MultiLayerNeuralNetwork.png.

[5] Cnn architecture. http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg.

[6] NOAA Catalog. http://rwcatalog.neaq.org/Default.aspx.

[7] How useful is in-camera face detection? http://www.photoreview.com.au/tips/shooting/insider-how-useful-is-in-camera-face-detection.

[8] Smile shutter. http://docs.esupport.sony.com/dvimag/DSCH70_guide/eng/contents/03/04/03/03.html.

[9] Seeing ai app. http://blogs.microsoft.com/next/2016/03/30/decades-of-computer-vision-research-one-swiss-army-knife.

[10] Google photos. https://photos.google.com/.

[11] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[12] Image features: Edges and corners. https://classes.soe.ucsc.edu/cmpe264/Fall06/Lec5.pdf.

[13] Color histogram. https://en.wikipedia.org/wiki/Color_histogram.

[14] Convolutional neural network. https://en.wikipedia.org/wiki/Convolutional_neural_network.

[15] Feature learning. https://en.wikipedia.org/wiki/Feature_learning.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[18] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[20] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[21] CIFAR. https://www.cs.toronto.edu/~kriz/cifar.html.

[22] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The MNIST database of handwritten digits, 1998.

[23] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[25] Vijay John, Seiichi Mita, Zheng Liu, and Bin Qi. Pedestrian detection in thermal images using adaptive fuzzy c-means clustering and convolutional neural networks. In *IAPR International Conference on Machine Vision Applications (MVA)*, pages 246–249. IEEE, 2015.

[26] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[27] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[28] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proceedings of IEEE Computer Vision and Pattern Recognition Workshop on Fine-Grained Visual Categorization (FGVC)*, 2011.

[29] Wearable technology. https://en.wikipedia.org/wiki/Wearable_technology.

[30] Seek thermal. http://www.thermal.com/.

[31] Flir. http://www.flir.com/.

[32] Visible spectrum. https://en.wikipedia.org/wiki/Visible_spectrum.

[33] Infrared. https://en.wikipedia.org/wiki/Infrared.

[34] Massimo Bertozzi, Alberto Broggi, Alessandra Fascioli, Thorsten Graf, and Marc-Michael Meinecke. Pedestrian detection for driver assistance using multiresolution infrared vision. *IEEE Transactions on Vehicular Technology*, 53(6):1666–1678, 2004.

[35] Frédéric Suard, Alain Rakotomamonjy, Abdelaziz Bensrhair, and Alberto Broggi. Pedestrian detection using infrared images and histograms of oriented gradients. In *IEEE Intelligent Vehicles Symposium*, pages 206–212. IEEE, 2006.

[36] Li Zhang, Bo Wu, and Ram Nevatia. Pedestrian detection in infrared images based on local shape features. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[37] Bin Qi, Vinod John, Zheng Liu, and Seiichi Mita. Pedestrian detection from thermal images with a scattered difference of directional gradients feature descriptor. In *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2168–2173. IEEE, 2014.

[38] Harsh Nanda and Larry Davis. Probabilistic template based pedestrian detection in infrared videos. In *IEEE Intelligent Vehicle Symposium*, volume 1, pages 15–20, 2002.

[39] Jan Portmann, Simon Lynen, Maria Chli, and Roland Siegwart. People detection and tracking from aerial thermal views. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1794–1800. IEEE, 2014.

[40] James W Davis and Mark A Keck. A two-stage template approach to person detection in thermal imagery. In *IEEE Winter Conference on Applications of Computer Vision*, pages 364–369. IEEE, 2005.

[41] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013.

[42] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1058–1066, 2013.

[43] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649. IEEE, 2012.

[44] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[45] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[46] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md Patwary, Mostofa Ali, Ryan P Adams, et al. Scalable bayesian optimization using deep neural networks. *arXiv preprint arXiv:1502.05700*, 2015.

[47] Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1447–1454. IEEE, 2006.

[48] Gonzalo Martinez-Munoz, Natalia Larios, Eric Mortensen, Wei Zhang, Asako Yamamuro, Robert Paasch, Nadia Payet, David Lytle, Linda Shapiro, Sinisa Todorovic, et al. Dictionary-free categorization of very similar objects via stacked evidence trees. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 549–556. IEEE, 2009.

[49] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, European Conference on Compute Vision*, volume 1, pages 1–2. Prague, 2004.

[50] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3360–3367. IEEE, 2010.

[51] Bangpeng Yao, Aditya Khosla, and Li Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1577–1584. IEEE, 2011.

[52] Steve Branson, Catherine Wah, Florian Schroff, Boris Babenko, Peter Welinder, Pietro Perona, and Serge Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision*, pages 438–451. Springer, 2010.

[53] Ryan Farrell, Om Oza, Ning Zhang, Vlad I Morariu, Trevor Darrell, and Larry S Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *IEEE International Conference on Computer Vision*, pages 161–168. IEEE, 2011.

[54] Bangpeng Yao, Gary Bradski, and Li Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3466–3473. IEEE, 2012.

[55] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014.

[56] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.

[57] Ning Zhang, Ryan Farrell, and Trever Darrell. Pose pooling kernels for sub-category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3665–3672. IEEE, 2012.

[58] Alejandro Betancourt, Pietro Morerio, Carlo S Regazzoni, and Matthias Rauterberg. The evolution of first person vision methods: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(5):744–760, 2015.

[59] Thad Eugene Starner. *Wearable computing and contextual awareness*. PhD thesis, Massachusetts Institute of Technology, 1999.

[60] Thad Starner, Bradley Rhodes, Joshua Weaver, and Alex Pentland. Everyday-use wearable computers. In *International Symposium on Wearable Computers*, page 9, 1999.

[61] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998.

[62] Takeshi Kurata, Takashi Okuma, Masakatsu Kourogi, and Katsuhiko Sakaue. The hand mouse: Gmm hand-color classification and mean shift tracking. In *Proceedings of IEEE International Conference on Computer Vision Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*, pages 119–124. IEEE, 2001.

[63] Thad Starner, Bernt Schiele, and Alex Pentland. Visual contextual awareness in wearable computing. In *Second International Symposium on Wearable Computers, 1998. Digest of Papers.*, pages 50–57. IEEE, 1998.

[64] Antonio Torralba, Kevin P Murphy, William T Freeman, and Mark A Rubin. Context-based vision system for place and object recognition. In *Proceedings of IEEE International Conference on Computer Vision*, pages 273–280. IEEE, 2003.

[65] WW Mayol and David W Murray. Wearable hand activity recognition for event summarization. In *IEEE International Symposium on Wearable Computers*, pages 122–129. IEEE, 2005.

[66] Hongwen Kang, Alexei A Efros, Martial Hebert, and Takeo Kanade. Image matching in large scale indoor environment. In *IEEE Computer Vision and Pattern Recognition Workshops*, pages 33–40. IEEE, 2009.

[67] Yoshinari Kameda and Yuichi Ohta. Image retrieval of first-person vision for pedestrian navigation in urban area. In *International Conference on Pattern Recognition (ICPR)*, pages 364–367. IEEE, 2010.

[68] Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. Ai goggles: Real-time description and retrieval in the real world with online learning. In *Canadian Conference on Computer and Robot Vision*, pages 184–191. IEEE, 2009.

[69] Kris M Kitani, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3241–3248. IEEE, 2011.

[70] Alireza Fathi, Xiaofeng Ren, and James M Rehg. Learning to recognize objects in egocentric activities. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3281–3288. IEEE, 2011.

[71] Alireza Fathi, Ali Farhadi, and James M Rehg. Understanding egocentric activities. In *IEEE International Conference on Computer Vision*, pages 407–414. IEEE, 2011.

[72] Hamed Pirsiavash and Deva Ramanan. Detecting activities of daily living in first-person camera views. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2847–2854. IEEE, 2012.

[73] Michael Ryoo and Larry Matthies. First-person activity recognition: What are they doing to me? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2730–2737, 2013.

[74] Sven Bambach, Stefan Lee, David J Crandall, and Chen Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1949–1957, 2015.

[75] Minghuang Ma, Haoqi Fan, and Kris M Kitani. Going deeper into first-person activity recognition. *arXiv preprint arXiv:1605.03688*, 2016.

[76] Daniel Castro, Steven Hickson, Vinay Bettadapura, Edison Thomaz, Gregory Abowd, Henrik Christensen, and Irfan Essa. Predicting daily activities from egocentric images using deep learning. In *Proceedings of the 2015 ACM International Symposium on Wearable Computers*, pages 75–82. ACM, 2015.

[77] Michael S Ryoo, Brandon Rothrock, and Larry Matthies. Pooled motion features for first-person videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 896–904, 2015.

[78] Suriya Singh, Chetan Arora, and CV Jawahar. First person action recognition using deep learned descriptors. 2016.

[79] Kaggle Right Whale Recognition Challenge. https://www.kaggle.com/c/noaa-right-whale-recognition/data.

[80] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE, 2005.

[81] Feedforward Neural Network. https://en.wikipedia.org/wiki/Feedforward_neural_network.

[82] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[83] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[84] Chenyou Fan, personal communication, April 2015.

[85] Inductive Transfer. https://en.wikipedia.org/wiki/Inductive_transfer.

[86] Transfer Learning. http://cs231n.github.io/transfer-learning/.

[87] CNNs. http://cs231n.github.io/convolutional-networks/.

[88] NumPy. http://www.numpy.org/.

[89] Harrison's principles of internal medicine. Vol. 2. New York: McGraw-Hill Medical, 2008.

[90] David Tschumperlé. The CImg Library. In *IPOL 2012 Meeting on Image Processing Libraries*, pages 4–pp, 2012.

[91] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[92] OpenCV Library. http://opencv.org/.

[93] MATLAB. http://www.mathworks.com/products/matlab/.

[94] Sloth Annotation tool. https://github.com/cvhciKIT/sloth.

[95] Whale Face Annotations. https://www.kaggle.com/c/noaa-right-whale-recognition/forums/t/17421/.

[96] Narrative Clip. http://getnarrative.com/.

[97] Caffe documentation. http://caffe.berkeleyvision.org/doxygen/classcaffe_1_1SigmoidCrossEntropyLossLayer.html.

[98] Hamming distance. https://en.wikipedia.org/wiki/Hamming_distance.

[99] Clarifai. http://clarifai.com/.