博士論文 (要約)

Robust Object Detection with Deep Learning by Utilization of Motion Information and Generalization to Unknown Environment: Application to Wide-Area Surveillance of Birds

(頑健な物体検出のための深層学習における
動きの利用と未知環境への汎化
–野鳥の広域監視への応用–)

Graduate School of Information Science and Technology
The University of Tokyo

Ryota Yoshihashi
吉橋　亮太

Advisor:　Prof. Takeshi Naemura

December 2018

Abstract

The advancement in automatic image recognition has paved the ways to diverse real-world applications including wildlife monitoring, automatic driving, robotics, or security. Such applications require further robustness of recognition systems than that of current ones. Currently, the most used approach in image recognition is deep learning, which enabled learning-based acquisition of feature representations. Although it has been shown to be efficient and effective in many generic image recognition tasks, their major successes concentrate in web-based applications, where large-scale and well-annotated data is easily available. Deep learning in image recognition can be characterized as a data-hungry method for acquiring rich visual features. This means that it works best in environments where we can take advantage of the existence of big data. However, this is not the case in most other real-work applications. There are some characteristics of real-work applications that make them different from web-image recognition. Here we refer three of them: 1) Domain specificity, 2) low-resolution targets, and 3) open world. For such situations, there is room for discussion whether and how we can enjoy deep learning's strength.

This work studies how deep learning can be robustly applied to such real-world problems. Specifically, we introduce two ideas in deep-learning-based object detection. First, we utilize motion information to differentiate small objects with low visibility. In detecting small-looking birds in wide-area surveillance, such motion cues especially take essential roles. Second, we enhance the detector's generalization to unknown environments by introducing an unknown handling mechanism. Important existing work is the open-set classifier, which can safely reject unknown samples that the classifier did not learn. We extend this to be applicable to detection tasks.

The main focus of this study is in wild-bird surveillance, which is a novel and practical application of computer vision. It is also at an opposite extreme of generic image recognition, in term of its nature of domain specificity, low-resolution targets, and open world. Here we introduced three ideas to overcome these difficulties. First, we introduced domain-specific datasets for bird surveillance that offer challenges of robustness due to the "in-the-wild" nature of the task and are suitable to discuss the robustness with them. Second, we introduced motion-based object detection models that are more robust in detecting visibly small objects than detectors that rely only on appearances. Third, we introduce novelty-tolerant detectors that can handle 'unknown' objects that often appear in the open world.

The highlights of this thesis's contribution are summarized as follows:

- We provide the first practical image dataset for the task of bird recognition at wind farms (Chapter 3).
- We introduce the first deep-learning-based motion feature that is useful in detection tasks and improves detection accuracy by $\sim$10% in pedestrian detection and bird detection (Chapter 4).
- We introduce a novel joint detection and tracking framework, named *Recurrent Correlation Network*, where detection and tracking help each other in terms of motion-feature learning (Chapter 5).
- We introduce *novelty-tolerant detection*, which handles 'unknown' objects by exploiting open-set classifiers. We further improve the existing open-set classifiers by developing *Classification-Reconstruction learning for*

Open-Set Recognition (CROSR), a novel learning framework for open-set learning (Chapter 6).

Table of contents

List of figures

List of tables

1

Introduction

1.1 Background

Machines that can fully understand the external environment are desired but at this stage have never been achieved. However, over the last decade, the field of machine perception has made outstanding progress. In particular, the advancements in automatic image recognition technologies are so significant that they are now being applied in the real world, and many societies and companies have embraced diverse applications including wildlife monitoring [4], automatic driving [5], robotics [6], and security [7]. Despite the high expectations, it is not yet fully understood exactly how robust machine intelligence is. At least, even the state-of-the-art image recognition systems easily cause mistakes that are not likely to be cause by humans. Thus, to deploy recognition systems in the real world, we need to advance our understanding of machine intelligence's robustness and build systems that are more robust than the current ones.

Currently, the most commonly used approach in image recognition is deep learning [8, 9], which is recently able to perform learning-based acquisition of feature representations. Conventional visual recognition systems relied on *hand-crafted* feature representations [10], which were designed by image-recognition researchers on the basis of biological or geometrical inspirations and then hard-coded. While there was some success with this in the early stages, and some of the systems are even currently being used in a few applications, it turned out to be suboptimal for large-scale visual recognition. In contrast, deep learning-based image recognition has been successfully applied to large-scale image recognition tasks featuring more than one million images [11] and has shown excellent performance [12]. In light of the excellent performance of deep learning on large-scale benchmarks, one might conclude that visual recognition is as good as it can get, and is ready for application to real-world problems.

Unfortunately, such optimism is misplaced for many applications. Multiple studies have indicated that current deep networks, even when trained using one of the largest Web-image datasets [11] in the computer-vision community, do not perform well when the settings are different from the original training environment [6, 13]. As an example, Fig. 1.1 shows a failure case by a state-of-the-art recognition system with a non-generic image that we collected. To clarify the current limitations of deep learning, we must take into account that the most successful use cases tend to be concentrated in Web-based applications. In such applications, large-scale and well-annotated data is easily available. Deep learning in image recognition can be characterized as a data-hungry method for acquiring rich visual features. This means that it works best in environments where we can take advantage of the existence of big data. However, such conditions are not guaranteed in most real-world applications that are not on the Web. In many cases, we need to tackle visual-recognition problems without relying on the very factors that make deep learning successful. Fortunately, deep learning's ability to automatically learn visual representations from data seem promising even in these less-than-ideal situations. Our goal in this thesis is therefore to evaluate and improve the robustness of deep learning-based methods in real-world scenarios.

Robustness is defined as follows: "A statistical method is robust when it provides almost valid results with regards to data that do not satisfy the requirements or assumption that the method needs [14]." Thus, to determine whether a recognition system is robust or not, we should first question which assumptions are needed for successful recognition of the system. Among the multiple assumptions behind large-scale Web-image recognition responsible for the success of deep learning, we focus on three. 1) Generality. In general-purpose recognition systems, the research focus is limited to targets that are common enough to be caught by image search engines [15]. This has resulted in the availability of extremely big data, which is advantageous for data-hungry methods. We should point out that the word *general* was originally used to refer to the setting that put no constraint on tar-

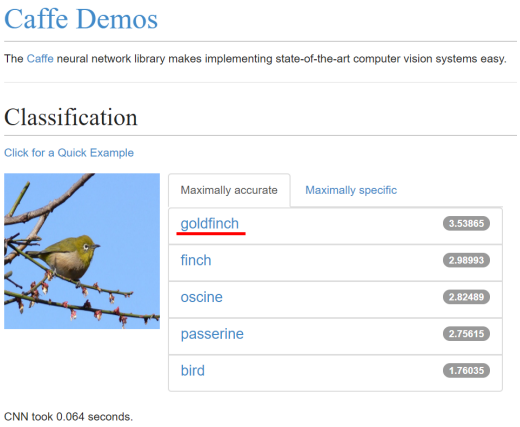
get objects, in contrast to more classical recognition settings that were examined under controlled illumination, poses, or types of instances [16]. However, recent studies simply refer to images randomly crawled from the Web as 'generic' images. 2) Visibility. Visual recognition basically concerns only things that are visible; small and non-salient objects outside of the focus may be ignored and considered part of the background in 'generic' usages. This has enhanced the effectiveness of rich features for object recognition. 3) Learnability. Learning-based systems need to learn before being deployed, and they can only learn from prepared training data. The behavior of a recognition system when confronted with an object it has not learned is essentially undefined, but this has not been of much concern in the conventional settings. In typical evaluation criteria for generic image recognition, we do not need to handle such 'unknown' things.

However, there are some characteristics of real-world applications that violate these assumptions. In correspondence to the three assumptions above, this thesis refers to three difficulties inherent in real-world problems: domain specificity, low-resolution targets, and open world.

Domain specificity Recognizing images from a certain application setup means handling objects and backgrounds in a specific scene. Therefore, in domain-specific problems, the assumption of generality is not satisfied. Domain specificity has both pros and cons. On the positive side, we only need to handle narrower visual variances compared to generic image recognition. The negative point is in the difficulty of data collection. Domain-specific recognition systems need domain-specific image data for training, and such data are difficult to collect from the usual sources, i.e., the Web. Carelessly collected images from the Web are hardly useful for domain-specific problems due to the gaps between generic and domain-specific tasks. For example, if we look at the 'people' class in a generic Web-image dataset (Fig. 1.2), we can see that a large portion of the images is from indoor scenes such as a party. Although this is the nature of images on the Web, it does not provide good training data for tasks like pedestrian detection in auto-driving vehicles due to the difference in the appearance of persons.

Low-resolution targets In outdoor tasks, we often work with targets that have low resolutions, which violates the assumption of visibility. Even if the real size of a target is large, it might look small in an image due to distance. This is problematic because detecting distant objects is quite important in some applications: for example, autonomous vehicles need to notice pedestrians even when they are distant from the vehicle in order to anticipate potential collision as early as possible. Somewhat surprisingly, Web-based applications have not seemed so concerned about this. The reason may stem from the thematic nature of user-taken photographs; specifically, photographers usually put their object of interest in the center of the field of view. This is not the case for fixed-point surveillance cameras.

Open world One of the major limitations of learning-based recognition systems is that they need to learn—they essentially cannot recognize things that they have not learned in advance. Thus, if an object that was not in the training data appears in the test settings, it violates the learnability of that object. Most of the existing systems assume that there is nothing they do not know, and they have no direction on what to do when they face unknown things. This assumption, called a *closed-world assumption*, can be easily violated in real-world problems, where covering all possible classes is extremely difficult or even impossible. There are a few possible causes for the appearance of an unknown object. One is dynamicity of the environment. If the environment is dynamic, a type of object that was not frequent in the training phase may be frequent in the testing phase. For example, the large ImageNet released in 2009 does not contain any images of an iPhone [18]. Another possible cause is long-tail



a) Successfully recognized image from the Web

b) Incorrectly recognized image from a surveillance setup

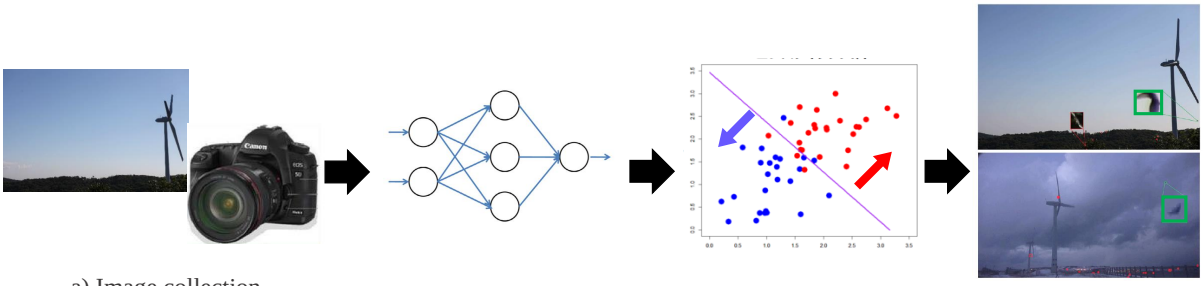
Figure1.1: A failure case of a generic-image recognition system in real-world data. Even systems with a state-of-the-art recognition model [17] which can recognize images from the web accurately, often fail to recognize images from real-world application due to the difference of appearance.

Figure1.2: Comparison between *generic* web images and domain-specific images from real-world applications.

distributions. In the real world, there are more objects that appear rarely than expected by usual (e.g. Pareto) distributions [19]. Such objects may matter after deployment, even if they were not observed in the finite training samples. A third possible cause is unintended use. People often use systems for purposes that are different to what the systems were originally designed for. This exposes a system to a lot of unknowns because the training data were not designed for this or that purpose. In such cases, machines may misrecognize unknown samples as known, thus essentially limiting their usability. For example, a system that was trained to recognize food in images may be fooled by non-food images [20].

1.2 Objective

This work examines how we can improve the robustness of deep learning for real-world application. To discuss the applicability of deep learning, we need a concrete example of real-world problems. The main focus of this



a) Image collection and annotation | b) Model definition | c) Training | d) Evaluation and deployment

Figure1.3: The development procedure of image recognition systems: a) image collection and annotation, b) model definition, and c) training. This thesis tackles problems in each step.

study is wildlife surveillance (specifically, birds), which is a novel and practical application area of computer vision and is at the extreme opposite of Web-based image recognition. Taking this application as an example, we investigate the holistic design of a domain-specific recognition system. The original motivation of our wild-bird surveillance comes from the wind-energy industry, which has looked at the collision of birds with turbines and its ecological impacts. In the present study, we aim to sublimate this concept into the challenge of visual recognition with a new type of difficulty, rather than to provide ad-hoc solutions to the problem.

Here we introduce two ideas in deep-learning-based object detection, which composes our technical contributions. First, we utilize motion information to differentiate small objects with low visibility. In detecting small-looking birds in wide-area surveillance, such motion cues especially take essential roles. Second, we enhance the detector's generalization to unknown environments by introducing an unknown handling mechanism. Important existing work is the open-set classifier, which can safely reject unknown samples that the classifier did not learn. We extend this to be applicable to detection tasks.

To solve a problem in a new domain, we need to holistically re-design the pipeline of the solution. The steps in this solution are shown in Fig. 1.3 and consist of a) image collection and annotation, b) model definition, and c) training. This thesis discusses each step in relation to the above-mentioned difficulties of real-world problems: namely, domain specificity, low-resolution targets, and open world.

Image collection and annotation First, we need to collect image data and annotate them manually. The purposes of this step are to understand the nature of the problem and to provide training and testing data for the system. The problem that we tackle in this step is making our dataset *domain-specific* to the application area of wide-area surveillance. Without domain-specific data, our effort on the downstream pipelines would not produce practical results in the domain of interest. To achieve practical data collection, we developed automatic image acquisition systems with fixed-point cameras and deployed them at wind farms.

Model definition After preparing the datasets, we need to come up with recognition models that are appropriate for the collected data. While convolutional neural networks perform well as image modeling tools, their applicability to *low-resolution targets* is limited. We found that motion information can play a complementary role to visual information in low-resolution moving object detection. Thus, in this step, we discuss how motion information can be incorporated into CNN-based object-detection models.

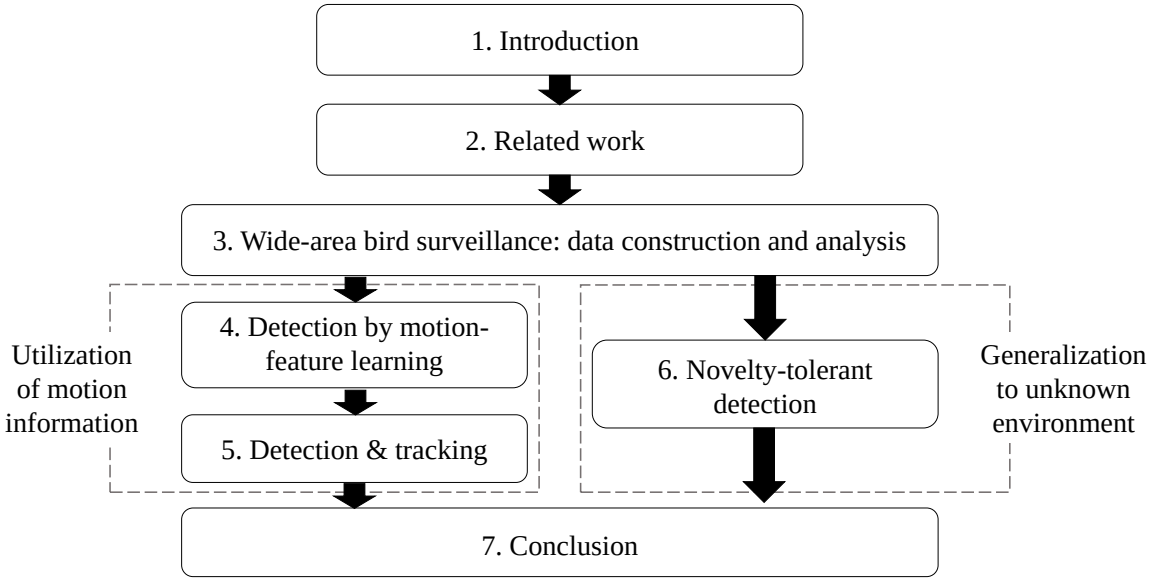


Figure1.4: The structure of this thesis.

Training The conventional recognition models are trained as closed-set classifiers, and this causes problems when deployed in the *open world*. To enable robust object detection in open worlds, we discuss methodology to train neural networks for open-set classification in this step. We further extend the open-set classification method to detection. We name our object detection scheme utilizing an open-set classifier *novelty-tolerant detection* and demonstrate that the novelty-tolerant detectors can perform well in unknown environments that differ from the ones used in training.

1.3 Overview of this thesis

The overview of this thesis is shown in Fig. 1.4. In Chapter 1, *Introduction* we describe the background and objectives. Chapter 2, *Related work* provides a brief history of image-recognition research and the position of this thesis within it. Chapter 3, *Wide-area bird surveillance: Data construction and analysis* corresponds to the a) image collection and annotation step. Here, we describe our image-collection methodology and present analyses of the constructed datasets, which gave us insights and motivations for following chapters. Chapter 4, *Detection by motion feature learning* and Chapter 5, *Detection & tracking* describe our methods for utilizing motion information. They correspond to the b) model definition step. Chapter 6, *Novelty-tolerant detection* corresponds to the c) training step. In Chapter 7, *Conclusion*, we summarize the results of the above chapters and discuss their implications and outlook for the future.

2

Related work

2.1 Overview of image recognition

While the focus of this thesis is in object detection, we briefly review wider area of image recognition to introduce concepts and techniques that are necessary to describe our methods in later chapters.

2.1.1 Concepts and algorithmic frameworks

Image recognition is one of the centric topics in computer vision [1, 21], which aims to enable machines to understand contents of images and extract information. It is a long-standing challenge that dates back to early 1950s. As a legend, the origin of image recognition was 1966 when Marvin Minsky asked a student to "spend the summer linking a camera to a computer and getting the computer to describe what it saw" [1]. However, we can see some precedents of pattern-recognition studies that tried recognizing characters buffered in 2D arrays [22,23], which still can be considered as primitive but pioneering forms of vision.

A framework unchanged from the 1950s to now is that visual recognition is modeled as a function whose input is images and whose output is some information related to its input. The framework can be denoted as:

$$Understanding = Recognition(Image). \tag{2.1}$$

Within the framework, research efforts have been made to give it some variations:

- **Domain**: What input modality is, e.g., RGB image, depth image, thermal image, etc.
- **Task**: What kind of information output is, e.g., a category of an image, location of objects in an image, etc.
- **Model**: What family of functions is used to model input-output relationships, e.g., if-else rules, linear functions, probabilistic graphical models, or neural networks.
- **Learning**: Whether and how the model is trained, e.g., non-learning algorithms, supervised learners, or un-/semi-/weakly-supervised learners.

Domains What kind of information to input to the systems is an important aspect of vision, which has relationships to all the design of the latter parts in the systems. The most primitive form of visual input is **binary patterns**, which was considered in the earliest optical character recognition systems (OCR) [24] or fingerprint identification [25]. **Natural images** are the most usual domain of visual recognition. This refers to three-channel (RGB is the most common) images taken by daily cameras. While giving the exact definition of natural images is difficult, there are some agreed properties that natural images have: 1) They capture continuous values of light intensity, and are quantized in much more levels than in binary images (usually in 8-bit, i.e., 256 levels). 2) They reflect 3D structures of the world behind them. 3) They may be affected by the optical condition such as lighting or reflectance. These make recognition of natural images more difficult than that of binary patterns. As richer forms images, **depth images** and **thermal images** are often used in combination with RGB images, which are called RGB-D images or RGB-T images. While RGB-D or RGB-T images may give more cues for understanding the contents, a problem is that we now have only much fewer image data available for training, and thus the design of training schemes can be tricky to be successful. **Videos** are temporal sequences of images. Videos are similar to **time-lapse images**, but the difference is that in time-lapse images the temporal interval between two frames are so large that we can not treat them as continuous due to substantial appearance changes between frames.

Further the word "domain" is also used to refer to specific image acquisition or collection schemes within

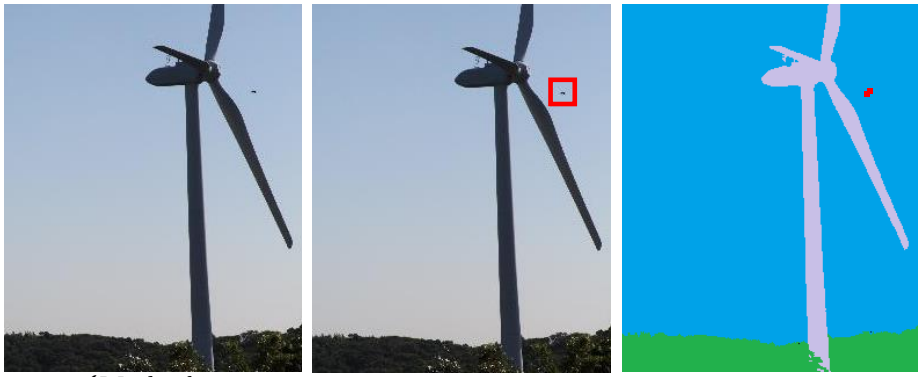


Figure2.1: Three major tasks in image recognition: classification, detection, and segmentation.

natural images. For example, **generic images** refers to natural images without any constraint of objects' pose, illumination, or backgrounds, typically collected by crawling the web. **Surveillance images** usually refers to images taken by fixed-point cameras, and thus, they have less variation in backgrounds.

Tasks While the ultimate goal of computer vision is to make machines understand the world outside of them, that "understanding" is too vague to tackle. We need to cut down pieces of visual understanding that are simple enough to evaluate whether we have solved them. Such pieces are called *tasks*. Below we briefly introduce typical tasks in visual recognition.

Classification is the most fundamental task in visual recognition that outputs binary labels $\{0,1\}^N$. An elements in the output binary vectors are interpreted as corresponding to a class, e.g., $[1,0,0]$ to a dog, $[0,1,0]$ to a cat, etc. These label-class correspondences may be arbitrary, since optimization processes during training ensure them. The dimensionality of the output is the number of assumed classes and classification with N-dimensional output is simply called N-class classification. Particularly, we refer to the case that $N = 2$ as binary classification and $N \leq 3$ as multi-class classification. Usually, we assume that each of the classes is mutually exclusive, in other words, the output vectors are one-hot. Otherwise, it is called multi-label classification. In computer vision, classification is used to describe properties of an input image as a whole, for example, whether a class of objects is presented in the image (object classification), or in what kind of landscape the image was taken (scene classification). However, classification does not consider where in the image the recognized object is. In addition, in our surveillance setting, a classifier may classify our image into "wind turbine" regardless to the appearance of birds (Fig. 2.1 a), influenced by the larger objects in the scene.

Detection is a task to output not only a class but also the location of the class in the image(Fig. 2.1 b). The most usual form to encode location information is a **bounding box**, which defines a region in an image by a rectangle parallel to the image axes. Such rectangles can be denoted by integer-valued four-dimensional vectors as

$$B = (x, y, w, h), \tag{2.2}$$

where B denotes a bounding box, (x, y) the coordinate of its left-top corner, w its width, and h its height. Further, since multiple objects can appear in a single image in many cases, usually detection outputs multiple pairs of (class, location) for an image, and a detector should decide how many boxes to output by itself. Thus, detection

outputs a variable-length list of bounding boxes and can be denoted as follows:

$$[(C_1, B_1), (C_1, B_2), ..., (C_N, B_N)] = Detect(Image), \tag{2.3}$$

where $Image$ is an input image, C_i is a class, and $B_i \in \mathcal{Z}^4$ is a bounding box. Although the output space of detection is much larger than classification's, detection still can be seen as an extension of classification. A classifier trained to classify regions to detect from regions not to detect (background) is easy to extend to a detector. For example, the algorithm gives a detector by a classifier.

Modern detectors often have additional post-processing modules to refine the detected bounding boxes. 1) bounding-box regression, 2) non-maximum suppression, and 3) context rescoring.

Algorithm 1 Object detection using a classifier

Input: Input image $Image$, a classifier $Classify$
Output: Detected bounding boxes $D = \{b_1, b_2, ..., b_N\}$

```
B ← ∅
for Box b in {all possible boxes} do
    ROI = crop(Image, b)
    if Classify(ROI) ≠ background then
        Add b into D
    end if
end for
return D
```

Segmentation is a more detailed form of localization that indicates whether every pixel in an image belongs to a class or not. A segment, also called mask can be denoted as:

$$S : P \rightarrow \{0, 1\} \tag{2.4}$$

$$p \mapsto S(p), \tag{2.5}$$

Where P denotes the set of all pixels in the input images. For a segment, we also call the set of pixels such that $S(p) == 1$ as a foreground and that $S(p) == 0$ as a background. An extension of segmentation that output classes instead of 0/1 labels is called **semantic segmentation**. Jointly conducting detection and segmentation is known to be beneficial in some cases [26–28], segmentation costs much more annotation labor than detection. Furthermore, more recent studies unified semantic segmentation and detection into **instance segmentation** [29, 29, 30]. This refers to a single task where every instance of objects needs to be detected with their own foreground masks. A drawback of segmentation is the large cost of annotation; To apply a supervised method to segmentation, we need to prepare per-pixel ground-truth labels of training data.

The above three tasks, classification, detection, and semantic segmentation are the most major generic image recognition tasks and they occupy the largest part of image-recognition literature. The counterpart of generic image recognition is **instance recognition** [1]**.** In instance recognition, algorithms collate an image that contains the target object (called a query) to a set of images that is already stored in a database (called a gallery), and extract a subset of the images that contain the same objects. Thus, instance recognition can be regarded as learning of instance-to-instance relationships, while generic image recognition can be regarded as learning of instance-to-

[1] Instance segmentation is not included in instance recognition, following the definition we adopted.

concept relationships.

Tracking can be seen as an instance-level counterpart of detection. A tracker outputs bounding boxes of an object in the same manner to a detector, but a tracker utilizes a bounding box of the object to track, which is typically given by the user, as an additional input to an image. In the most usual setting, a user defines a bounding box that indicates the object to track at the first frame of a video sequence, and then the tracker tracks the object in the following frames. Object trackers can be further categorized into two groups: single-object trackers and multi-object trackers. A single-object tracker [31, 32] only tracks a user-defined object and ignores all the other objects. Contrarily, a multiple-object tracker [33, 34] tracks all the objects within the image. To handle multiple-object tracking, the most major approach is tracking-by-detection [35, 36], which first detects all the objects in all the frames and then matches the objects to define their trajectories.

2.1.2 Hand-crafted features

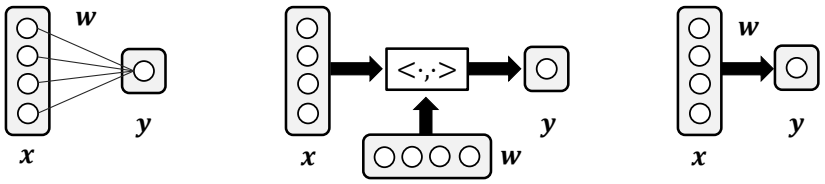
Raw sensory outputs from cameras, i.e., pixel values are for some reasons not the best for visual recognition: first, they can be changed by non-informative condition changes such as illumination or view angles. Second, they have too large dimensionality to handle, for example, approximately 20K dimensions for a $255 \times 255 \times 3$ RGB images. Third, spatially close pixels tend to have the same or close values, and they have a little information per dimension. These facts raised a question: can we engineer a transformation that makes images tractable for machines? This question drove researchers to design *hand-crafted features*, which encode information that seem to be important for visual perception. In fact, such approaches achieved quite good performances in small- to middle-scale problems, such as key-point matching [37], instance recognition [38], and early generic-image recognition [39]. However, usage of hand-crafted features in generic-image recognition has turned out to be ineffective in larger-scale problems, and is obsolete.

Below, we briefly review representative hand-crafted features. The early successful features are Haar-like [40] and Histograms of Oriented Gradients (HOG) [41], and followed by their many variants. The ideas of feature design are the use of gradients [37, 41], edges [42], robust color descriptors by self color similarity [43], multi-resolution [44], covariance and co-occurrence features [45, 46], and speed increase with binary features [47]. Modern detection methods often use multiple features in combination. Some papers found effective sets of features that work better together [48] and others focused on the methods for aggregating different types of features and channels. These excellent hand-crafted features maintain the competitive performance in pedestrian benchmarks [49] even after the appearance of deep learning [50].

2.1.3 Machine learning and probabilistic models

In visual recognition, the success of non-learning-based algorithms is largely limited due to the difficulty mentioned above. Rule-based systems work well only when the world is abstracted enough in the form of symbols. However, the problem is that the vision itself is a process of abstraction of visual worlds, and it is extremely difficult to write down the process into rules. For example, we humans can distinguish cats from dogs, but we can not describe how we do exactly. Some early attempts tried to model the abstraction via geometrical shape fitting by lines, planes, cylinders, or cones [51, 52].

Machine learning can overcome the limitations of rule-based systems by exploiting large data. One of the most well-used definitions of machine learning is to "give computers the ability to learn without being explicitly programmed" [53]. This is helpful for designing visual recognition pipelines by eliminating efforts to manage



a) Network-like illustration b) Computational-graph-like illustration c) Simplified illustration

Figure2.2: Multiple ways of illustrating neural networks. All of the shown illustrations show a fully-connected layer with 4-dimensional input and one-dimensional output. a) A network-like illustration is conventional and most often seen in the literature. b) A computational-graph-like illustration is mathematically more exact. c) A simplified illustration is visually the plainest and we use this unless there are particular reasons.

the huge number of hard-coded rules. Learning-based visual recognition can be formulated by an extension of Eqn. 2.1

$$\theta = Learn(D) \tag{2.6}$$

$$Understanding = Recognition(Image; \theta). \tag{2.7}$$

Here θ is called a **parameter** of the model. A parameter is typically a real-valued vector, and it is optimized by $Learn$ using a training dataset D.

Below, we briefly review some representative learning methods. **The nearest-neighbor method** [54] is the simplest recognition model that compares input to every training data. Whether we should categorize the nearest-neighbor method into a learning-based method is not obvious, but it still can be regarded as a learning-based method the training algorithm of which is simply storing training data in the memory. While training the nearest-neighbor method is simple, its major drawback is that its time complexity tends to be large, especially when the training data is huge. This demerit makes the method unfavorable in large-scale recognition problems, although very early recognition systems used the nearest-neighbor-based methods [55]. Nevertheless, some variations of the nearest neighbor method have been used when the application requires the one-to-one relationship between the input and a training data point, for example in information retrieval. In such cases, techniques to compress data and accelerate neighbor search [56–60] are used.

To avoid time-consuming inference, the main approaches in machine learning use a function that can be evaluated in constant time with regard to the size of training data to the training dataset D. **Linear discriminative models** [61] use the simplest functions, i.e. linear combinations of input features. A linear function can be denoted as follows:

$$\boldsymbol{y} = Linear(\boldsymbol{x}; \boldsymbol{\theta}, \boldsymbol{b}) = \boldsymbol{\theta x} + \boldsymbol{b} \tag{2.8}$$

$$= \begin{pmatrix} \theta_{11} & \theta_{12} & \dots & \theta_{1N} \\ \theta_{21} & \theta_{22} & \dots & \theta_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{M1} & \theta_{M2} & \dots & \theta_{MN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}, \tag{2.9}$$

where $x = (x_1, x_2, ..., x_N)^T$ denotes the input vector, y denotes the output vector, and the matrix θ and a vector

b denotes denotes the parameter of the layer. The parameters are decided to minimize a loss function, which represents a model's performance on the training set. There is a huge variety of what loss to use and what solver to use to minimize the loss, and each combination of the linear discriminative model and losses has a different name [61–63]. One of the most popular methods is the support vector machine [63], which used a hinge loss + margin maximization. The largest advantage of linear models is simplicity in training and testing, and they are still surprisingly effective in many datasets [64]. However, the simplicity is in exchange for their small expressive power; they can not learn patterns that are not linearly separable, for example XOR. To overcome the limitation, linear models are often used in combination with kernel tricks [65], that exploit non-linear transformation of input features in implicit manner. In image recognition, linear models that exploit pixels as features do not perform well even in relatively easy image datasets [24]. Thus, the combination of linear or kernel SVMs and hand-crafted features was the first choice for visual recognition before the advent deep learning.

Decision trees are useful to draw a non-linear separating hypersurfaces. A decision tree can be denoted as:

$$y = node_0(x) \tag{2.10}$$

$$node_i(x) = \begin{cases} y_i & (i \in terminals) \\ node_{left_i}(x) & (i \notin terminals \quad \& \quad rule_i(x) = True) \\ node_{right_i}(x) & (i \notin terminals \quad \& \quad rule_i(x) = False) \end{cases}, \tag{2.11}$$

where x denotes the input, y denotes the output, $node_i$ denotes the i-th node in the tree, $left_i$ and $right_i$ the children of the i-th node, and $rule_i(x)$ the decision policy at the i-th node. The set $terminals$ consists of the indices that indicate nodes without children, and at such nodes final decision y_i is made. In training, decision policies $rule_i(x)$ are selected on the basis of some statistical criteria for separation, such as Gini coefficient or entropy. In this formulation only binary trees are shown (all the nodes in the tree have two children), but its extension to N-ary trees is easy. A preferable property of decision trees is partial activation; only small parts of the nodes are computed in test, and thus they are fast. For construction the trees, various algorithms have been proposed, including ones utilizing statistical criteria [66, 67] to ones based on random separation [68, 69].

While training very large and deep trees is difficult, gathering decisions from relatively shallow trees is easier and gives more stable results. Thus, *ensemble learning* is important in tree-based learning. Ensembles of trees are especially called forests [68]. While early boosting algorithms were based on greedy minimization of loss functions with heuristic sample reweighting [70], more recent gradient-based loss minimization generally achieve tighter convergence and is promising [71–73]. However, in this work, we mainly use AdaBoost with greedily-learned decision trees [74], which is de facto standard in the pedestrian detection bench marks [75].

Compared with deep neural networks, decision trees are less effective when applied to raw visual data. This is because the trees do no have ability to learn feature hierarchy. Nodes in the trees learn split of given data in each node, and one of the nodes only can be related to one concept. In contrast, neural networks learn distributed representations [76], where one concept is encoded by activation patterns of multiple neurons, and one neuron is reused in multiple concepts. This makes representations learned by the neural networks hierarchical and efficient. Thus, in recent work, the decision trees are used in combinations with learned representations by neural networks [77, 78].

2.1.4 Deep learning in image recognition

Deep learning [8, 9] is a group of models that utilize multi-level representation of input signals to produce inference results. A L-layered deep function can be denoted as

$$Output = Layer_L(Layer_{L-1}(Layer_{L-2}(...Layer_1(x;\theta_1)...;\theta_{L-2});\theta_{L-1});\theta_L), \tag{2.12}$$

where x is the input and θ_i is the parameter of i-th layer. This can be also seen as a composite function made of $Layer_1(\circ;\theta_1)$, $Layer_2(\circ;\theta_2)$, ..., $Layer_L(\circ;\theta_L)$. The merit of this structure is that we can define complex functions easily by *stacking* relatively simple layers. Further, when all $Layer_l$ for $l \in [0, L]$ is differentiable, the parameters of all layers are jointly learnable with gradient-based optimizers.

Although the consensus that deep learning is necessary for visual recognition has been made as recently as 2012 after it won a large-scale competition [12], it is worth noting that some seminal works [79–81] of neural networks in the late 20th century had already incorporated architectures that can be referred to as being deep. Nevertheless, deep learning was basked in the limelight for its first time after the some background conditions were met:

- Large and relatively inexpensive computation resources became available by commoditization of graphic processing units (**GPU**s).
- Large-scale and diverse image datasets became available in the web and crowd sourcing of annotation.
- Conventional visual-recognition systems were saturated in their performance in the large-scale datasets and drastic change was desired by researchers.
- Techniques to overcome the limitations of conventional neural networks accumulated sufficiently.

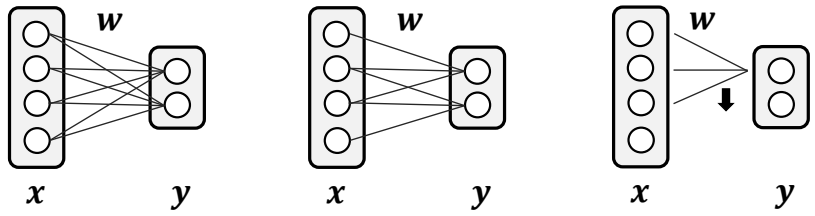
Bellow, we briefly introduce some important techniques.

Network architecture

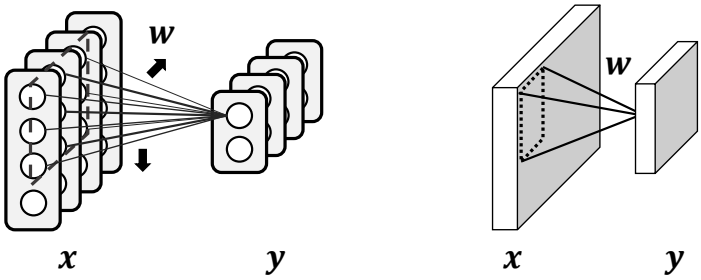
Convolutional neural networks Deep neural networks are so general grouping that there is a huge variety of network structure, and the structure is important for networks to perform a task effectively and efficiently. Methodologies to design preferable functions via network structure are called **architecture**. Specifically, efforts to define new network topology, layers, and activation functions are categorized into the architectural study. The most naive form of deep neural nets is fully-connected networks, also known as **multi-layer perceptrons**. Perceptrons are a group of neural networks that consist only of linear layers and activation functions. A linear layer parametrically represents all of the linear transformations (Eqn. 2.8). The layer is illustrated in Fig. 2.2. A perceptron with one or more hidden layers (a layer output of which is not the output of the total network, in other words, a non-final layer) is called multi-layer perceptron, and it can be regarded as the most simple form of deep neural networks.

However, naive perceptrons are not the best choice for spatially structured high-dimensional data such as images. This is because fully-connected layers are parameter-intensive. For example, they have $(O)(N^2)$-dimensional parameters when their input and output dimensionality is set equally to $M = N$. This makes fully-connected layers inefficient in images, since even a small one with the size of 64 pixel-square becomes a $64 \times 64 \times 3 = 12,888$-dimensional vector. Of course, N can be selected independently from the raw input's dimensionality, but much smaller N than input dimensionality may result in loss of information and it may be a bottleneck of the performance.

To deal the problems with fully-connected layers applied to images, a good idea is to exploit intrinsic properties



a) Fully-connected layer b) Locally-connected layer c) Convolutional layer (1D)



d) Convolutional layer (2D) e) Convolutional layer (2D): simplified illustration

Figure2.3: The concept of convolutional neural networks. a) a fully-connected layer. b) A locally-connected layer limits the connections in local regions within the kernel size (in this figure $k = 3$). c) A convolutional layer reduces the number of parameters than that in the locally-connected layer by introducing spatially shared weights, which are the single weight vector that is applied in all the locations. d and e) 2D extension of convolutional layers.

in images. Among such properties, one of the most intuitive ones is translation invariance; for example, translation of objects within images do not affect the class of that images (e.g., a dog in an image makes the image 'dog' regardless of its position). **Convolutional neural networks** (CNN) [81, 82] exploits the translation invariance of patterns. A CNN can be defined as a neural network that has at least one convolutional layer. Convolutional layers accept a feature map, that is a two-dimensional array of feature vectors as input. A convolution layer can be denoted by the following location-wise relationship:

$$\boldsymbol{y}(\boldsymbol{p}) = Convolution(\boldsymbol{x}; \boldsymbol{w})(\boldsymbol{p}) \tag{2.13}$$

$$= \sum_{dp \in [0,k]^2} \boldsymbol{x}(\boldsymbol{p} + \boldsymbol{dp})\boldsymbol{w}(\boldsymbol{dp}), \tag{2.14}$$

where $\boldsymbol{p}$ is locations in the input feature map, k denotes the size of the convolution kernels and $\boldsymbol{dp} \in [0, k]^2$ denotes pixels in the kernel. In convolutional layers, unlike, linear layers whose input was a vector, their input vector has a two-dimensional coordinate system. This operation si inspired by and can bee regarded as a linear filtering of images using $k \times k$ filters. In short, a convolution is also denoted as feature-map-to-feature-map

relationship as follows:

$$\boldsymbol{y} = Convolution(\boldsymbol{x}; \boldsymbol{w}) \tag{2.15}$$
$$= \boldsymbol{x} * \boldsymbol{w}, \tag{2.16}$$

and the operator $*$ is used to denote the convolution. The convolutions can be defined also for multi-channel feature maps, and usually neural networks exploit multi-channel convolution to extract rich information. For extending Eqn. 2.15 to multi-channel feature maps, 1) the convolutional kernel $\boldsymbol{w}$) should have channels as many as the input's, in other words, $k \times k \times N$- dimensional kernels are applicable to $W \times H \times N$-dimensional feature maps. 2) when we want M-dimensional feature maps as the outputs, we use M kernels and regard a feature map from a kernel as one of the channels in the output feature map. In other words, a convolutional layer with a $k \times k \times N \times M$-dimensional kernel transforms N-dimensional feature maps into M-dimensional ones. In this way, we can use arbitrary numbers of output channels for an input feature map.

A convolutional neural network that consists only of convolutional layers and related spatial transforming layers (pooling, upsampling, etc.) are especially called *fully-convolutional networks* (FCNs) [83]. FCNs can be regarded as CNNs from which their fully-connected layers are removed. They are efficient for providing dense pixel-wise predictions and often used in depth estimation [84, 85], optical-flow estimation [86, 87], colorization [88], and image processing [89–92].

Recurrent neural networks While CNNs are designed for efficient handling of spatial information, temporal information is also important. Recurrent neural networks (RNN) [79, 93, 94] are designed for temporal for sequential information processing. Specifically, with regard to sequential inputs $bmx_0, bmx_1, bmx_2, ...$, an RNN outputs the sequence Recurrent neural networks are a class of neural networks that can be denoted as:

$$(\boldsymbol{y}_t, \boldsymbol{h}_t) = Reccur(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}), \tag{2.17}$$

where t denotes the timestep, $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ denote input and output at the timestep t, respectively. In addition, an RNN has internal representation $\boldsymbol{h}$, which is called a hidden state. Especially, in the narrow sense, the term RNN also refer to a specific type of simple RNNs that can be denoted as follows:

$$\boldsymbol{h}_t = \text{sigmoid}(\theta_h[\boldsymbol{x}_t|\boldsymbol{h}_{t-1}] - \boldsymbol{b}_h), \tag{2.18}$$
$$\boldsymbol{y}_t = \text{sigmoid}(\theta_y[\boldsymbol{h}_t] - \boldsymbol{b}_y). \tag{2.19}$$

This type of RNNs the possible simplest form and is also called simple RNNs or Elman networks [93]. This network is also illustrated in Fig. 2.4 a. While Fig. 2.4 a shows a cyclic graph, it can be illustrated by an acyclic graph with temporal unfolding (Fig. 2.4 b) without changing its meaning.

The simple RNNs have limitations learning long-term sequences. This is due to gradient explosion and vanishment [95]. In long sequences, the RNNs does many multiplications between their parameters and activations. This makes the values of parameters and activations too large or too small, and harms training of the networks with gradient-based optimizers. For example, a linear RNN with parameter $a > 0$ would be stable only when

$a = 1$, if the sequence length is ∞ as follows:

$$a^{\infty} = \begin{cases} \infty & (a > 1) \\ 1 & (a = 1) \\ 0 & (0 <= a < 1). \end{cases} \tag{2.20}$$

The gradient explosion and vanishment can be mitigated by introducing constant error carousels (CEC) in the RNNs. The CECs are recurrent connections that propagate their hidden states to the next timestep without transformation (Fig. 2.4 c). Its role can be explained to enforce $a = 1$ in Eqn. 2.20. This idea is later generalized into skip connection [96], which has been clear to be also useful in non-recurrent but extremely deep networks.

Long short-term memories (LSTM) [97] are a recurrent network architecture that features CEC, and it is useful to model long sequences. Their structure is illustrated in Fig. 2.4 d. The network is constructed by the constant error carousel and multiple gatings on it. The LSTM can be denoted as follows:

$$\begin{aligned} i_t &= \sigma(w_{xi}x_t + w_{hi}h_{t-1} + b_i) \\ f_t &= \sigma(w_{xf}x_t + w_{hf}h_{t-1} + b_f) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(w_{xc}x_t + w_{hc} \circ h_{t-1} + b_c) \\ o_t &= \sigma(w_{xo}x_t + w_{ho}h_{t-1} + b_o) \\ h_t &= o_t \circ \tanh(c_t), \end{aligned} \tag{2.21}$$

where c_t denotes the CEC, and i_t, f_t and o_t denote the gates for controlling the flow of information, each of which is called the input gate, forget gate, and output gate. The hidden state h_t and CEC c_t are passed to the next timesteps, and h_t doubles as the output of the LSTMs. LSTMs are until now a strong baseline of long-term sequence learning. Recently more RNN-variants have been developed. Gated recurrent units (GRU) [98] re-designed the gating mechanisms and achieved superior or competitive performances against LSTMs in some datasets. Neural turing machines [99] and their variants [100, 101] explored the ways to exploit 'external memories' in addition to CEC as an 'internal memory'. The external memories were selectively updated by pointers and read-write operations emitted by neural networks, and they enabled more flexible sequence operations. However, even after the intensive architectural explorations, a recent benchmarking work showed that an LSTM, if correctly tuned, can compete against newer networks [102] in sequence-modeling tasks.

Activation functions The layers introduced above are not powerful in representing complex data by themselves, but they become effective by stacking them into a deep network. To stack linear transformations such as fully connected layers or convolutional layers into deep networks, it is important to insert non-linear layers between every linear layer. Otherwise, the network can not represent any non-linear transformations, because a composition of any two linear transformations is still a linear transformation. For such non-linear transformation, per-element transformation of the representation is sufficient in most cases. The functions used for this purpose are called **Activation functions**. The most conventional activation functions in neural networks are σ-shaped functions such as the sigmoid or hyperbolic tangent [103]. Specifically, the sigmoid function can be denoted as:

$$\text{sigmoid}(x) = \frac{1}{1 + \exp(-x)}, \tag{2.22}$$

and the hyperbolic tangent function ($tanh$) can be denoted as follows:

$$tanh(x) = 2 \quad \text{sigmoid}(x/2) - 1. \tag{2.23}$$

While the sigmoid and tanh are similar σ-shaped functions, a difference is in their range of the possible value: while $tanh(x)$ may take values over $(-1, 1)$, while $sigmoid(x)$ only takes positive values. Thus, $tanh(x)$ is zero-centered and is less likely to cause biases in output [104], while $sigmoid(x)$ is preferable when the output needs to be positive, for example, when we need to interpret the output as probability or gating.

However, a problem in the σ-shaped functions is saturation, which causes the vanishment of the gradient when the absolute value of the input is too large. Thus, in modern neural networks non-saturating functions are preferred. The most used one is the rectified linear unit (ReLU) [105], which can be denoted as:

$$ReLU(x) = \max(0, x). \tag{2.24}$$

Later more variants of ReLUs were developed [106–108].

Higher-level architecture Given layers and activation function described above, how to construct a deep network by combining them is another issue. Especially, the design of large-scale networks have begun to converge, and a few, well-designed networks are mainly used as the de facto standards. The background of this convergence seems that ImageNet-pretrained networks' wide range of applicability turns out, and perhaps re-designing and re-training large networks is too much burden for the majority of researchers.

The earliest network that achieved the large-scale success is AlexNet [12] designed for image classification. Other than its scale, the philosophy behind it is not much different from the LeNet [81], the simplest archetype of the convolutional networks. VGG-net [109] is based on a similar principle to AlexNet's, but it introduced multiple additional techniques. First, it eliminated convolution layers with large kernels, which was used in AlexNet in earlier layers. Rather VGG-net stacks multiple 3×3 convolutional layers to process larger regions in its input images. For example, a 5×5 convolutional layer and two stacked 3×3 convolutional layers samely see 5×5 local regions in the images, but a 5×5 convolution has 25-dimensional kernels but two 3 $times3$ layers have only $9 \times 2 = 18$-dimensional parameters, and thus the latter is more efficient. GoogLeNet [110] introduced a novel design based on microarchitecture, which first design microblocks that and later stack them. In other words, the earlier networks used convolutional layers as microblocks, but after GoogLeNet, more complex architecture than single layers are used. GoogLeNet's microblock was named Inception module, and it consists of concatenation of convolutional layers with various kernel sizes, which are useful for extracting diversified features from the input. ResNet [96] is based on VGG-net's macroarchitecture, but its distinguished feature is skip connection put in each microblock. The skip connections prevent gradients from vanishing and ease training of extremely deep networks ($\tilde{1}$,000 layers). We also note that the effect of skip connections is known in RNNs (CEC), and ResNet may be regarded as a non-temporal LSTM [111]. DenseNet [112] can be regarded as a variation of ResNet, where more skip connections are added to all layers densely.

Regularization and normalization

In training parametric layers, there are some difficulties in optimization. Failures of training in machine learning can be categorized into the two: underfitting and overfitting. Underfitting is a problem in training that causes In neural networks, underfitting may be caused by 1) insufficient learning capacity of networks or 2) suboptimal

solution acquired in training. While the former is easy to avoid by designing larger and stronger networks (when the computational budget is not limited), the latter may be more problematic. In gradient-based training of neural networks, gradient vanishment is one of the most often reasons of underfitting.

In contrast, a network that caused overfitting achieves small values of training objectives and thus well-performing on the training data, but does not perform so well on the test data. In contrast, if a model performs well on the test data, it is said that the model generalizes to the test data. For example, the one-nearest-neighbor method can fit almost any data, but it rarely generalizes to new test data.

Here we describe techniques to avoid under- or overfitting in neural networks. Regularizers are designed for preventing networks overfitting. One of the most popular reguralizers is dropout [113], which randomly 'drop' the neurons activations to zero during training. Originally, the effect of dropout was explained as an 'implicit ensembling', which enabled co-existing of multiple models that were differently dropped out the neurons within a network [113]. Later the dropout was analyzed and it became clear that it has a similar regularization power to L2 regularization [114]. Furthermore, it As variations of the dropout, there is more random-dropping-based regularizers [115–117]. In addition

Normalizers normalize inputs or hidden representation in networks by dividing by some denominator. This has multiple merits: 1) It avoids underfitting by fastening convergence of training process. Without normalization, layers in deep neural networks suffers inconsistency of input distributions caused by updates of upstream layers. This inconsistency is called internal covariate shift [118], and harms the progress of training. While there are various ways to calculate denominator [119–122], the most popular one is batch normalization [118], where the averages over the activations in each location ((x, y) coordinates) across channels in a minibatch are used. Furthermore, such minibatch-based normalizations also benefit in generalization by disturbing inputs of each layer with normalization within randomly selected mini-batch, which can avoid overfitting.

Application to visual recognition

While deep CNNs were first designed to image classification [12,81], their ability to learn visual representation had seemed to be preferable in many other visual tasks. Thus, since 2012 when AlexNet was invented, large research efforts was committed to re-designing CNNs for other tasks.

Detection In detection, networks need to output not only objects' class but also their location. Deep detectors can be categorized roughly into two groups: two-stage detectors and single-shot detectors. The two-stage detectors first extract **region proposals**, which are bounding boxes in images that possibly contain objects, and later classify the proposals into true objects and non-object. Contrarily, single-shot detectors directory estimate objects' bounding boxes and their confidence scores. The single-shot detectors can be also used as region-proposal generators in the two-stage pipelines. Below, we briefly review representative deep object detectors.

The possible simplest approach to extend a deep image classifier into a detector was described in Alg. 2.1.1. **Region-based CNN** (R-CNN) [123] is one of the earliest deep detectors and the most similar to the conceptual algorithm. In R-CNN, a CNN as a classifier is applied to the cropped regions. Where and how many regions to crop is determined by an external region proposal methods, which were conventionally done by non-learning-based grouping algorithms [124–126], and later replaced by deep-learning-based methods [127]. we refer to this framework as *region-wise classification* (Fig. 2.5 a). Furthermore, the CNN re-estimates bounding-box coordinates (x, y, w, h) by regression jointly to the classification, and this provides tighter boxes to the objects. However,

a major demerit of R-CNN is its speed: it requires forward computations of the full CNN per region. Thus, in large-scale generic-object detection, R-CNN is obsolete. Nevertheless, it is still a major method in pedestrian detection [128] and other surveillance tasks, because we can acquire more sparse proposals from existing other methods than in generic images Region-based CNN's largest demerit is its computational cost, which requires forward computation of the CNN as many times as the number of the region proposals, Typically candidates from a region-proposal algorithms count 30K per image.

The computational bottleneck of R-CNN can be mitigated by per-image convolutional feature extraction, which is computed once per image rather than per proposal. We refer to such pipelines that consist of 1) convolutional feature extraction and 2) per-region processing as two-stage detectors (Fig. 2.5 b). **Fast R-CNN** [129] is the first two-stage detector. Fast R-CNN first computes the convolutional feature map over the whole input image, which is shared by all region proposals. Next, per-proposal descriptors are extracted from the feature map with **region-of-interest (ROI) pooling** layer. In the ROI-pooling layer, the proposal bounding boxes are resized and projected to the feature map. The projected boxes are spatially divided into a fixed number of bins (e.g., 3×3). Within each bin, the activation values are reduced with max or average pooling and as the result we can acquire fixed-dimensional region descriptors (e.g., $3 \times 3 \times$ the number of channels). Finally, the descriptors are fed into fully-connected layers, and they perform classification and bounding-box regression. In ROI pooling, a non-trivial point is whether we may resize or resample feature maps as well as RGB images. However, an earlier work in object detection analyzed the effect of resizing in HOG-like features and show that competitive performance can be achieved with such feature resizing. **Faster R-CNN** [130] further speeded up the Fast R-CNN by eliminating the necessity of external region proposal algorithms. Faster R-CNN computes proposal boxes from its own convolutional feature maps. Particularly, the sub-network that does this proposal regression is called region-proposal network (RPN). The RPN is a fully convolutional network that outputs bounding boxes' coordinates and their objectness scores, and it works as a region-proposal generator that is trainable and runs faster than the bottom-up methods [124, 125].

The two-stage detectors still have computational bottlenecks in per-region processing, where fully-connected or other computationally heavy modules per region proposal are needed. Single-shot detectors remove ROI pooling and all latter modules and aim to directly predict bounding boxes from convolutional feature maps. A typical method for bounding-box prediction from the feature maps is dense regression: the final layer of the single-shot detector network has $(4+N)$ channels, where N denotes the number of class, and they estimate (x, y, w, h) and the class probabilities per pixel in the feature map. The most popular single-shot detectors include YOLO [131–133] and SSD [134]. The single-shot detectors are the fastest among the a) – c) models, but tend to perform worse than two-stage detectors with similar networks due to the lack of per-region processing. and there exists the trade-off between accuracy and speed [135].

Roughly following the above frameworks, there have been more techniques to improve deep-learning-based generic-object detection. In CNNs, the output feature maps where detection performed are excessively down-sampled ($\$$times16 $\$times$ 32), which results in large spatial quantization. Feature upsampling or upconvolution [136–138] mitigate this problem and enable better detection of relatively small objects. We briefly refer to such techniques. Receptive fields (RFs) refer to the regions in input images where a neuron sees. In CNNs, RFs are usually a local region within a rectangle that is defined by the sizes and numbers of convolutional and pooling layers in the network. Learned filters' weights also matter the effective RFs [139]. For detecting objects with various sizes and poses, introducing multiple RFs [140, 141] or adaptive RFs [142–144] is helpful. Hard negatives

refer to negative samples that are visually difficult to distinguish from true detection targets, which causes misdetection. Actively mining such hard example results in better convergences of the training and improves detection performance [145–147]. Cascading multiple networks [148] has a similar effect, by later stages in the cascade can focus on the hard samples.

In parallel to the discussion over the algorithmic framework, implementations of large-scale detection networks have been developed. MegDet [149] is a detector Megvii [150], and its technical feature is better convergence of training by cross-GPU batch normalization. PFDet [151] is a detector used in the Google AI Open Image competition [152] by Preferred Networks (PFN) [153], Japan. While putting the basis on the feature pyramid network [138], it also introduced some novel techniques such as cosine annealing [154], co-occurrence loss, and expert models for rare classes. PFN announced that a part of the detection technologies will be incorporated in ChainerCV [155], the open-source computer-vision library.

Tracking Recent studies intensively examined CNNs and RNNs for tracking. CNN-based trackers learn convolutional layers to acquire rich visual representation, but they do not exploit multi-frame clues. Their localization strategies are diverse. Classification-based approaches [156] involve classifying densely sampled candidates into target and non-target regions. This approach yields high-quality localization but the computation is considerably slower than in real time because it needs online retraining. Similarity-learning-based approaches [157] also handle densely sampled patches but avoid online training by replacing the classifier by learned similarity, and are thus faster. Correlation-based localization is also a faster alternative to region-classification-based approaches, which compute the cross-correlation between convolutional representations of a template and frames [158, 159], and has the merit of allowing for the interpretation of correlation heat maps.

Combining CNNs with recurrent nets [79, 97], which efficiently handle temporal structures in sequences They have been used for tracking [160–163]. However, most utilize separate convolutional and recurrent layers, and have a fully connected recurrent layer, which may lead to a loss of spatial information. Thus, currently recurrent trackers do not perform as well as the best single-frame convolutional trackers in generic benchmarks. One study used ConvLSTM with simulated robotic sensors for handling occlusion [164].

Joint detection and tracking The relationship between object detection and tracking is a long-term problem in itself; before the advent of deep learning, it had only been explored with classical tools. In the track–learn–detection (TLD) framework [165], a trained detector enables long-term tracking by re-initializing trackers after temporal disappearance of objects. Andriluka et al. uses a single-frame part-based detector and shallow unsupervised learning based on temporal consistency [166]. Tracking by associating detected bounding boxes [167] is another popular approach. However, in this framework, recovering undetected objects is challenging because tracking is more akin to post-processing following detection than to joint detection and tracking.

2.2 Robust image recognition

How to make image recognition robust?—This is a long-standing question without clear answers. In fact, especially after deep learning, a large part of research resources are committed in benchmark racing, where true robustness of the methods that is needed outside of the benchmark may be overlooked. However, there had been some research efforts around robustness in the pre- and post-deep-learning eras. We categorize the approaches toward robustness into two parts: model-level and learning-level approaches.

2.2.1 Model-level approaches

Model-level approaches try to remove the source of illness within the recognition function by modifying the models used to define the function. The most typical direction in this approach is augmenting input modality. For example, using depth images or thermal images to mitigate visual ambiguity can be categorized in model-level approaches. Other than augmenting input, we also include the approaches to exploit output structures here. Below, we briefly review existing model-level approaches toward robustness.

Additional modality When available visual information is limited, a simple idea is to add other sensors that work complementarily. While it is promising when a proper type of sensors has been commoditized, in general, additional sensors increase the total cost to the systems. Yet, there have been research efforts to boost vision by additional modality from richer sensors. RGB-D image recognition is the second most usual setting in image-recognition community after RGB recognition. Especially in application to robotics it is important because robots' manipulation need distance data to objects and they are often equipped with depth sensors [168]. Other than adding depth sensors, it is also possible to estimate depths from two RGB cameras exploiting binocular disparity. This technique is called stereo matching [169–172]. However, in wide-area surveillance, it is often difficult to set two cameras in meaningfully distant locations due to land acquisition.

RGB-T image recognition has been less studied but promising research area. Its strength is availability in night time, and it is especially useful in detecting heat-source objects such as animals and humans. Already a RGB-T-bases or hyperspectral-image-based pedestrian detection dataset has been publicly available [173–175].

Occlusion handling Because occlusion is common in 3D visual worlds, robustness against it is one of the most important ones. However, as an exception, occlusion does not often happen in airspace surveillance for birds due to the sparse appearance of objects in the air and we did not put a priority in the occlusion handling. Occlusion is a more serious problem road scenes, especially when crowded [176]. The early studies for occlusion handling include designing features [42,48] or matching algorithms [177,178] that are robust to the occlusion. More recent studies try to handle occlusion via 'deeper' understanding and reasoning including layer decomposition [179] or multi-view fusion [180]. Such methods often exploit bottom-up segmentation [127] to model partially occluded subregions in objects. Occlusion boundary detection [181, 182] is tractable in supervised learning with deep learning by using similar networks to ones in edge detection [183,184] or semantic boundary detection [185,186], and it should be useful in occlusion handling.

Context In visual recognition, the context can be defined as information about an object that is available from combinations of the object and its surroundings or the whole scene, other than the object itself. Humans often rely on contexts where the objects were put, especially for recognition of objects [187]. The visual example in Fig. 2.6 shows that nearly the same patches can be understood as a car or a pole in the load, or a bottle or a notebook on the table. In algorithms, contexts can be exploited in data structures that explicitly models such object-surrounding relationships, for example, as graphs [188, 189]. In inference with the graphs, graph-based optimization methods such as MRF [190] and CRF [191] took important roles [192, 193]. However, even without such explicit modeling, CNNs may use contexts in an implicit manner. Since CNNs can automatically learn hierarchical visual representations, contexts can be incorporated in the learned block-box representations. Also, network architectures such as spatial pyramids [140] and attention [194, 195] can be considered as enhancers of

the implicit context learning. Nevertheless, combinations of probabilistic-graph-based inference and CNN-based representation learning have attracted many researchers. For examples, fusion of a CNN and a CRF [196] or MRF [197] were examined in semantic segmentation. In our task of bird surveillance, it seems that the roles of the contexts are limited because birds and non-birds samely appear in the sky and it does not provides rich cues.

Motion In this thesis, we consider how to effectively exploit motion information. In video-based applications to tasks such as bird surveillance or pedestrian detection in automatic vehicles, use of motion is natural for pedestrian detection. Several studies involved motion in detection with optical flow [43,198,199], multi-frame features [200], temporal differencing [201,202], and detection by tracking [166]. The most popular among the scoreboard leaders in the benchmark [75] is SDt [2], which can remove camera-centric and object-centric motions by warping the frames with coarse optical flow and detect informative deformation of objects in fine-scale by time differencing.

There are more hand-crafted and deep motion features in other video-based tasks, such as video segmentation, activity detection, motion analysis, and event detection. Recent perspectives on the evolution of deep motion features with respect to hand-crafted features can be found *e.g.*, [203] and [204]. In short, rather than using CNNs over spatio-temporal space, as in multi-frame CNNs [205,206], it is more successful to use separate CNNs for spatial and temporal (optical-flow) features [207]. Similar two-stream architectures have been introduced in more recent studies [203,204,208,209].

Generic object detection in videos also draws attention recently, as the largest competition in image recognition (ILSVRC2015 [210]) has started a competition of this task. Currently, most methods in the competition adopt single-frame detection and tracking of detected objects, rather than utilizing motion features. Nevertheless, novel techniques are introduced in this dataset, including flow-based feature propagation [211, 212] and joint tracking [213]; however, they are for minimizing motion to maintain temporal consistency of object classes, but not for exploiting motion as a clue by itself. In other challenging domains such as bird surveillance, recurrent-net-based detectors have been examined [214, 215], but recurrent nets can be difficult to train [216] especially in complex environments such as road scenes, and the careful design is needed. Thus, in this thesis, we first apply simpler two-stream detection network in Chapter 4 and later extend it to recurrent-network-based network in chapter 5.

2.2.2 Learning-level approaches

Learning-level approaches exploit training methodologies for given models that are more sophisticated than simple supervised or unsupervised learning. These approaches are more suitable to achieve robustness against incomplete or noisy training data. In addition, most methods of the learning-level approaches are about how to train a model, and flexible to variations of models. Especially, in deep neural networks, due to simplicity in their training procedure, learning-level methods are often applicable to a large part of network variations regardless of their input modality, depths, or architecture. Thus, it is valuable to jointly consider model-level and learning-level methods to improve the robustness of a recognition system, Below, we briefly review existing learning-level approaches toward robustness.

Semi-supervised learning Semi-supervised learning is a form of supervised learning that simultaneously exploit unlabelled data as well we labeled data, while purely supervised learning only utilizes the latter. Usually, in supervised learning, more powerful a model is, more unstable it is when the amount of available data is small. Semi-supervised learning can make models stable and robust against the small amount of training data. Typically,

semi-supervised learning is performed by propagating labels from the labeled data to unlabeled data, and pull decision boundaries away from the unlabeled data points as well as the labeled. The most straightforward way to perform semi-supervised learning is self-training [217]. The self-training is a heuristic to repeat the following procedures: 1) Train a classifier on the given labeled samples. 2) Classify the unlabeled data with the trained classifier. 3) Label the unlabeled data with the classifier's prediction, and add a part of the data that are confidently classified to the labeled data. 4) Repeat 1) – 4). While this is very simple, it is applicable to almost all classifiers and was actually applied to visual object detection [218]. However, with complex data distributions, self-training often misclassifies the unlabeled data and falls into suboptimal solutions. Modern semi-supervised learning methods include graph-based label propagation [219], label interpolation by generative models [220, 221], and distributional smoothing [222, 223] that penalizes steep changes of predictions in the feature spaces. However, in some application, collecting large amount of unlabeled data itself is difficult. For example, we consider detectors' generalizability to unknown environments, but even unlabeled data is not available from 'unknown' environments. While semi-supervised learning may be helpful also in surveillance settings to avoid laborious annotation, we put it out of this thesis scope.

Domain adaptation Domain adaptation [224] is a group of learning methods that aims to overcome domain gaps. In domain adaptation, the training data comes from two different domains: a source domain and a target domain. Among domain-adaptation settings, unsupervised domain adaptation (UDA) [225] is especially important because it is applicable in the cases that we have no labeled target-domain data. 'Unsupervised' in UDA means that there are labeled data only in no labeled data in the target domain, and thus, domain adaptation can be regards a variation of semi-supervised learning that additionally gives models robustness against domain gaps. Reconstruction-classification learning has been clear to be also useful in UDA settings early [226], while we later try to extend it to open-set settings. However, later distribution-matching-based methods that the distributional distance between source and target data points closer in the representation spaces became the mainstream in UDA. Such methods are implemented with self- or co-training [227], backpropagation [228] or adversarial learning [229] with DNNs.

Outlier detection Historically outlier (also called anomaly or novelty) detection was a separate area from classification. While supervised outlier detection can be regarded as an application of supervised classification, unsupervised outlier detection is especially important in real-world tasks because in most cases collecting training data for 'anomaly' patterns is difficult. Outlier detectors are useful to make systems robust against unknown patterns that may appear in the real world. However, unsupervised outlier detectors by themselves have no discriminative power within known classes, and can not be applied to the surveillance tasks where rejecting known negative samples (e.g., backgrounds) is also important. Some of the generic methods for anomaly detection are one-class extension of discriminative models such as one-class SVM (OCSVM) and support vector data description (SVDD) [230–232] or isolation forests [233], generative models such as Gaussian mixture models [234], and subspace methods [235]. However, most of the recent anomaly-detection literature focuses on incorporating domain knowledge aspecific to the task at hand, such as cues from videos [236, 237], and they cannot be used to build generic-purpose open-set classifiers.

Deep nets have also been examined for outlier detection. The deep approaches mainly use autoencoders trained in an unsupervised manner [238], in combination with GMM [239], clustering [240], or one-class learning [241]. Generative adversarial nets [242] acan be used for outlier detection [243] by using their reconstruction errors and

Table2.1: Each Chapter's relationships to the existing work.

Chapter	Domain	Task	Model
3.	Bird surveillance	Detection and classification	Hand-crafted features and CNN
4	Bird surveillance and on-board	Detection	CNN
5	Bird and UAV surveillance	Detection and tracking	CNN + RNN
6	Generic image and bird surveillance	Open-set detection and classification	CNN

discriminators' decisions. This usage is different from ours that utilizes latent representations. However, in outlier detection, deep nets are not always the absolute winners unlike in supervised learning, because nets need to be trained in an unsupervised manner and are less effective because of that.

Some studies use anetworks trained in a supervised manner to detect anomalies that are not from the distributions of training data [244, 245]. However, their methods cannot be simply extended to open-set classifiers because cvprthey use input preprocessing, afor example, adversarial perturbation [246], cvprand this operation may adegrade known-class classification.

Open-set recognition Compared with closed-set classification, which has been investigated for decades [61, 63, 70], open-set classification has been surprisingly overlooked. The few studies on this topic mostly utilized either linear, kernel, or nearest-neighbor models. For example, Weibull-calibrated SVM [247] considers a distribution of decision scores for unknown detection. Center-based similarity space models [248] represent data by their similarity to class centroids ain order to tighten the distributions of positive data. Extreme value machines [249] model class-inclusion probabilities using aan extreme-value-theory-based density function. Open-set nearest neighbor methods [250] utilizes the distance ratio to the nearest and second nearest classes. Among them, sparse-representation-based open-set recognition [251] shares the idea of reconstruction-based representation learning with ours. The difference is in that we consider deep representation learning, while [251] uses a single-layer linear representation. These models cannot be applied to large-scale raw data without feature engineering.

The origin of deep open-set classifiers was ain 2016 [252], and few deep open-set classifiers have been reported since then. G-Openmax [253], a direct extension of Openmax, trains networks with synthesized *unknown* data by using generative models. However, it cannot be applied to natural images other than hand-written characters due to athe difficulty of generative modeling. DOC (deep open classifier) [254], which is designed for document classification, enables end-to-end training by eliminating outlier detectors outside networks and using sigmoid activations in the networks for performing joint classification and outlier detection. Its drawback is that the sigmoids do not have the *compact abating property* [247]; namely, they may be activated by an infinitely distant input from all of the training data, and thus its open space risk is not bounded.

Finally, Table 2.1 summarizes the usages of the above-mentioned technologies in each Chapter of this thesis.

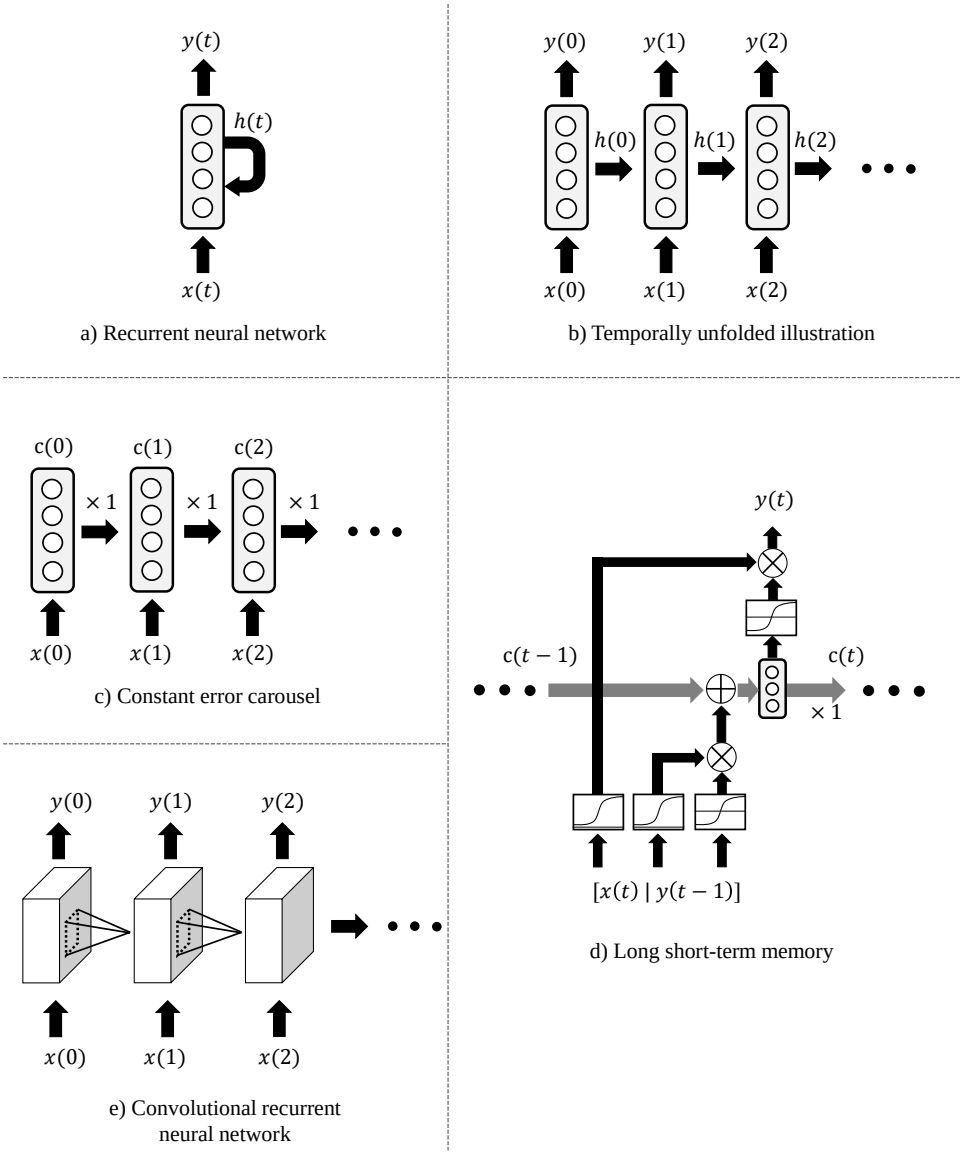


Figure2.4: Illustration of recurrent neural networks. a) Simple RNN in a recurrent form. b) The same RNN in a temporally unfolded form. This is equivalent to a) but it can be regarded as a feed-forward network with this unfolding. c) Constant error carousel. d) Long short-term memory, a recurrent neural network constructed by the constant error carousel and multiple gatings on it. e) Convolutional recurrent neural network.

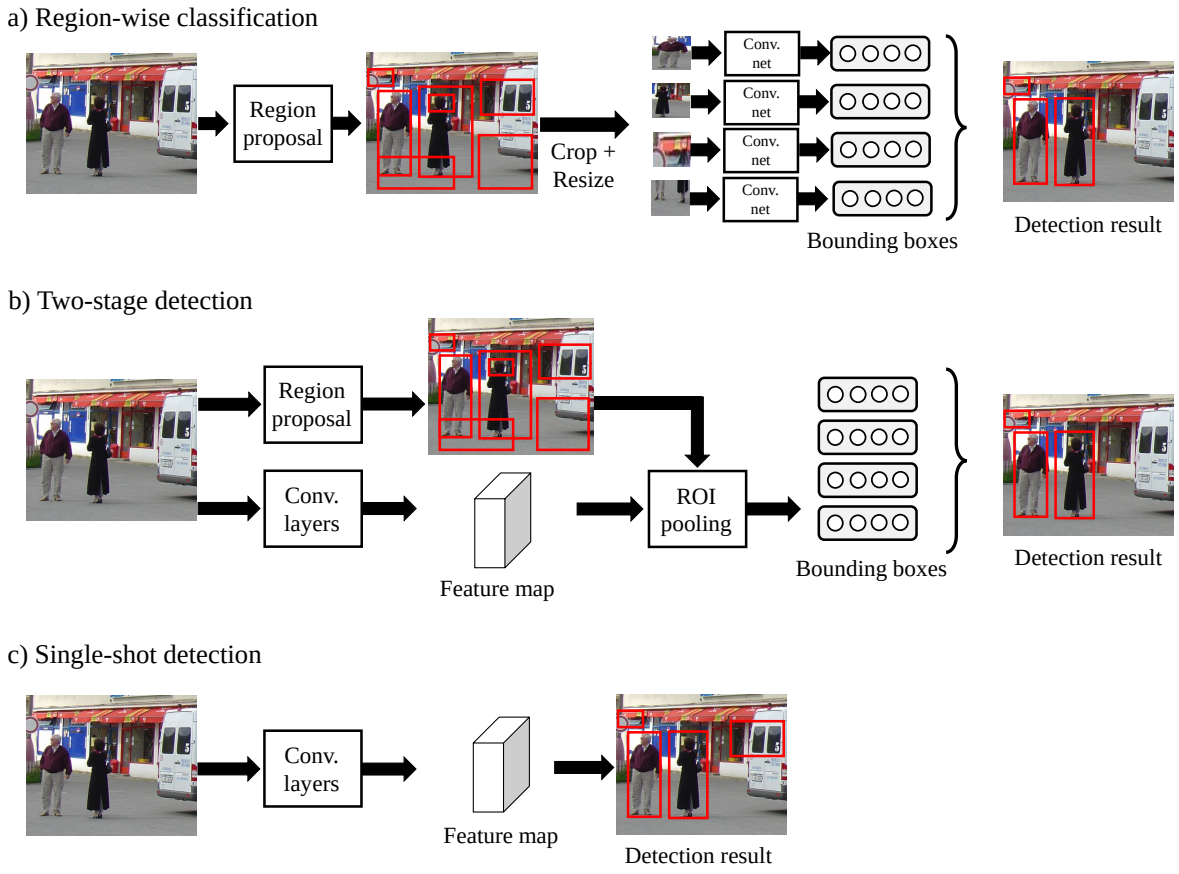


Figure2.5: CNNs for detection.

Figure2.6: Importance of context information in humans' visual recognition. Nearly the same patches can be understood as a car or a pole in the load, or a bottle or a notebook on the table. The image is from [1].

3

Wide-area bird surveillance: Data construction and analysis

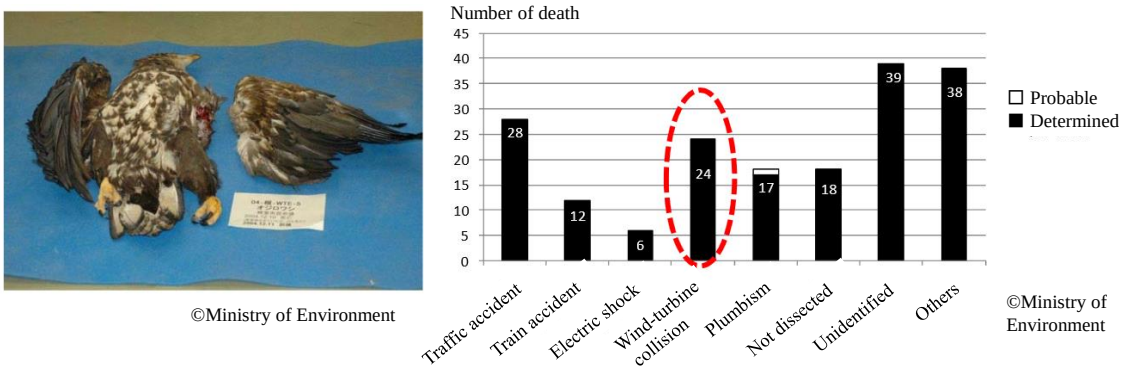


Figure3.1: Major causes of deaths of white-tailed eagles (*Haliaeetus albicilla*) in Hokkaido, Japan. Preventing the death is the motivation for bird-monitoring systems. The picture was made by Ministry of Environment, Japan and translated by the author.

3.1 Introduction

Wind energy has been seen as an environmentally friendly way to generate power and balance the need for protecting the environment with the demand for energy. However, as demand for wind energy grows rapidly around the world, the environmental impact of wind farms themselves has become an issue [255–257]. One of the primary concerns is the increase in bird mortality caused by collisions with blades, loss of nesting and feeding grounds, and interception on migratory routes [255, 258, 259]. Hundreds of bird fatalities have been reported annually at several sites [255]. Automatic bird detectors have thus drawn attention in the wind energy industry [260]. The primary reason for the attention is that many countries have regulations for environmental impact assessments during the establishment and operation of wind farms [261, 262]. These assessments require operators to collect sufficient data on the surrounding environment and estimate ecological risks posed by the farm [263]. Bird monitoring is an expensive and laborious task when it is carried out manually [264]. Here, automation can lower the cost, enable long-term monitoring, and lead to higher accuracy and reproducibility. In addition, automatic bird detectors can work with systems that decelerate blades or sound an alarm when birds approach [260, 265]. Such systems may alleviate the environmental impact, shorten the time needed for the risk assessment survey, and help to facilitate the construction of wind farms.

Performing detection and classification of birds, however, is not a trivial task for machines [266, 267]. Yet, image-based detection remains one of the promising approaches [260, 264, 268], as the information provided by visual detection is rich and detailed. Image-based detection can be complementary to radar-based detection,16-20 which is practical for nighttime monitoring.19,20 Image recognition has flourished in the last decade, driven by the progress in machine learning and the development of larger and larger datasets for training. Datasets, ie, pairs of inputs and desirable outputs, are crucial for building machine-learning algorithms. Furthermore, having access to the same datasets allows researchers to share the same goal and compare methods in the same manner, and it has advanced the fields of handwriting recognition [81], face and pedestrian detection [75,269], and generic image classification [11,210,270,271]. In addition, it has produced robust features [37,40,41], good classifiers [63,70], and new image structures [39,272]. The biggest advances in recent times have been the development of web-scale general image datasets with tens of millions of images [210] and deep neural networks trained on them [12], whose strength is in adaptive learning of features and classifiers during training. In analogy to this history of

computer vision and machine learning, a clean, detailed, and realistic dataset is also required for automatic bird detection and classification.

This section describes the construction of the first image dataset that is of practical value for recognition of birds around wind farms. The dataset is based on time-lapse images captured at a wind farm in Kinki, Japan. Each bird image is annotated by experts with a bounding box and a tree-structured label indicating its species, e.g., “bird-hawk-black kite”. The dataset contains over 60,000 annotated bounding boxes of birds and 6000 annotated bounding boxes of non-birds. It consists of 32,000 images of 5616×3744 resolution, and the total dataset size is over 100 gigabytes. Our dataset is unique and practical, because the birds tend to appear at low resolution within high-resolution images. Such a large resolution difference comes from the need to cover a wide field of view in order to assess the distribution of birds in a wide area and to notice their approach well ahead of time. As shown in Figure 1.2, the actual appearance of birds is significantly different from those in generic datasets [273–275], on which most computer vision methods are designed and experimented. Our experimental results on the constructed dataset reveal the accuracy, precision, and recall of the state-of-the-art computer vision methods in practical environments for wild bird monitoring, which have remained uncertain until now. We evaluated various recognition methods exploiting hand-designed image features and deep learning, as the performance of learning-based methods highly depends on the properties of the dataset, such as the image resolution, number of training samples, and visual similarity between categories. In fact, because of the large visual difference between images in generic object detection competitions [210, 276], and those in our dataset, the methods need to be re-examined for a realistic wind farm setting. Although several computer vision researchers have started focusing on bird detection and classification [274, 275], they have not considered this actual situation. Our results also reveal whether a simpler learning algorithm that is easy to train on a common central processing unit (CPU) suffices, or whether a more powerful deep learning method that makes a massive number of GPU (graphics processing unit) computations is necessary. This determination is important for efficient design of a practical monitoring system. Our results show that shallow learning works as well at bird detection as state-of-the-art deep learning. However, deep learning achieves better inter-dataset generalization in detection when it is applied to data acquired at different locations. Species classification is a harder problem, and deep learning outperforms shallow learning combined with various hand-designed features with a sufficient margin. Our dataset and codes for the experiments are publicly available at http://bird.nae-lab.org/dataset.

The contribution of this chapter is 3-fold. First, it provides the first practical image dataset for the task of bird recognition at wind farms. Among the bird image datasets [274, 275] for image recognition, ours is unique in that it is based on images taken at a wind farm, where a bird monitoring system is actually needed. Analysis of this data provides insights for wild bird recognition, ie, on the low-resolution image properties that indicate birds and the existence of hard negatives such as insects and airplanes. Second, the dataset can be used to evaluate various established image recognition methods for bird monitoring and reveal their actual performance. By applying image recognition methods to our dataset, we concretely assess their performance, which will be useful for designing actual systems. These results are valuable because the performance of bird detectors has been hard to assess due to the lack of available benchmarks. Our results indicate that a simple AdaBoost-based detector works as well as a deep-learning-based one in classifying birds and other objects in our dataset, but deep learning has an advantage in generalizability. Third, our study provides a state-of-the-art image recognition method from the research field of computer vision for bird monitoring at wind farms. Deep neural networks, especially convolutional neural networks (CNN), are the main driving force behind the recent advancements in image recognition. In our evaluation,

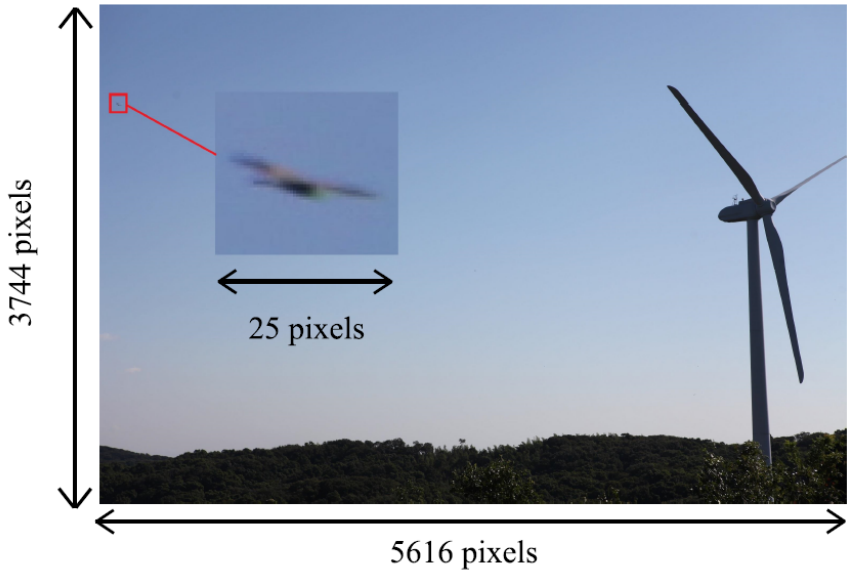


Figure3.2: A typical scene captured with our telephoto setup and stored in the database. Although the resolution of the images is as large as 5616 × 3744 pixels, the birds look small.

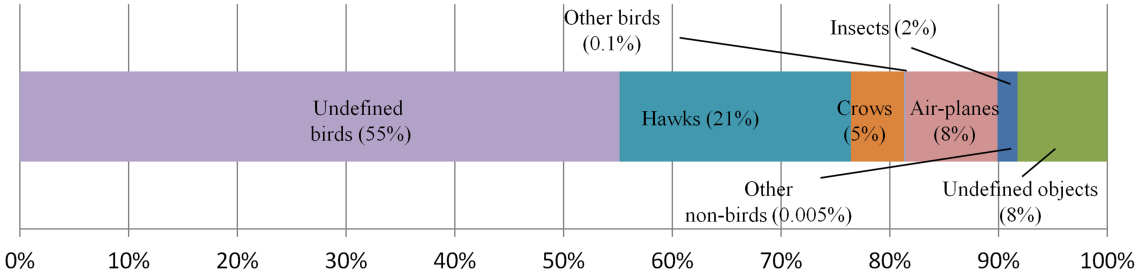


Figure3.3: Proportions of categories of found objects. Hawks were the most frequently observed, crows second most.

a CNN outperformed other methods at bird species classification and showed the possibility that this task can be automatically performed, something that existing bird detectors are not capable of [260, 265].

3.2 Dataset overview

3.2.1 Statistics

The dataset consists of images taken for three days, 10,814 images per day, and 32,442 images in total. The frame rate was 0.5 fps, because of the large amount of data and slow data transfer speed. Image variances other than birds include movements of clouds, the spinning blades of the wind turbine, shaking of nearby bushes by the wind, and illumination changes. Such variances pose a challenge when we try to detect birds from image differences.

Figure 3.5 shows the categories and their proportions. Hawks are the most frequent, with crows being second

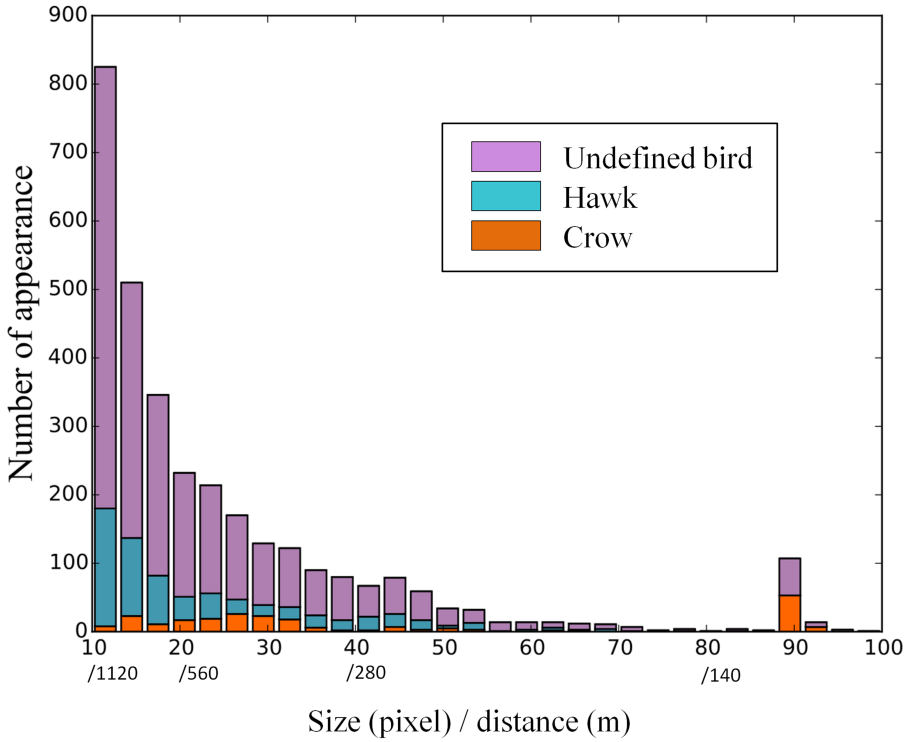


Figure3.4: Size distribution of birds in the dataset. It also shows distances to birds from the camera corresponding to each size. Distances are calculated assuming that the bird size is 1 m.

among specified birds; 30% and 5is large but reasonable because small images of birds are more frequent than larger ones, and smaller ones are often difficult to specify. Such unspecified birds are, however, still useful for distinguishing birds from bird-like patterns and for identifying target species amongst the other birds. Hence, both specified and unspecified birds were utilized in the experiments described in Section 4. Other birds include falcons, gulls, meadow buntings, sparrows, and swallows. Their numbers of appearances are small.

Figure 3.4 shows the size distribution of bird categories, namely undefined birds, hawks, and crows in the images. The bird species cannot be distinguished on the basis of their apparent size. The proportion of specified birds is smaller when the size is less than 15 pixels, while around a third of all found birds are specified when the size is larger than 20 pixels. Crows smaller than 25 pixels appeared less often, while smaller hawks appeared more often, seemingly because hawks are more likely to fly high.

3.2.2 Examples

Figure 3.5 shows examples of birds found by the users. Some images are relatively clear, and thus, they can be specified in detail. Even some of the not-so-clear images are specified in detail. For example, the eastern marsh harriers in Figure 3.5 are not so clear. These birds, however, could be identified by their actions. The 3 images are a sequence of a single individual, and it kept a V-pose while flying during the sequence. This is a characteristic feature of eastern marsh harriers, and it made it possible to specify the species of the individual.

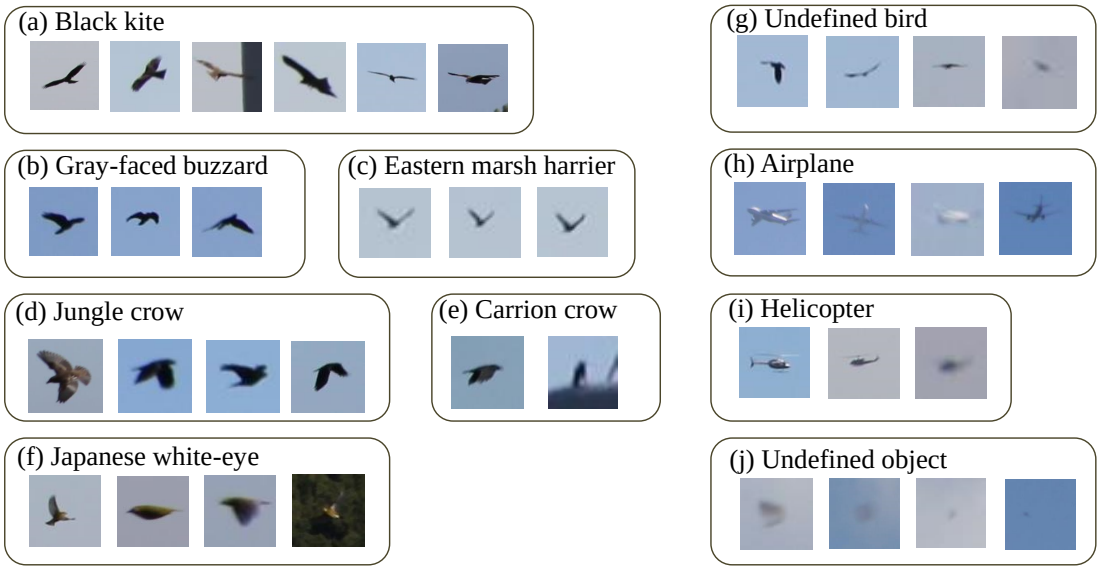


Figure3.5: Examples of found birds and other objects. The images have been resized for visualization.

Table3.1: Properties of the existing and our datasets for various object detection tasks.

Name	Target	#Classes	#Images	#Boxes	Year
FDDB [277]	Face	1	2,845	5,171	2010
CUB-2011 [274]	Bird	200	11,788	11,788	2011
Caltech Pedestrian [75]	Pedestrian	1	∼250,000	∼35,000	2012
PASCAL VOC [273]	Generic object	20	11,540	31,561	2012
UAV [278]	UAV	1	5,800	∼8,000	2017
iNaturalist [279]	Animals and plants	5,089	579,184	561,767	2018
Cattle [280]	Cow	1	656	1,886	2018
Ours	Bird	12	32,442	32,000	2015

3.2.3 Comparisons with the existing datasets

While our focus of data collection is on wild birds, how are the different from others as a detection task? We summarize other existing detection datasets and compare them with ours. Table 3.1 summarizes properties of the recent representative datasets in various detection tasks. Among task-oriented single-class datasets [75, 274, 277], ours matches to the popular ones in the term of scale. An exception is iNaturalist [279], a very recent large-scale dataset that focuses on the wildlife, similarly to ours.

Further, among the datasets, we computed size distributions of typical single-class object detection datasets, namely, Face Detection Datasets and Benchmarks (FDDB), Caltech Pedestrian, and our bird dataset in Fig. 3.6. The reason that the small objects become important comes from the scene geometry, as shown in Fig. 3.7. In usual indoor scenes of face detection (Fig. 3.7 a) or road scenes of pedestrian detection (Fig. 3.7 b), the monitoring spaces is limited by walls and no or less objects become visible in the far distances. In bird surveillance, we need to monitor open spaces where wider areas are visible in larger distances in the view angles. This causes more appearance of smaller objects in the scenes. We also note that we can remove small objects by filtering the

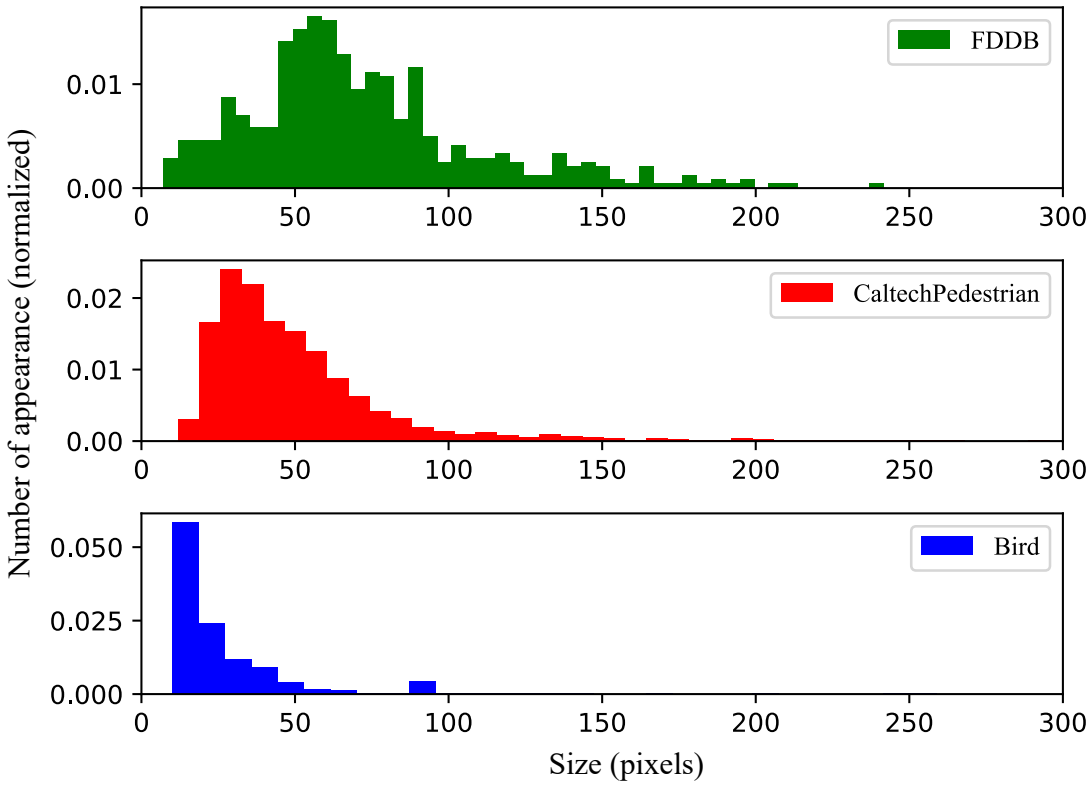


Figure3.6: Size distributions of typical single-class object detection datasets, namely, Face Detection Datasets and Benchmarks (FDDB), Caltech Pedestrian, and our bird dataset. Ours specializes in small-size objects of which detection is challenging.

detected regions using size thresholding, and by this, detectors causes neither true detection nor misdetection due to small objects. However, this results loss of many objects in the scene and decrease the value of surveillance setups, and thus detectors that can detect even small objects are preferable.

3.2.4 Extension to video-based dataset

While the firstly constructed Kinki dataset was by time-lapse imaging with 0.5 fps due to the high resolution and large capacity, around 2016 we got access to a 4K video camera that just had started to commoditized. Exploiting it, we remake a video-based bird detection dataset with similar protocol to the Kinki dataset. The imaging setup and example frame bu the setup is shown in Fig. 3.8. We refer to this dataset Tomamae. In Tomamae, the frame resolution is slightly smaller than in Kinki (3840×2160), but its frame rate is 30 fps and much higher than 0.5 in Kinki. We later use this dataset to develop video-based detectors. (Chapter 4 and 5).

3.3 Construction of the dataset

3.3.1 Image capturing

We follow the design of image-capturing system for birds in the distance shown in [281, 282]. The setup consisted of a digital still camera (Canon EOS Mark II 5D) controlled by a laptop and equipped with a telephoto

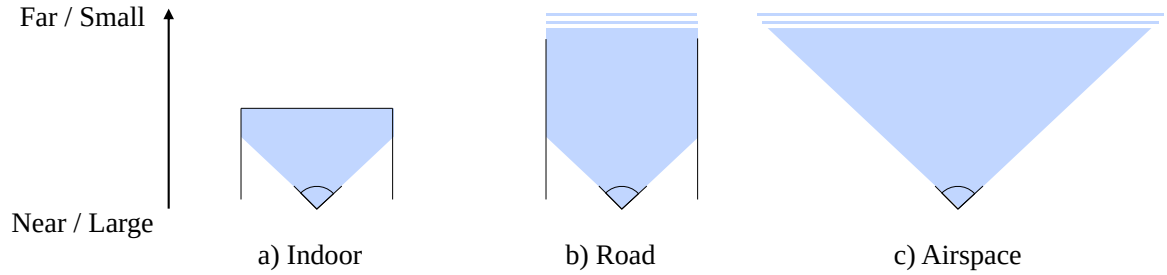


Figure3.7: Monitoring spaces in different applications. a) In indoor applications, the depths of monitoring visible spaces are limited by the walls. b) In load-scene applications, the depths is ideally unlimited. However, in usual urban environments, the both sides are occluded by buildings and this makes far regions smaller in observed images due to the perspective transformation. c) In wide-area surveillance, there is no occlusions (except the wind turbine). In this situation, far and smaller objects appear more often, which makes the low-resolution of targets the dominant difficulty in detection.

Figure3.8: Our setup for the video-based dataset and an example frame.

lens (Canon EF70-200 mm F4 L USM). The resolution of the sensor was 5616 × 3744 pixels, the focal length of the lens was set to 70 mm, and the field of view was 27° × 19°. In the images shot with this system, a bird with a 1-m wingspan 580m away, ie, the distance between the cameras location and the wind turbine, would cover an area of 20 pixels. Examples of captured images are shown in Figure 3.2. The images include those of a 2-MW wind turbine that is 80 m in diameter and located in the Kinki region of western Japan. The images were recorded near the wind turbine for 3 days. The capture system took a picture every 2 seconds for 7 hours from 9:00 to 16:00.

3.3.2 Labeling format

Each bird in the dataset is enclosed by a bounding box labeled with its species by experts. The labeling format is similar to those of other detection datasets such as the Caltech Pedestrian Detection Benchmark,39 which includes bounding boxes on time- series images. In addition, ours has fine-grained category annotations on each bounding box. Negative samples of other flying objects such as planes and bugs are also labeled. For annotating the categories of bird, we designed a tree- structured list of categories so that an expert can annotate the bounding box with labels consisting of the names listed in the tree. The names of the kinds of birds in the list were selected based on the results of a preparatory field survey. The granularity of the label can be selected depending on how clear the image of the bird is. For example, when a black kite appears, we may categorize it, depending on clarity,

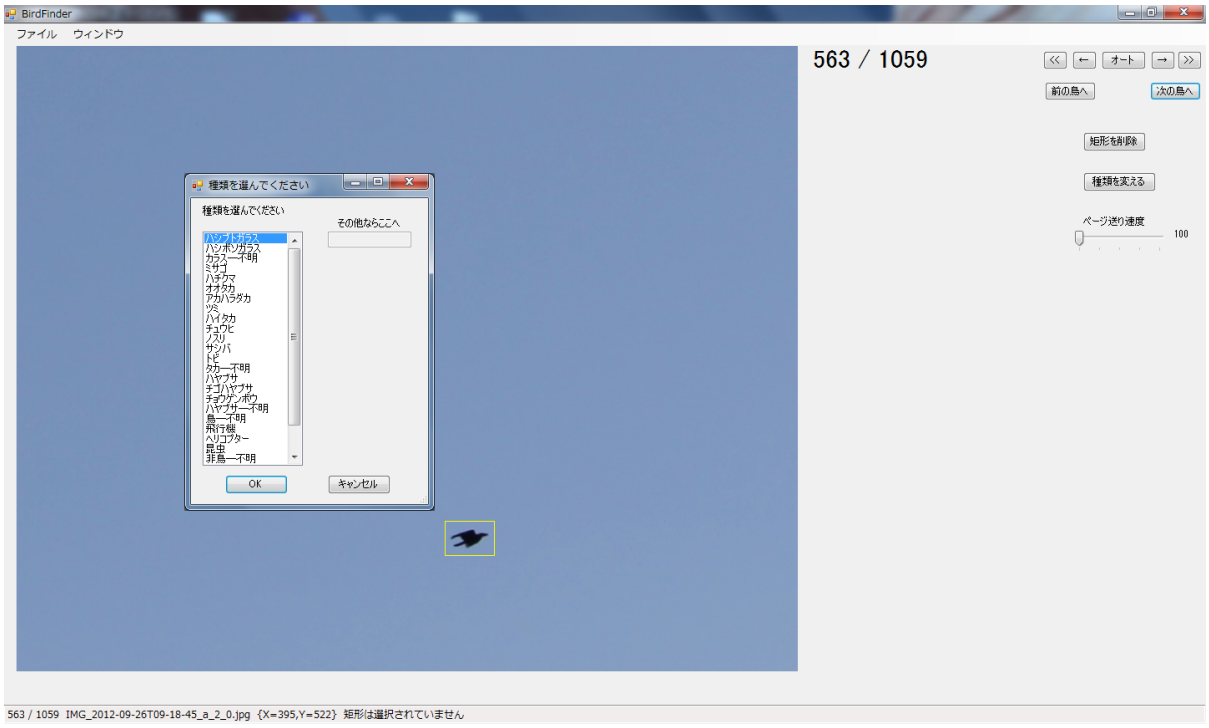


Figure3.9: Our user interface used in the annotation.

as a black kite, a kind of hawk, as a bird, or as an unclear flying object. These options become the nodes of the tree, and the depth of the tree corresponds to the level of detail. We made the list updatable, so that when an expert finds a bird that is not listed, he or she can add it to the list. Besides birds, other flying objects, such as airplanes, helicopters, insects, and falling leaves, are also recorded. By doing so, these objects can be distinguished from birds that might have been missed by the experts. Non- bird images can also be used as negative samples for machine learning. Objects that are too ambiguous for experts to distinguish are also recorded and labeled. Thus, the dataset contains 3 types of object: birds, non-birds, and unclear flying objects.

3.3.3 Manual labeling

Manually labeling sequential images of the dataset faces several issues. First, manual labeling is time consuming, because the images number as many as 32 442. To efficiently process the data, we developed a user interface that enables us to check images sequentially and label a bird with 2 actions, i.e., by making a bounding box by dragging a mouse and selecting a category from the list. The user's actual procedure is as follows: a user goes through a sequence frame by frame and checks if there is any flying object. When a flying object is found, he or she inputs the bounding box and selects the category from the given list, or else types in the category if it is not listed. The procedure is iterated until the end of the sequence. A screen shot of the interface is shown in Fig. 3.9.

Second, it is often difficult for non- experts to confirm an image to be of a bird, because of their small size in the images. We thus requested dozens of members from a wild bird society to inspect the images and input the data. Their efforts ensured that the labeling is precise and fine grained.

Third, due to the large size of the images (5616 by 3744 pixels), we cannot display their full size on an ordinary display. Therefore, we divided the original image into 30 (6 by 5) parts. One segment was assigned to a user,

who then went through a 1- day sequence of it (a total of 10 814 images). Birds that are on the boundaries of segments may be missed easily. To prevent this from happening, we asked the users to check the images twice, and we divided the images differently in each instance. In the first check, the images were divided into 30 (6 by 5) segments. In the second check, we shifted the dividing lines by half a segment. Fourth, manual checks may easily miss birds due to the large image size. To prevent this, the checks of each sequence were conducted by 2 different experts. We saved every bird that was annotated by at least 1 user. When the 2 experts labeled the same bird, and their overlap was larger than 25merge different birds into one, we did not find such cases because the birds in the images were sufficiently sparse and rarely overlapped.

3.4 Image recognition methods

Our algorithm is a combination of background subtraction [202] and object classification. Background subtraction is a method for extracting moving objects from fixed backgrounds and works well with our scenes that are mostly static. However, extracted regions still include some background objects, such as parts of the turbine, trees, or clouds; thus, we utilize machine-learning-based classifiers to filter birds from other objects. Specifically, we will compare two classifiers in the next section. The first is AdaBoost [70], a widely used learning algorithm in computer vision. This algorithm is often combined with image features such as Haar-like [40] or Histogram of Orientated Gradients (HOG) [41] for robustness. The performance of these methods is known to depend highly on both the type of target (faces, people, birds, etc.) and scene properties (indoor, street, wind farm, etc.). The second type is a CNN,21 the most successful deep network for object recognition to date. The strength of a CNN is that it learns features by itself; ie, it does not need manually designed image features that are not guaranteed to be optimal. Yet, it is important to determine whether CNNs outperform other methods in low-resolution detection and classification tasks. Because the CNN method has not been explored much, it is unclear what types of data and tasks it prefers. Below, we briefly explain the details of each method.

3.4.1 AdaBoost

AdaBoost [70] is a binary classifier based on feature selection and weighted majority voting. A strong classifier is made from a weighted sum of many weak classifiers, and the resulting classifier is shallow but robust. The classifier is expressed as

$$y = \sum_{i=1}^{N} \alpha_i h_i(\boldsymbol{x}). \tag{3.1}$$

Here, x denotes an input feature vector, and y is a class score. This formulation means N weak classifiers $h_i(x) \in 0, 1(i = 1, 2, ..., N)$ vote for the output with weights α_i. Training AdaBoost entails finding the set of weak classifiers and weights that minimize the classification error in the training samples. AdaBoost handles this optimization in a greedy manner, ie, through sequential selection of weak classifiers and weights. Given M training samples of input-output pairs and a set of weak classifier candidates H, the algorithm works as follows. First, it uniformly initializes the weights of the training samples by $D_j = 1/M(j = 1, 2, ..., M)$, which is later updated and used to compute the weak classifiers weights. Second, it selects one weak classifier with the lowest

error rate out of H by using the weighted training samples. The error rate of the i-th weak classifier is defined as

$$e_i = \frac{1}{M} \sum_{j=1, h_i(\boldsymbol{x}_j) \neq y_j}^{M} D_j. \tag{3.2}$$

Third, the weight of the selected weak classifier is set on the basis of the error it produces, as

$$\alpha_i = \frac{1}{2} \ln(\frac{1 - e_i}{e_i}). \tag{3.3}$$

Here, a larger weight is set for a smaller error rate, because weak classifiers with smaller error rates are more reliable. Fourth, it updates the weights of the training samples on the basis of the error rate of the reweighted classifier by

$$D_j \leftarrow \frac{D_j \exp(-\alpha_j y_j h_i(\boldsymbol{x}_j))}{Z}, \tag{3.4}$$

where Z is a normalization factor defined as the sum of the numerator values over $j = 1, ..., M$. This reweighting gives larger weights to the samples misclassified by and helps to select a complementary weak classifier to the current one in the next step. After that, the algorithm repeats the steps from 2 to 4 a fixed number of times. In practice, we need to select the number of weak classifiers N to be used and a set of weak classifier candidates H. H is given as elements of the features described below, and N is a tunable hyperparameter.

3.4.2 Haar-like

Haar-like image features [40] utilize contrasts in images. It extracts the light and shade of objects by using black-and-white patterns, as shown in Figure 3.10 (A). Feature extraction using these patterns can be performed by convolution. Let us denote an input image of size $W \times H$ as I, and a pattern of size $k \times k$ as w; then, the feature extraction by convolution can be written as

$$f(x, y) = \sum_{dx \in [0,k]} \sum_{dy \in [0,k]} I(x + dx, y + dy) w(dx, dy). \tag{3.5}$$

The pattern resembles a 2- dimensional Haar function whose value is one in white regions and minus one in black regions. Convolution of Haarlike patterns is equivalent to subtracting the sum of the pixel intensities in the black regions of the patterns from the sum in white regions, and hence, the features encode contrast in images. Specifically, we adopted all of the 14 patterns used in Viola and Jones [40] (see Figure 3.10 (A)). Haar-like features have been used for face detection and are considered fast and robust [40].

3.4.3 HOG

HOG [41] is a feature used for grasping the approximate shapes of objects. The method using this feature computes the spatial gradient of the image and makes a histogram of the quantized direction of the gradient in each local region, called a cell, in the image. Next, it concatenates the histograms of cells in the neighboring groups of cells, ie, blocks, and normalizes them by dividing by their Euclidean norms in each block. The pipeline of HOG feature extraction is shown in Figure 3.10 (B). HOG was first used for pedestrian detection [41], and it has since been applied to various tasks including generic object detection [272].

3.4.4 CNN

As described in Chapter 2, CNNs are the defacto standart methods in many visual recognition problems. Among the various CNN architectures, we examined 2 different ones, LeNet and ResNet. Our LeNet is based on a handwriting recognition method [81], in which the CNN appeared for the first time in the literature. LeNet is easier to train than modern deeper networks for middle-scale datasets such as MNIST [24] (10 classes, 60,000 training samples) because its number of parameters is smaller than deeper networks. Our LeNet was refined by utilizing 2 recent discoveries for improving performance: rectified linear units (ReLU) [105] and dropout [113].

The other CNN we used is a deep residual network (ResNet) [96]. ResNets are a special type of CNN that perform better than previous CNNs on generic image-recognition tasks [210]. The largest difference between ResNets and traditional CNNs is that ResNets have shortcut connections which jump over multiple convolutional layers. These shortcut connections can make the training of deeper networks easier because they prevent the gradients of the training error from vanishing or exploding by propagating the error directly to the lower layers. Our ResNet is a modified version of the one described in Takeki et al. [283, 284] and is more suitable to small-bird detection than the original one. The network configuration is shown in Figure 3.10 (D). The network has 32 layers and is deeper than LeNet.

3.5 Experiments

We conducted experiments involving three recognition tasks: bird detection, species classification, and species filtering using images taken at the wind farm. Here, we defined detection as a classification of objects into birds and non-birds, given the candidate regions suggested by the motion information. We defined classification as a classification between hawks and crows. These are the most frequent classes of birds in the area, and we had a sufficient amount of data for making an accurate evaluation of them. This 2-class classification is also practical because many endangered species are hawks. Finally, we defined filtering as a classification between hawks and all the other birds, including other species and unspecified birds, to extract target species from the whole dataset. Our dataset revealed that a large number of birds are not specified due to their small size in the image. While classification is an idealized setting within labeled data, through filtering, we can see whether the classifiers work as well in a practical situation as in an idealized one. Figure 3.11 show examples of the candidate regions in each experiment. In addition to these intra-dataset experiments, ie, training and testing on our dataset, we tested the trained detector on another wild bird dataset [214] to see inter-dataset generalizability, ie, how well detectors trained on our dataset generalize to another environment.

3.5.1 Experimental setup

Any machine learning method needs positive and negative samples for training. Both positive and negative samples were created by background subtraction40 from the images in our dataset. In the detection experiment, positive samples (birds) were regions labeled as birds in the dataset. The negative samples (non-birds) were background regions, or regions not labeled as birds in the dataset. Examples of birds and non-birds are shown in Figure 3.11. We used 6000 positive samples and 20,000 negative samples. We used five-fold cross-validation to conduct the experiment efficiently. In the classification experiment, hawks labeled in the dataset were positive samples, and crows were negative samples. Classification is a more difficult task than detection on this dataset;

thus, in order to analyze each methods behavior in detail, we investigated the effect of image resolution by dividing the positive and negative images into groups based on resolution. Specifically, the images of hawks and crows were divided into groups of 15 to 20, 21 to 30, and 31 to 50 pixels, as shown in Figure 6. On each group, we conducted holdout validation using 800 hawks and 150 crows for the training data and rest of each group for the test data. In the filtering experiment, we used the set of hawks, the same data as in classification, and the set of other birds including crows, other labeled birds, and unidentified birds. The set of others consisted of 15,000 samples, and we conducted 5- fold cross- validation again. We re-evaluated the three best methods in the classification experiment (hawk-vs-crow), namely RGB + AdaBoost, LeNet, and ResNet. We evaluated 2 CNNs, LeNet, and ResNet [96], as well as AdaBoost [70] combined with three types of features, Haar-like [40], HOG [41] features, and RGB (image pixel values without transformation). We quantified the detection and classification performance by using 2 measures, the true positive rate (TPR) and false positive rate (FPR). TPR is the number of true positives divided by the number of all positives in the test data. FPR is the number of false positives divided by the total number of negatives in the test data. Because there is a trade- off betweenTPR and FPR, the total performance of an algorithm is represented by the receiver operating characteristic curve (ROC), a curve of TPR versus FPR. A curve that goes near the upperleft-hand corner means better performance. Finally, we applied the bird detectors (LeNet and RGB-AdaBoost) trained only on our dataset to another bird dataset from a different location [214] to clarify the limitations of the trained classifiers and to see how generalizable they are to a new environment. The dataset used in Trinh et al. [214] was taken from 4K-resolution, 30-fps video of a different wind farm and has annotations of birds in a similar format to ours. The major differences between this dataset and ours are in the backgrounds, e.g., in the colors of the sky and clouds and in the shapes of the wind turbines.

3.5.2 Implementation details

Careful parameter setting of the used classifiers is necessary for a fair comparison of the methods. For AdaBoost, we set the number of weak classifiers used with Haar-like, HOG, and RGB features to 400. We found that more weak classifiers resulted in slightly better scores with all features. For example, AdaBoost with 800 weak classifiers performed around 0.2%-point better than with 400 weak classifiers in detection, at the cost of using more memory and training time. However, it did not change the order of the score of the methods. Cascading of weak classifiers is often used to speed up AdaBoost, but we did not use it because we wanted to avoid degradation in accuracy. Thus, all of the test images were classified with all weak classifiers. The features were extracted from images resized to 24 pixels square in all the experiments. After resizing, the dimensionality of the features was 5,567 in Haar-like, 1,296 in HOG, and 1,728 in RGB. CNNs have more parameters, ie, more layers and more and larger kernels in each layer, to specify the structure of networks. The parameters of LeNet were the same as in LeCun et al. [81], and those of ResNet were the same as in Takeki et al [284]. For executing the experiments, we used OpenCV2 [285] for the hand-crafted features and AdaBoost, and Caffe [17] for CNN and ResNet.

3.6 Results

The detection results are shown in Figure 3.12. In the figure, FPR means the rate of misrecognizing backgrounds as birds, and TPR means the rate of correctly recognizing birds. The best performance is achieved by ResNet and RGB. Even at the FPR of 0.05, ResNet detected over 0.98 of the birds. Figure 9A shows example images that were misrecognized as birds. They are moving backgrounds such as parts of the turbine, trees blown

by the wind, and flying objects such as airplanes and insects. Flying objects are more difficult negatives due to their visual similarity to birds. Note that the number of false detections depends on the number of negative samples in the data. More negative samples mean more false detections with the same FPR. Thus, the actual number of false detections may change depending on the test environment. The results of the classification (hawk vs crow) are shown in Figure 3.13. Here, FPR is the rate of misrecognizing crows as hawks, and TPR is the rate of correctly recognizing hawks. This curve shows the overall performance in the resolution groups. Because of visual similarity, the species classification is more difficult than the birds-versus-others classification; thus, its performance is lower. However, among the methods, the deep learning methods showed relatively promising results for classification. For example, at the FPR of 0.1, LeNet detected 0.83 of the hawks. By contrast, when we set the TPR at as high as 0.9, LeNet misclassified 0.4 of the crows as hawks. Figure 9B shows examples of correct and incorrect classifications with LeNet in each resolution group. Sometimes, visually similar images are correctly classified, sometimes not. The CNNs do not have explicit misclassification trends because of their black-box training process. The results of filtering (hawk vs other bird) are shown in Figure 3.14. In this case, ResNet slightly outperformed LeNet. For example, at the FPR of 0.1, ResNet detected 0.87 of the hawks. Similar to the results of hawk-vs-crow classification, the deep learning methods outperformed the methods based on hand-designed features. However, ResNet performed better than LeNet in filtering in contrast to classification. The results of detection in the dataset of Trinh et al44 are shown in Table 1. Here, we used the area under the curve (AUC), which shows the averageTPR over all FPRs, to see the overall performance. While RGB's performance greatly deteriorated from 0.992 to 0.511 because of the difference between the training and testing data, LeNet showed a smaller degradation (0.991 to 0.915). The results indicate better generalizability when using LeNet than when using RGB-based AdaBoost. Examples of detection in our dataset and in that of Trinh et al. [214] are shown in Figure 3.16. A bird was successfully detected in the dataset of Trinh et al44 despite the large visual differences between the scenes, but there were more misdetections around the turbine and the ground than in our dataset.

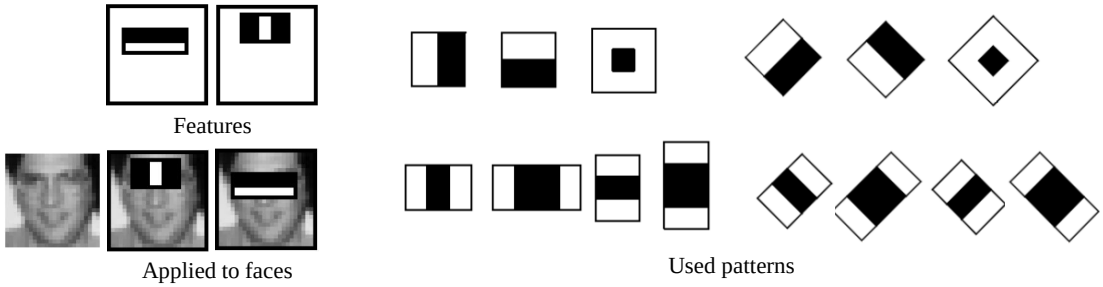
3.7 Discussion

In the detection experiment, RGB and ResNet performed the best among the methods, and the two methods performed almost equally. This may have been due to the low quality of the images. The existing features are designed for detecting objects such as faces and pedestrians, which are not often at low resolution. Thus, these features are not necessarily effective in our bird detection because of the limited object resolution. For example, HOG represents details of images by gradients and is preferred in tasks like pedestrian detection and generic object detection. However, it is less robust for low-resolution bird detection. We also note that the performance of CNNs depends on the parameters of the network and optimization. Although we used the parameters established in handwriting recognition [81], there may be better parameters for our images. A more extensive parameter search may improve the performance of both LeNet and ResNet. Our detection results are slightly different from those on a previous version of our dataset [286] because the dataset was updated. However, the qualitative results are consistent, that is, pleasant results with simpler methods (previously Haar-like features and currently RGB) and no significant advantage of deep learning. In the classification experiment, LeNet outperformed the other methods in all groups with different resolutions, and ResNet performed the second best. The hand-crafted features may be less effective in classification because of the subtle differences between the classes. Conversely, the learned features of the CNNs succeeded in adapting to the classification task through training. ResNet and LeNet changed

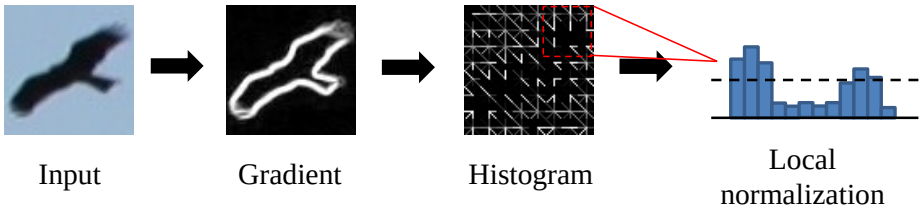
places in this classification, although ResNet performed the best in detection. The size of the training data may have been the reason. The classification experiments were conducted with less training data than in detection, and this put deeper networks, which are more difficult to train, at a disadvantage. The results of the filtering experiment suggest that classifiers work well even when unspecifiable birds exist in the environment. This means that our classifiers can extract a single species from all the data, and this is useful for investigation purposes. Each method performs 1020specifiable hawks, and this makes filtering easier than classification. Another interesting observation is that the deeper ResNet outperformed LeNet in filtering, as opposed to the results in classification. The major barrier to utilizing deeper networks is the difficulty in training them, which especially matters with smaller datasets. The results suggest that the smaller class in the dataset may become a bottleneck to exploiting deeper networks.

3.8 Conclusion

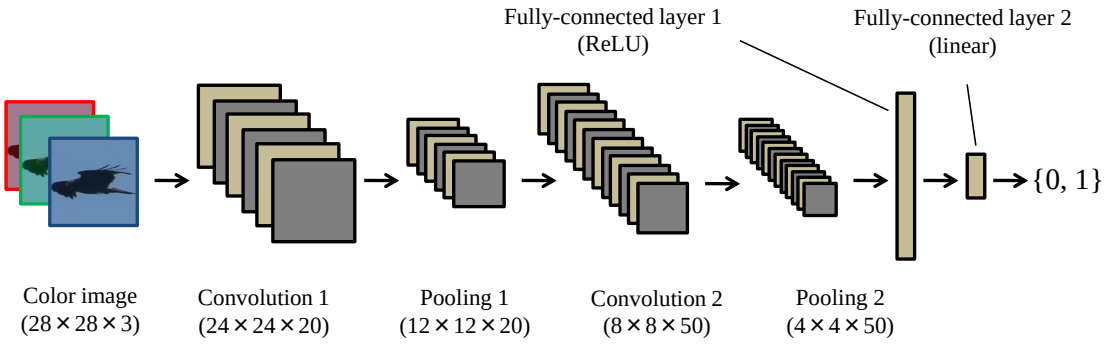
We constructed a bird-image dataset and evaluated typical image recognition methods for the purpose of developing an automatic bird detection and classification system for wind farms. By using our dataset from a realistic environment and representative methods in computer vision, we provided practical results and analyses of recognition performance. Interestingly, state-of-the-art deep learning did not outperform the simplest RGB features in bird detection, while deep learning was able to acquire generalizable features. The experimental results demonstrated the possibility of using image recognition for species classification. They also showed the effectiveness of using a state-of-the-art CNN for classification. However, there is still room for improvement in the species classification problem. Finally, we would like to emphasize that our computer-vision- based bird detection system is a potential solution to the problem of bird strikes and would thereby promote the social acceptance of wind energy.



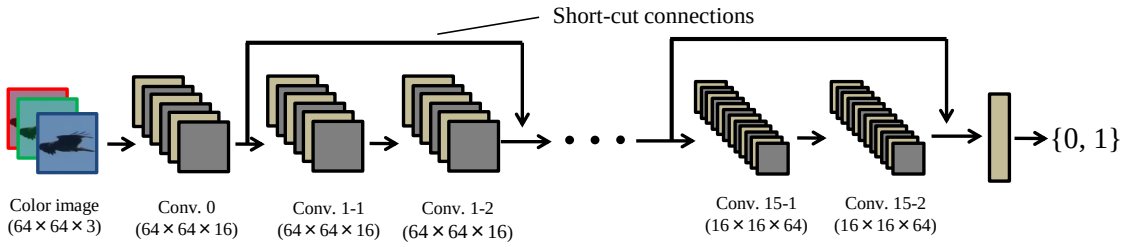
(a) Haar-like [22]



(b) HOG [23]



(c) LeNet [21]



(d) ResNet [38]

Figure3.10: Illustrations of image recognition methods we tested.

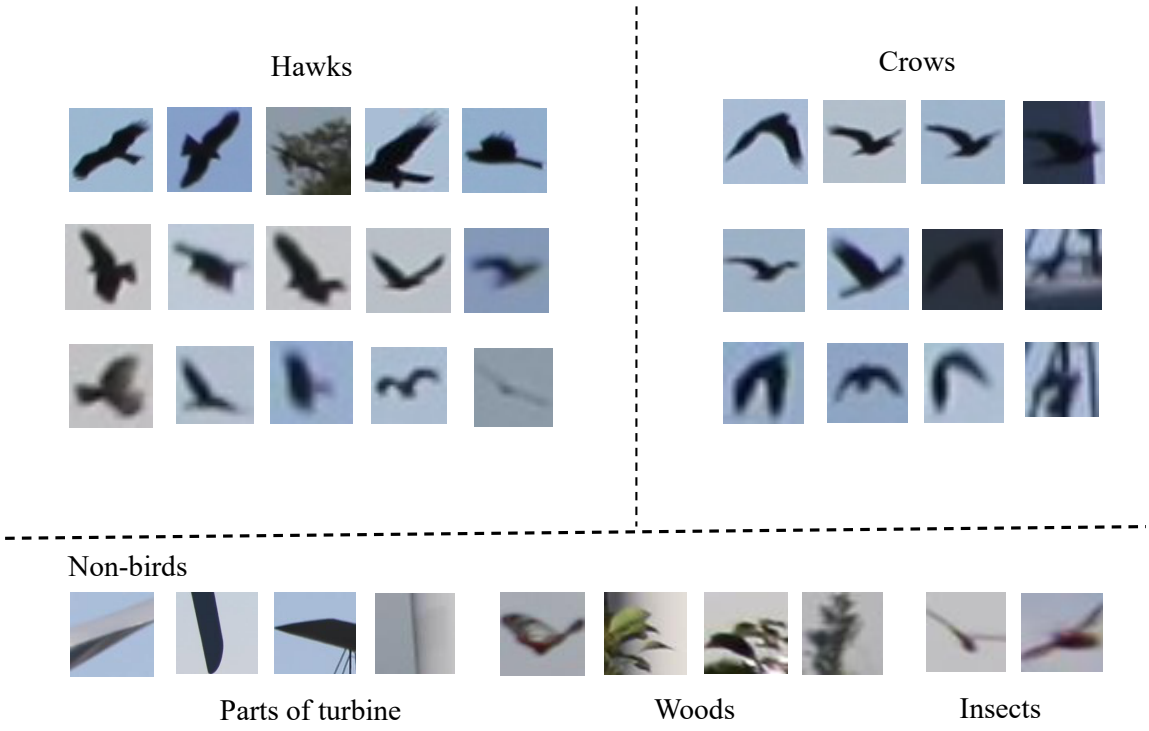


Figure3.11: Examples of bird and non-bird images used in the evaluation.

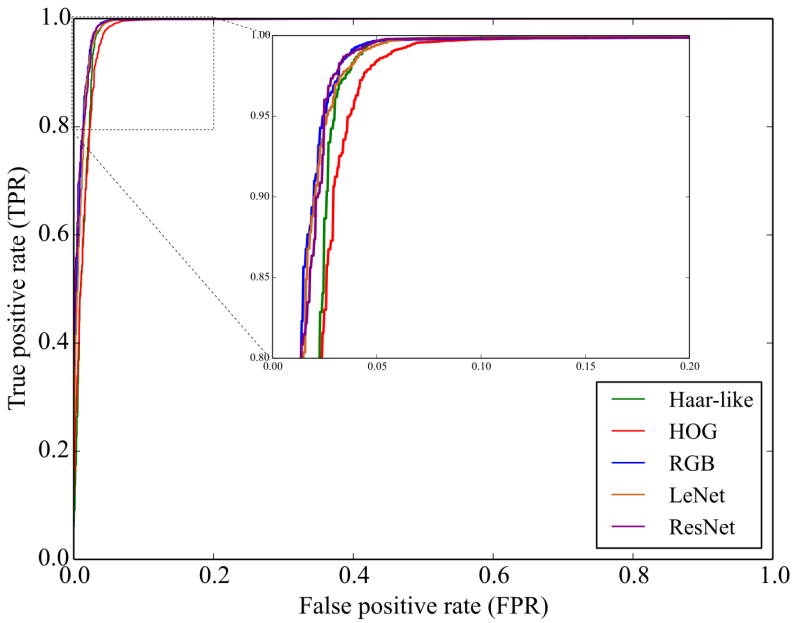


Figure3.12: Results of detection (birds versus others). Curves that go closer to the upper left-hand corner have better performance.

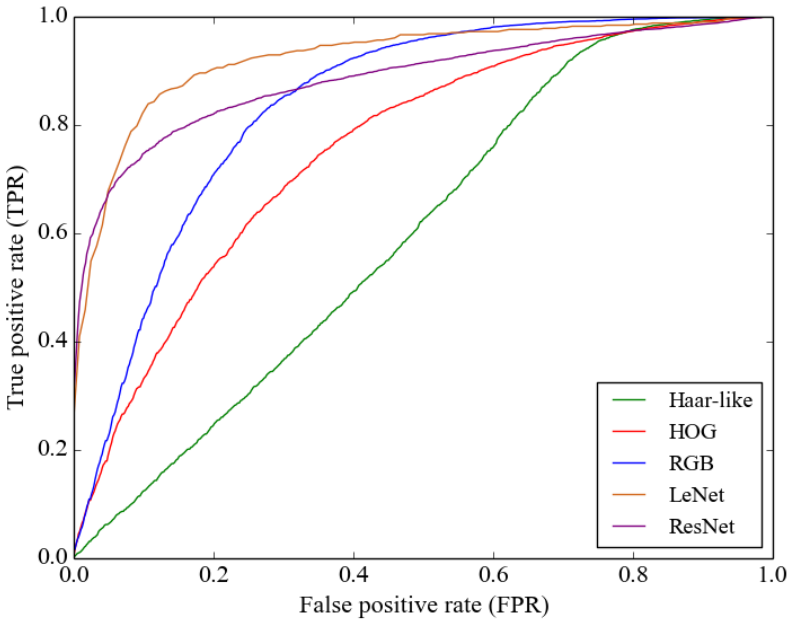


Figure3.13: Results of classification (hawks versus crows).

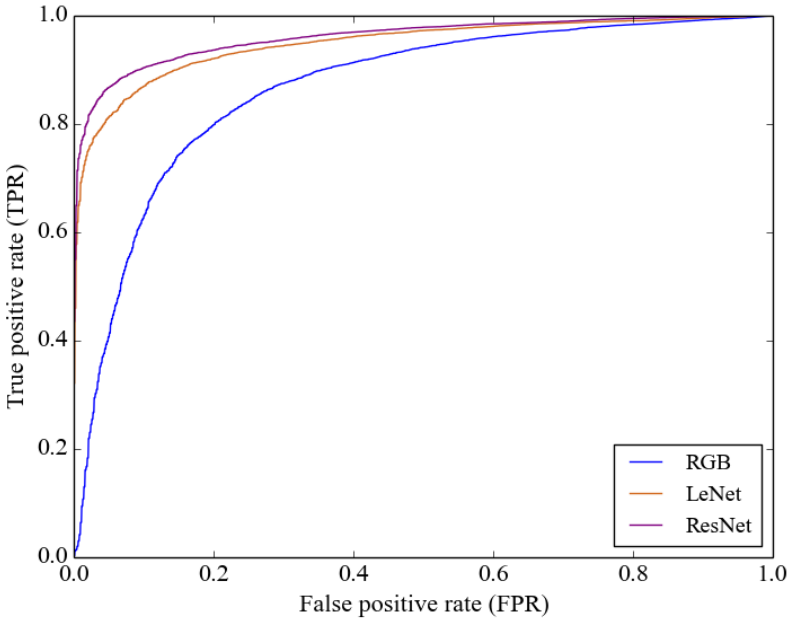
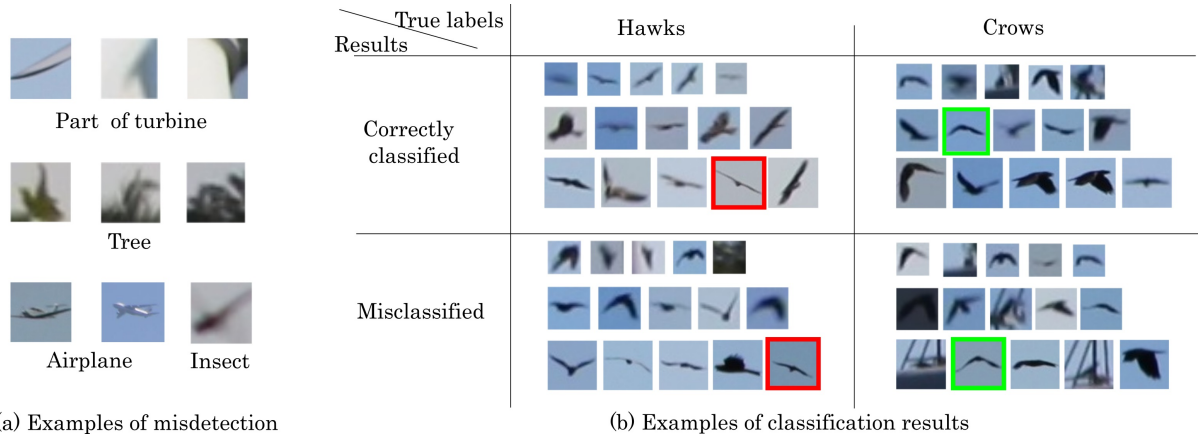


Figure3.14: Results of filtering (hawks versus other birds)



(a) Examples of misdetection

(b) Examples of classification results

Figure3.15: Examples of detection and classification. A) Images that were mis-detected as birds. B) Examples of correctly and incorrectly classified images.

Figure3.16: Examples of detection using images from our dataset (upper) and a dataset gathered at a different location [214] (lower) by LeNet trained on our constructed dataset. Green boxes show correct detections of birds; red ones show misdetections.

4

Detection by motion-feature learning

4.1 Introduction

A large part of today's object detectors are fully based on appearace information from still images. Howver, such detectors, even state-of-the-art ones, still miss many object under hard situations. For example, in pedestrian detection from an on-board camera, it is still important to detect pedestrians far from the vehicle for anticipating potential accidents [287]. Such pedestrians who appear small, vague or non-salient are diffcult to detec, because appearance features cannot be helpful to resolve such hard instances. Contrarily, one of the important factors in this detection is to exploit motion information. Human brains process motion in the early stage in the pipeline of the visual cortices to enable humans to notice and react to moving objects quickly [288]. Ambiguous objects in still images that are not recognizable to humans can be sometimes easily distinguished when motion is available. Following this idea, several studies have shown that detection performance can be improved by combining motion features with appearance features [2, 43, 166, 198, 200, 201]. However, performance gain by motion features has been slight, and how to incorporate motion clues into detection to achieve the best performance is still under discussion.

Despite the importance of motion information, they are less exploited with current state-of-the-art pedestrian detection methods. According to the benchmark Caltech Pedestrian [75], incorporated motion features still rely only on hand-crafted features such as histograms of oriented gradients of optical Flow (HOF [43, 198]), or Temporal Differences of weakly Stabilized frames (SDt [2]), which is relatively more popular as it ignores non-informative motions. The key insights of these hand-crafted features are that (1) contours of moving objects in fine scale is important for detection, and (2) informative motions have to be extracted by factoring out unnecessary motions (such as camera-centric motions).

Deep motion features, on the other hand, have been actively explored in activity recognition [205,206,289–292]. Though deep convolutional neural networks (convnets) over spatio-temporal space had not been competitive against hand-crafted features with sophisticated encodings and classifiers, recently proposed two-stream convnets [207] has achieved remarkable progress. Two-stream convnets are inspired by two (ventral and dorsal) streams in the human brain, and capture spatial and optical-flow features in separate convnets. Spatial features from convnets have been extensively used in pedestrian detection, and deep methods [77,293,294] have produced state-of-the-art results. Thus, a two-stream framework to combine spatial and temporal features should enable significant and natural improvements over these methods.

We present a deep learning method for detection that can exploit both spatial and temporal information with two-stream convnets. Based on the findings in hand-crafted motion features, we demonstrate that deep learning over SDt [2] efficiently models the contours of moving objects in fine scale without unwanted motion edges. Our deep motion features are more discriminative than hand-crafted or raw features, as it has been the case for still-image features. The presented convnets are novel in the following two aspects. First, SDt, an effective motion feature for pedestrian detection, is used as inputs for temporal convnets instead of raw optical flow, as it factors out camera- and object-centric motions that are prominent in videos from car-mounted cameras. Second, we adopt concatenation of mid-level feature maps instead of late fusion of class scores [207], which is similarly done in channel-feature-based methods [44, 50, 77]; thus, our networks can be applied to arbitrary input image sizes, and enable sliding-window detection rather than video-level classification. Our detector is implemented on the basis of the convolutional channel feature detector [77], which provides a simple yet powerful baseline for deep pedestrian detection methods. The resulting architecture can effectively learn features from multiple datasets. Furthermore, we also implement a bird detector based on the same idea and show that it outperform singe-frame

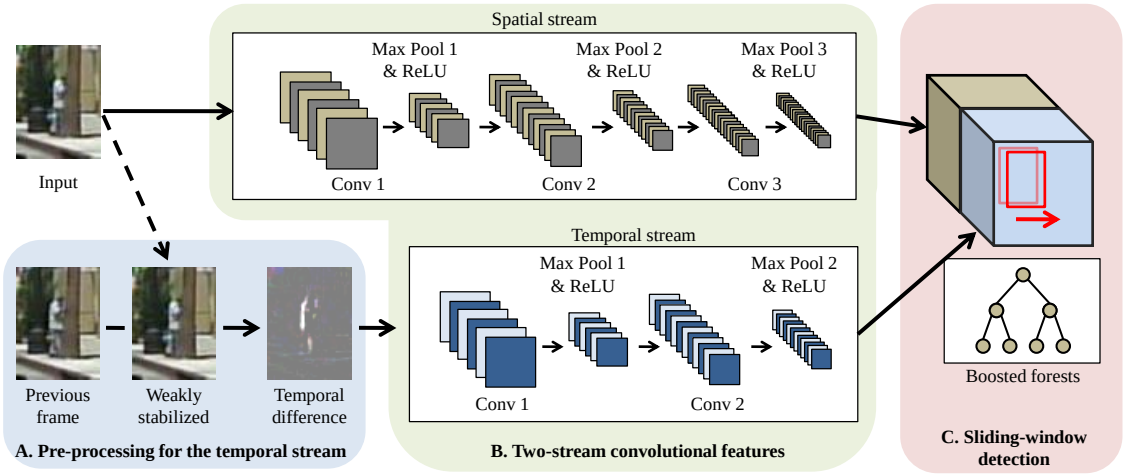


Figure4.1: Overview of our two-stream detecotr. First we pre-process video frames by coarse alignment and temporal differencing for effective motion description (a). Next, we input raw current frames and temporal differences into convolutional layers to extract spatio-temporal features (b). Finally boosted forests perform sliding-window detection over the convolutional feature map (c).

based detectors.

Our contributions in this chapter are four-fold. First, We propose the first method that utilizes deep motion feature for pedestrian detection. Second, we design a two-stream network structure that enables sliding-window detection. Third, we show that an activity recognition dataset [3] can improve pedestrian detection through transfer learning. Finally, our experimental results reveal how deep motion features can be effective by benchmarking in open datasets and via intensive analysis. The proposed method with deep motion features achieved 10% reduction of miss rate from a deep detector only with still-image feature [77] in popular *Caltech Pedestrian* and *DaimlerMono*, and this is a larger improvement than directly incorporating SDt [2].

4.2 Motivation from biology

Our motivation of exploiting motion features in pedestrian detection and designing two-stream network architecture is from insights from neuroscience and visual psychology on how human vision processes motion information. From the structural viewpoint, visual cortices have two main streams for recognition and motion, namely ventral stream and dorsal stream [295]. This inspired two-stream structure in artificial neural networks for video tasks [207, 296], including ours. While the processing pipeline of motion is hierarchical, it is worth noting emphasis of temporal changes are performed at early stages by cells sensitive to temporal changes [288].

From the functional viewpoint, perception of motion is related to recognition of objects even when appearance features of objects are unavailable. An example is biological motion [297], which is a phenomenon that motion patterns showed via individually meaningless signals (i.e., point light) can be perceived as human locomotion or gestures. Based on these insights, we design two-stream pedestrian detection method aided by motion information.

4.3 Two-stream convnets for pedestrian detection

The overview of our method is shown in Figure 4.1. It consists of a two-stream convnet that takes the current frame for spatial convnets, and the temporal difference of weakly stabilized frames via coarse optical flow for temporal convnets. The outputs from the two streams are concatenated and compose feature maps, and the boosted forests perform sliding-window detection over them.

4.3.1 Two-stream convolutional features

Our pipeline incorporates motion information via two-stream convolutional features (Figure 4.1 B), which is a two-stream network architecture [207] adopted in sliding-window detection. The first stream for spatial features follows the framework of Convolutional Channel Features (CCF) [77] as a baseline. It consists of the lower part of VGGNet-16 [109], which includes three convolutional layers and three max-pooling layers, and is pre-trained on the ILSVRC2012 dataset [210]. According to Yang et al. [77], VGGNet-16 produces better features for pedestrian detection than AlexNet [12] or GoogLeNet [110], supposedly because deeper networks are difficult to transfer to other tasks.

For the second stream for temporal features, we build convolutional features specialized in motion feature extraction. While ImageNet-pretrained features generalized well to detection tasks [77, 123], it does not work for motion handling because they are derived from the still-image classification task. Therefore, we train the convnets over difference images collected from video datasets. The training is necessary, if we want to work on temporal differences, because statistical distribution of difference images has different nature from that of still images. We collect difference images of UCF-101 [3], a large activity recognition dataset with 13,320 video clips categorized into 101 classes, and we train AlexNet on them. AlexNet is chosen because of its trainability. It has less layers than VGGNet-16 or GoogleNet, and therefore is easy to train on datasets smaller than ImageNet such as UCF-101. For feature extraction, we used the second convolutional layer in AlexNet, which makes 256-dimensional feature maps, the same dimension as that of the spatial stream. We refer to this trained AlexNet as UCFAlex.

There are several options for the second covnets. For instance, one may think that we can reuse VGGNet-16 of the first stream, if we want to input the previous frame or weakly aligned previous frame to the temporal stream. However, such an approach can be weak, because statistical distribution of difference images has different nature from that of still images. We see the performance gain by our motion convnet and covnets in the Experiments section.

For training of the temporal stream, we use split01 of UCF-101. Specifically, we use the first group (divided according to the source videos of each clip) for validation and the remaining 24 groups for training. We choose stochastic gradient descent [298], the most common convnet solver for training in the same setting as those by Simonyan and Zisserman [207], except that we initialize network parameters with those of AlexNet trained on ILSVRC2012 for smooth training. The accuracy on the validation set is 56.5% after training, which outperforms AlexNet on the raw RGB values of each frame (43.3%) and underperforms the temporal stream on the optical flow reported by Simonyan and Zisserman [207]. However, this is not a problem because our purpose was not to achieve the best performance on the activity dataset but to acquire effective features for temporal-difference images. In training, we used difference images of four-frame intervals. We do not apply weak stabilization to UCF-101 videos because camera-centric motion is much less in this dataset.

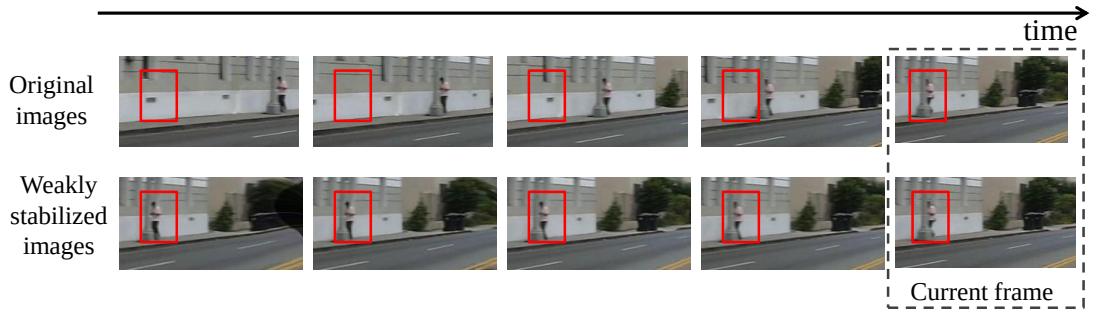


Figure4.2: Effects of weak motion stabilization [2]. Upper images are original input images and lower images are stabilized ones. Background motion is stabilized while deformation of the person remains.

4.3.2 Sliding-window detection

The final output of our network is determined using boosted forests. Our detector is a sliding window detector that classifies densely sampled windows over the feature maps at test time (Figure 4.1 C).

A main challenge in training the two-stream detector is the larger dimensionality of the feature maps. The output from the two streams has 512 channels after concatenation and this becomes twice as large as the single spatial stream which makes forests likely to cause overfitting. Thus, we apply data augmentation in the training of the forests. We include both images with and without motion features in the training data. Let us denote the pair of the inputs to our two-stream convnets as $(I, D(I))$, where I is the input image and $D(I)$ is the temporal difference of I. We prepare both $(I, D(I))$ and $(I, 0)$ as training data for each labeled bounding box. The term $D(I) = 0$ means that the object in I is stationary. This data augmentation mitigates overfitting and prevents detectors from missing stationary people in the scenes.

We set the hyper parameters of boosted forests to the same as those discussed by Yang et al. [77]. Namely, the number of trees was 4096 and maximum depth of each tree was 5. The ensemble was constructed with the Real AdaBoost algorithm and the trees were trained with brute-force-like search and pruning [74]. Convnet features are known to be powerful enough even without fine-tuning on the target tasks [299]. Thus, we did not fine-tune either of the two streams and only the forests were trained on the Caltech Pedestrian Detection Benchmark.

4.3.3 Pre-processing for the temporal stream

We use temporal differences of weakly stabilized frames [2] as motion descriptors to input to the temporal stream (Figure 4.1 A). We describe it here for completeness and convenience in notation. First we calculate the optical flow [300] between the current frame that contains the target and its previous frames. Next, we warp the previous frames using the optical flow. We denote these frames as $I_p(t_0, t_{prev})$ where t_0 is the current time, t_{prev} is the previous time, and p is the smallest size of the patches used in calculating the optical flow. The optical flow is calculated in a coarse-to-fine manner, and p controls the fineness of the flow. We can use several previous frames and obtain longer motion features as follows. Weakly stabilized frames are

$$S(t_0) = \begin{pmatrix} I_p(t_0, t_1) \\ I_p(t_0, t_2) \\ \cdots \\ I_p(t_0, t_n) \end{pmatrix} \tag{4.1}$$

and temporal differences of these frames are

$$\begin{aligned} D(t_0) &= \begin{pmatrix} I(t_0) \\ I(t_0) \\ \cdots \\ I(t_0) \end{pmatrix} - S(t_0) \\ &= \begin{pmatrix} I(t_0) - I_p(t_0, t_1) \\ I(t_0) - I_p(t_0, t_2) \\ \cdots \\ I(t_0) - I_p(t_0, t_n) \end{pmatrix}, \end{aligned} \tag{4.2}$$

where $I(t_0)$ is the current frame and n is the number of frames, which gives the time range of motion features.

The choice of p is important for removing relatively uninformative camera-centric or object-centric motions while preserving object deformation and pose change. The coarse flow is useful for this purpose, as previously discussed [2]. We set p to 32 pixels square. For calculating motion, we used the second and fourth previous frames, following [2] for Caltech Pedestrian. There are a variety of optical flow methods for stabilization. Our choice is Lucas-Kanade flow [300] because it is simple and robust enough to work on Caltech Pedestrian images (30 fps, 640×480 pixels). Other options [301, 302] are also possible and may perform better on more complex scenes or low frame-rate videos.

4.3.4 Implementation details

Feature map generation We use convolutional layers as feature transformation per frame to enable sliding-window detection over the feature maps, differently from per-window classification approaches [123, 294]. In classical neural networks, the input size (of the image) has to be fixed; however, convnets have been recently used as flexible filters, which are applicable to images of arbitrary size (over the size of the convolutional kernels) and fixed channel numbers [77, 130, 303]. This becomes possible when convnets are without fully-connected layers. We use convnets in this manner; namely, they are applied to clipped windows in training but to full-size input images during the test. During the test, the forest classifier looks up pixels in the feature maps, the pre-computed convolutional features of the whole image, to execute sliding-window detection. We also adopt pyramid patchworking [304] during the test. That is, we stitch the spatial pyramid of an input image into a larger single image, which shortens test runtime. This makes the input image size 932×932 pixels while the original size of frames is 640×480 pixels.

Computation Our implementation is based on the CCF [77] implementation, which is based on Piotr's MATLAB Toolbox including the aggregate channel feature (ACF) [44] detector. It also uses Caffe [17] for convolu-

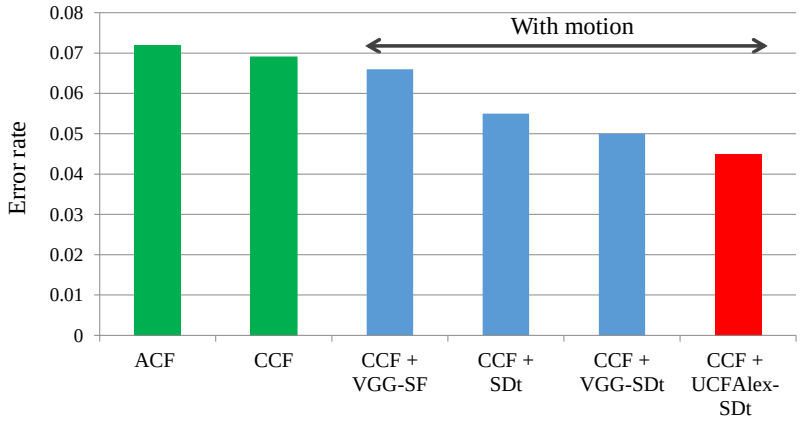


Figure4.3: Error rates of the methods in Table 4.1 in patch-based validation.

tional feature extraction on GPUs. Weak motion stabilization is also included in Piotr's Toolbox. Our test and training environment was a workstation with two Intel Core i7-5930K CPUs (3.50 GHz, 12 cores), 512 GB memory, and two NVIDIA GeForce GTX TITAN X video cards. Note that 512 GB memory is necessary for training our boosted forests on memory with data augmentation.

4.4 Experiments

To provide comprehensive understanding of the deep motion feature of our proposed mothod, we present experiments and analyses consisting of the following three parts: validation, evaluation, and visualization. First, we validated our configuration of networks by comparing it with baselines such as hand-crafted motion features and ImageNet-pretrained convnets used as the temporal stream (*validation*). Next, we evaluated our detector with the motion features in the test sets of Caltech Pedestrian Detection Benchmark [305] and Daimler Mono Pedestrian Detection Benchmark Dataset [306], and compared our detector with the state-of-the-art methods from the leader boards (*evaluation*). Finally, we visualize the detection results and learned models to understand how and when the motion features improve pedestrian detection (*visualization*).

4.4.1 Network selection and validation

First, we compared three candidate combinations of the network architecture, pre-training data and inputs for the temporal stream of our network: (1) VGGNet-16 pre-trained on ILSVRC2012 by inputting previous weakly stabilized frames (**VGG-SF**), (2) VGGNet-16 pre-trained on ILSVRC2012 by inputting the temporal differences of stabilized frames (**VGG-SDt**), and (3) AlexNet fully trained on UCF-101 by inputting the temporal differences of stabilized frames (**UCFAlex-SDt**). We also validated **ACF**, **CCF**, and our new implementation of **CCF+SDt** for comparison. We used previous second and fourth frames in CCF+SDt. Only the previous fourth frames is used in temporal convnets for simpler implementation. The compared combinations are listed in Table 4.1.

The performances of the candidates were evaluated with a bounding-box-level classification task using the training sets. First, we collected clipped bounding boxes from the training sets of the benchmark (set00–set05). We used every fourth frame from the training set to acquire more training boxes. This setting is often referred to as *Caltech10x*. We used the ground truth annotations. For negative windows, we used those generated in the

Table4.1: Methods we validated in the splits of training data in Caltech Pedestrian Detection Benchmark [75], sorted by validation error rates in descending order (corresponding to Figure 4.3.

Method	Spatial feature	Temporal feature	Temporal pre-training	Temporal input
ACF [44]	Hand-crafted	-	-	-
CCF [77]	VGGNet-16	-	-	-
CCF + VGG-SF	VGGNet -16	VGGNet-16	ILSVRC2012	SF
CCF + SDt	VGGNet-16	SDt itself	-	SDt
CCF + VGG-SDt	VGGNet-16	VGGNet-16	ILSVRC2012	SDt
CCF + UCFAlex-SDt	VGGNet-16	UCFAlex	UCF-101	SDt

hard-negative mining process for training of a baseline algorithm, i.e., ACF detector [44]. We conducted 3-stage hard-negative mining with ACF and refer to those collected negatives as *NegStage1–3*. Next, we split each group of bounding boxes into two parts, the first half for training and the latter half for validation. We did not apply data augmentation in this validation for comparison in an equal amount of training data. For hyperparameters of boosted forests, the maximum number of weak classifiers was fixed to 4096, and the maximum depth of trees to 5. Note that only the forest classifier was trained on Caltech Pedestrian and the networks were not fine-tuned after pre-training separately on Caltech Pedestrian or other datasets.

Figure 4.3 shows the validation results. All methods with multi-frame motion information improved in accuracy compared to single CCF. Motion information from stabilized differences had more positive effects on classification than that from stabilized frames without time differencing, even VGGNet-16 trained on still images was used (CCF+VGG-SF vs. CCF+VGG-SDt). This should be because of the correlation between frames, which is known to be harmful for decision trees [307]. Time differencing can mitigate this correlation and improve training of decision trees. Convnets over SDt worked better than simple SDt (CCF+SDt vs. CCF+VGG-SDt), and our motion-specialized AlexNet was more suited to process SDt (CCF+VGG-SDt vs. CCF+UCFAlex-SDt). It would be surprising that transferred features from the far task of activity recognition work in pedestrian detection. This suggests that the distance of input input domain, i.e., still-images vs. difference images mattered more than the distance of the tasks in reusability of lower-level convolutional features. Several examples for difference images for pre-training are shown in Figure 4.4. Overall, the combination of AlexNet re-trained on activity recognition and stabilized differences (CCF+UCFAlex-SDt) performed the best among the candidates. Below, we refer to CCF+UCFAlex-SDt as *TwoStream*.

4.4.2 Detection evaluation

Next, we conducted detection experiments on the test sets of the Caltech Pedestrian and DaimlerMono to evaluate the performance of TwoStream. We also compared the results of TwoStream to current pedestrian detection methods including ones using only single frames and using motion information. We also evaluated our implementation of CCF+SDt to compare to our temporal convnets.

In the evaluation, we strictly followed the evaluation protocol of the benchmark. For the training/test split, we used the suggested split on the benchmark, namely, set00–05 for training and set06–10 for test. Detection was run on every thirtieth frame of each test video. The previous frames of the test frames were also used in the test because TwoStream requires them. We used second and fourth previous frames in CCF+SDt and fourth frames in TwoStream. We conducted all the evaluation process using official APIs for the dataset. We also applied our

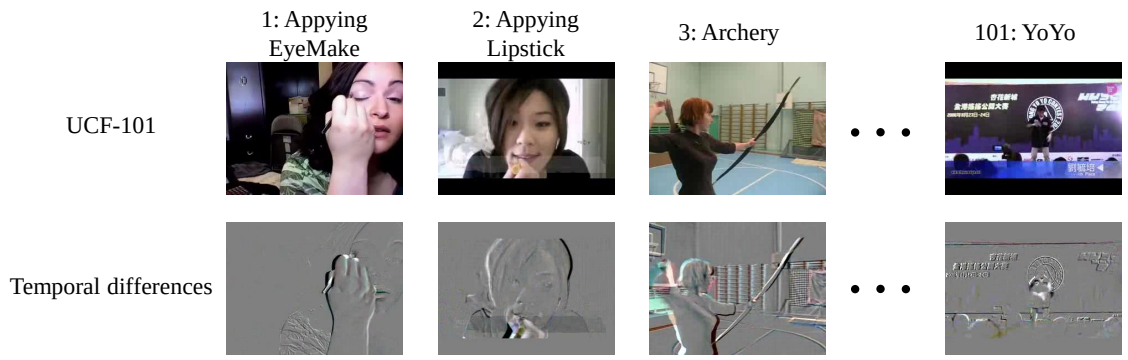


Figure4.4: Temporal differences in UCF-101 [3], activity recognition dataset we used to train temporal stream.

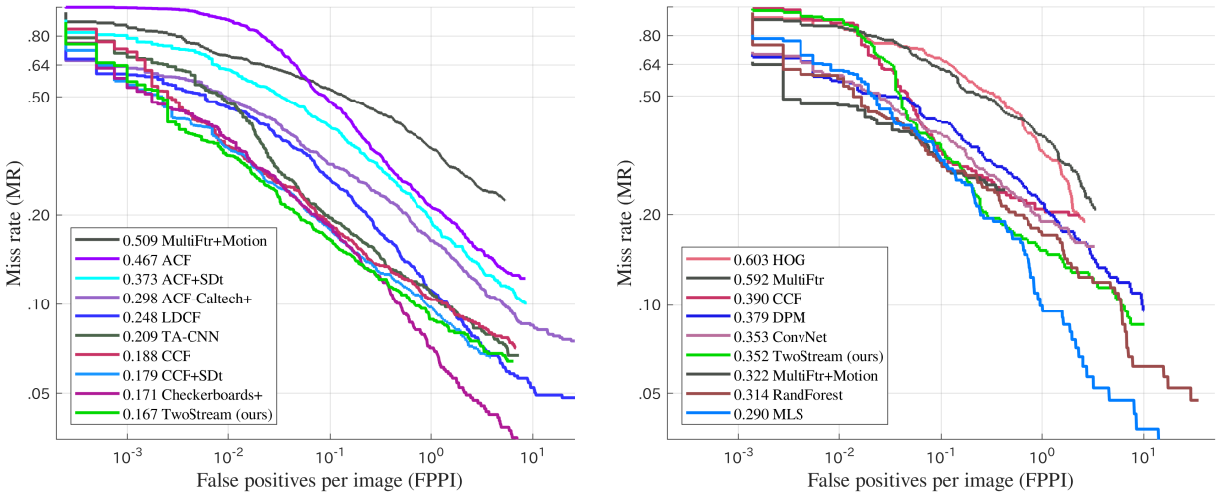
detector to the test set of DaimlerMono in the same manner to see whether our detector generalizes to another dataset. Because DamilerMono only provide gray-scale frames, we convert them to RGB by simply copying the channels. As we aim to evaluate transferability of our Caltech-trained detector, note that the training set of DaimlerMono is not used.

To train our detector for evaluation, we sampled bounding boxes from every fourth frames from the training sets (set00–05), using the baseline detector, i.e., ACF [44]. For better performance, we used flipped windows of positive samples in addition to the original training data. We also applied data augmentation to the positives and NegStage1 and 3, but not to NegStage2 due to memory reasons. The numbers of positive and negative training windows were 50,000 and 100,000, respectively, after data augmentation.

The results were evaluated using receiver operation characteristic (ROC) curves, which plot detection miss rates to the numbers of false positives per images. Figure 4.5 (a) shows the ROC curves of the detection results in Caltech Pedestrian. It also shows published results in the evaluation paper and scoreboard of the benchmark for comparison. We included the results of boosted-forest-based methods, namely, ACF and ACF-caltech+ [44], LDCF [307], Checkerboards+ [50], TA-CNN [293], CCF [77], MultiFtr+Motion [43], ACF+SDt [2], and our baseline implementation for a deep method with motion (CCF+SDt). The proposed method (TwoStream, log-average miss rate: 0.167) performed 10% better than motion-free baseline (CCF, 0.188), and this is a larger improvement than by simply adding SDt to CCF. TwoStream performed better than all the hand-crafted-feature-based detectors with or without motion and two deep detectors CCF and TA-CNN.

Figure 4.6 shows several examples of detection in Caltech Pedestrian. TwoStream detected more pedestrians than single-frame-based CCF due to temporal information, as shown in Figure 4.6 (a). Low-resolution pedestrians were particularly difficult even for humans to detect in single frames; however, TwoStream succeeded in detecting them from motion. In contrast, Figure 4.6 (b) shows suppressed false detections by TwoStream, which were misdetected by only single-frame information. This shows motion is also useful to reject non-pedestrian region more easily than only by appearance.

Figure 4.5 (b) shows the ROC curves of the detection results in DaimlerMono. In this transfer setting from Caltech to DamilerMono, the proposed TwoStream performed better than the baseline CCF by 3.8 percentage points, or 10.7 percent in relative. This means TwoStream is transferable to another dataset from Caltech Pedestrian. TwoStream also outperformed existing *ConvNet* [308], while the hand-crafted-feature-based methods (MultiFtr+Motion [43], RandForest [309], and MLS [310]), which are considered to be more robust in gray-scale



(a) Results in Caltech *Reasonable*

(b) Results in DaimlerMono

Figure4.5: Detection result curves. Lower miss rates show better detection performance. We also show log-average miss rates of the methods next to their labels.

images, marked better scores. We also note that these three detectors are trained in INRIA dataset, that seems to be closer to DamilerMono, while we used Caltech Pedestrian.

4.4.3 Analyses and visualization

How did detection confidences change? To understand how the motion stream helps recognition, we present examples of patches differently scored with TwoStream and single-frame baseline CCF in Figure 4.7. We also show each patch's temporal differences with weak stabilization, which were the input for the motion stream in TwoStream. Figure 4.7 (A) shows negative samples scored highly with CCF but were mitigated with TwoStream. Non-pedestrian objects such as poles (a and b) or parts of vehicles (c and d) also scored highly. They have vertical edges and may be misclassified as pedestrians only by appearance, but they can be correctly rejected by motion information because they are rigid and their temporal differences are negligible after stabilization. However, non-pedestrian samples may produce large temporal differences by fast motion information such as crossing vehicles (e). In this case, the motion stream CNN seems to discriminate difference patterns and correctly ignore non-pedestrian motions. (d) is a hard negative of a silhouette of a person in a traffic sign, that was successfully rejected by TwoStream. Figure 4.7 (B) includes pedestrians that scored higher with TwoStream. TwoStream scored highly for pedestrians with typical and salient motion patterns (g, h, and j) by large margins, which supports that it utilizes pedestrian motions for recognition. In addition, blurred or non-salient pedestrians (i, k, and l) tended to be scored lower on the basis of only appearance features, but many parts of such pedestrians were detected thanks to motion feature information. There were several contrasted cases, i.e., non-pedestrians scored higher and pedestrians scored lower with TwoStream, as shown in Figures 4.7 (C) and (D), respectively. Negative patches with complex shapes such as trees or parts of buildings were sometimes scored the same or higher with TwoStream (m and n), and a vehicle's motion that was not correctly stabilized (o) also caused misdetection. Pedestrians were scored lower when they were static (p), occluded by moving vehicles (r), or made rare motions such as getting into cars

Table4.2: Reduction of MR by combining our deep motion feature with various appearance features. Lower is better.

Base Method	ACF-ours	LDCF-ours	CCF	Two-stage CNN
Appearance feature only	36.7	33.4	18.8	13.6
With our deep motion feature	**31.4**	**30.6**	**16.4**	**12.4**

(q), whose motion patterns the detector may fail to learn from the training set.

Visualization We visualized trained forests on different features, as shown in Figure 4.8. The figure shows frequency distributions of where in feature maps was seen by decision nodes in the forests. Comparing ACF and CCF, a clearer pedestrian shape on the deep feature maps via CCF was observed, while ACF used backgrounds as well. Comparing the temporal features of CCF+SDt and TwoStream, SDt grasps pedestrian contours while TwoStream used more motion patterns around legs and heads.

Runtime The runtime of TwoStream on 640×480 videos were 5.0 second per frame on average. The runtime of CCF was 3.7 seconds per frame in [77] and 2.9 second per frame in our environment. Both were implemented on MATLAB and Caffe on GPUs. The overheads with our temporal stream and coarse optical flow were less dominant than other factors including video decoding and communication with GPUs in our implementation.

Tree depth Tree depth is an important parameter to fully exploit strong features in boosted-forest-based detectors. We further investigated the impact of tree depth in the forest in Figure 4.9. We found that the miss rate did not largely differs with depths of around 6. It consistently outperformed the baseline CCF with depth of 5 to 8 and thus it is robust against setting of this hyperparameter. Here we put best-performing CCF with tree depth of 5 as the baseline. Nevertheless, overly shallow (less than 4) or deep (over 9) trees degraded detection performance.

Failure cases Failure cases where TwoStream did not improve detection compared to the original CCF are shown in Figure 4.10. A major cause of failures is inaccurate optical flow around occlusion (the top sample in Figure 4.10) or image boundaries (the lower samples), which fails to remove background motions. However, we also noted that TwoStream improved the total detection accuracy despite such optical flow errors, thanks to complementary usage of appearance and motion features.

4.4.4 Combination with other methods

To show the effectiveness of our deep motion feature, we combined them with other types of methods and evaluated relative performances. First, we adopted popular hand-crafted channel-features, ACF [44] and LDCF [307], in addition to CCF. In implementation, we replaced CCF in TwoStream by the other appearance features without modification. Training details are the same as those for our implementation of CCF. The results are shown in Table 4.2. Our deep motion features decreased MR by 5.3% with ACF, and by 2.8% with LDCF. The results confirmed that our deep motion features consistently improve detection performance of various appearance-based features.

Second, we examined the combination with second-stage CNN (Two-stage CNN), that gives two-stage pipelines similar to state-of-the-art systems [294, 311, 312]. In this setting, we rescore the detected regions by CCF or TwoStream by a vanilla VGGNet-16 [109] fine-tuned on Caltech Pedestrian by ourselves, instead of the CNNs re-engineered for pedestrians [294, 311, 312]. TwoStream still provided significant improvement of MR by 1.2%, and this suggests that our deep motion features can improve the state-of-the-art detectors with two-stage pipelines.

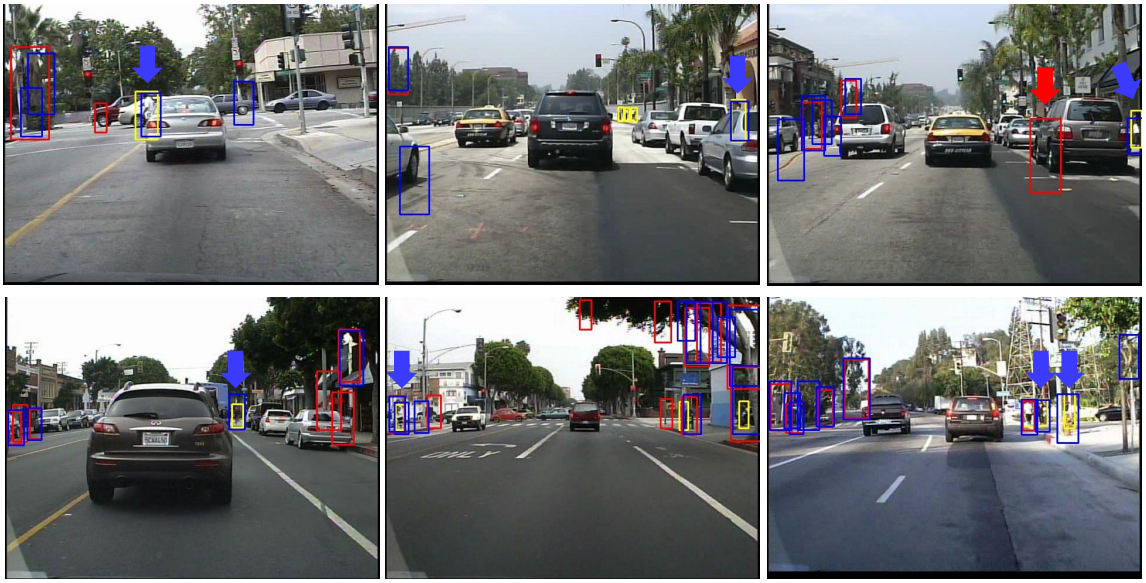
Some very recent deep-learning based methods outperformed ours in Caltech Pedestrian by using more complex networks. Three deep methods without motion, namely DeepParts (11.9% MR) [294], CompACT-Deep (11.7% MR) [311], and SA-FastRCNN (9.7% MR) [312] outperformed TwoStream, because of the differences in the frameworks and the learning methodology. Specifically, one main difference is that these methods adopted two-stage architecture, i.e, first generate candidate bounding boxes by other detectors and then re-categorize them by second-stage convnets. They also introduced techniques to improve second-stage convnets, such as part mining [294], cascading with shallow features [311], or scale adaptation [312]. The other difference is that the convnets in them are pre-trained using ILSVRC2014-DET, and then fine-tuned by Caltech Pedestrian. TwoStream realizes simpler system, as it does not require other detectors for object proposals; however, it may improve by updating the ImageNet dataset for pre-training, and introducing fine-tuning over pedestrian datasets. In addition, TwoStream can alternate the sliding-window detectors in those deep methods, and this would further improve overall performance, since TwoStream can detect some pedestrians that the above methods miss due to visual obscurity or hard blur, as shown in Figure 4.11. Specifically, [294] and [312] used LDCF [307], and [311] used ACF [44] + LDCF [307] + CheckerBoard [50] as their first-stage detectors, and ours outperformed all of them. Combination of ours and the state-of-the art approaches are promising, since our contribution in motion feature learning is orthogonal to the techniques to improve second-stage convnets, used in [294, 311, 312] to achieve better performances. End-to-end deep-learning-based approaches [313, 314] also work well in Caltech, but they require full-frame annotated training data; thus, are not applicable to datasets that only provide pre-cropped training windows such as DaimlerMono.

4.5 Conclusion

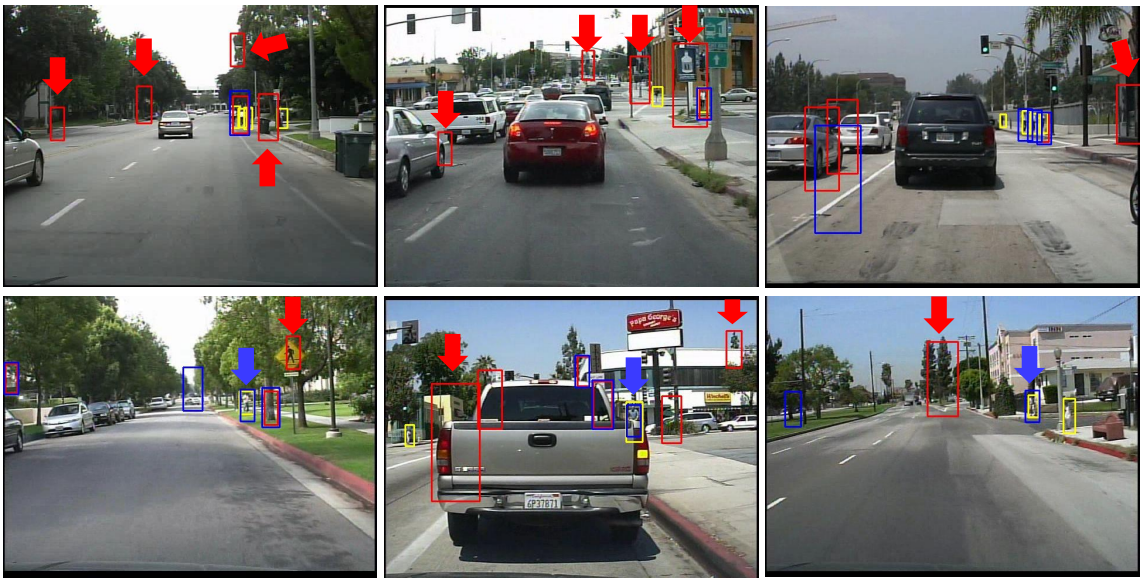
We demonstrated a method for exploiting motion information in pedestrian detection with deep learning. With our method, we fused the ideas from a two-stream network architecture for activity recognition and convolutional channel features for pedestrian detection. In the experiments on the Caltech Pedestrian Detection Benchmark and Daimler Mono Pedestrian Detection Benchmark, we achieved a reasonable decrease of detection miss rate compared to existing convolutional-network pedestrian detectors, and the analyses revealed that the motion feature improved detection in recognizing hard examples, which even state-of-the-art detectors fail to discriminate.

other video-based localization tasks, such as generic object detection in video and video segmentation.

(a) Examples of finding hard positives



(b) Examples of suppressing hard negatives



Red: CCF Blue: TwoStream (ours) Yellow: ground truth

Figure4.6: Detection examples and comparison with CCF (baseline) and TwoStream (ours). TwoStream detects more pedestrians, including hard-to-detect ones, with help of temporal information (a). TwoStream is also more robust against hard negative samples than CCF (b).

Figure4.7: Patches scored with TwoStream and single-frame baseline CCF. Numbers below each samples indicate TwoStream score, CCF score, and their difference, respectively. (A) Negative patches scored lower with TwoStream, which were more confidently rejected by motion information. (B) Positive patches scored higher with TwoStream. Pedestrians not clear due to blur or low contrast were more robustly detected. (C) and (D) show contrary cases.

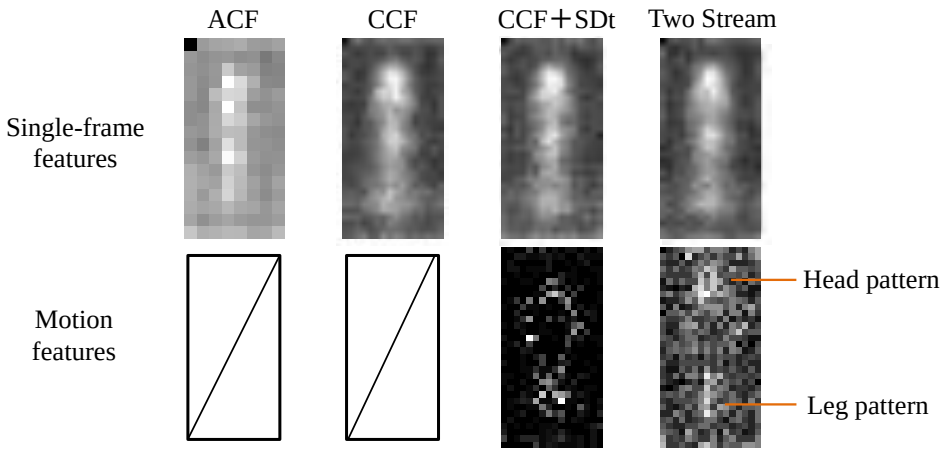


Figure4.8: Visualization of learned pedestrian filters via boosted forests: frequency distributions of features selected by boosting on pedestrian windows.

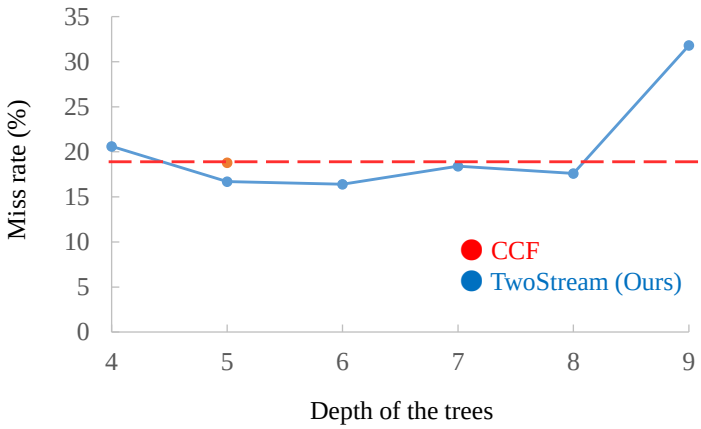


Figure4.9: Relationship between tree depth in boosted forests and log-average miss rate in Caltech *Reasonable* subset. TwoStream consistently outperformed the baseline with depth of 5 to 8; thus was robust against this parameter, while overly shallow (less than 4) or deep (over 9) trees degraded detection performance.

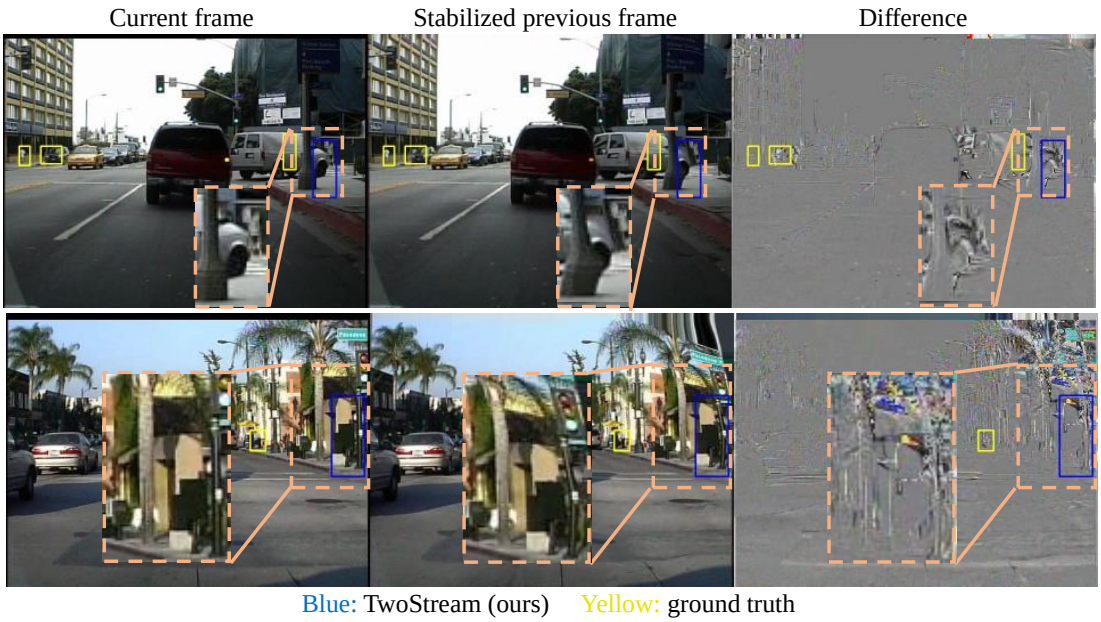


Figure4.10: Failure cases. In these frames, TwoStream caused new misdetections that CCF did not misdetected, due to improperly stabilized motions by inaccurate optical flow.

Figure4.11: Comparison with state-of-the-art detectors. Our motion-based detector succeeded in detecting pedestrians that were missed with both DeepParts and CompACT-Deep. Thus, ours can be complementary for these motion-free deep detectors.

5

Detection and tracking

（未発表部分の為省略）

6

Novelty-tolerant detection

（未発表部分のため省略）

7

Conclusion

7.1 Summary of contributions

This thesis discussed the way to enhance the robustness of deep-learning-based object detection for 'in-the-wild' usages such as wide-area bird surveillance.

In Chapter 3, *Wide-area bird surveillance: Data construction and analysis*, we constructed the image and video datasets bird-surveillance. We analyzed the datasets to understand the novel challenges that were cast by the bird-surveillance tasks, and we implemented and evaluated existing popular recognition methods. The datasets were the first large-scale ones in the computer vision community that focused on the bird surveillance. At the same time, these datasets are useful for the wind-energy industry to develop and evaluate measures against bird strikes.

In Chapter 4, *Detection by motion feature learning*, we developed an object-detection method that exploit deeply learned motion features via two-stream CNNs. While the two-stream CNN is a simple idea to incorporate motion features in CNNs, it is useful to investigate whether motion can help detection, and we successfully validated the usefulness of motion. Our method the first adopting learning-based motion features in the field of pedestrian detection, which has been intensively researched. Our detector showed significant performance gains against single-frame-based detectors in pedestrian detection, which is highly a competitive research area. The two-stream CNNs also improved bird detection.

In Chapter 5, *Detection & tracking*, we further advanced the usage of motion in detection. To enable exploitation of longer-term motion than by two-stream, we introduced the recurrent correlation network (RCN), which jointly performs detection and tracking cooperatively. The RCN is the first deep recurrent network that performs joint detection and tracking, while there exist some other recurrent networks. With the RCN, we found that detection and tracking mutually aids each other and as a result large performance improvement was achieved than with the simpler two-stream CNNs in bird detection.

In Chapter 6, *Novelty-tolerant detection*, we discussed the necessity to handle 'unknown' objects for detection in wild. First we develop CRSOR, the first deep open-set classification method that exploits classification-reconstruction learning. Next, we applied CRSOR to object detection in unknown environments, which is a challenging task for current supervised detectors. In such environments, our novelty-tolerant detector empowered by CROSR performs better.

7.2 Outlook for the future

AI for ecology Our work in bird surveillance was done as a part of the measures against bird strike in wind farms [315]. Recently, there exist more movements to introducing AI and machine vision technologies in ecological purposes such as nature monitoring [4, 316], which would be promising direction with high-performing and robust deep learning, and should be pursued further. However, the ultimate goals are not only monitoring but taking actions for nature conservation. To achieve such goals, vision is not sufficient. For taking possible measure against ecological impacts, higher level reasoning and a decision-making mechanism is necessary. While machine vision can be a help for such higher-level thinking, current AIs are not at the stage of taking the overall process by themselves. For example, in the wind-energy industry, operators consider stopping or slowing down the turbines' rotation when endangered species are approaching [260]. To enable the direct connection between AIs and large-scale mechanical systems, machines need to estimate the collision risk and make decisions to balance economic loss and environmental impact.

Applications of small-object surveillance While our focus was on bird-surveillance, we also notice that it has a lot of properties common with UAV detection, which is attracting industrial interests for logistics, delivery, or security. The technologies we discussed may be influential to safe UAV operation, by detecting UAVs that are out of control or illegally used. For example, a system for UAV detection [317] have already used our bird datasets as a part of negative samples and reduced their misdetection.

Toward deeper understanding of the intelligence Our approaches in bird detection still rely on supervised learning based on large-scale labeled datasets, which is similar to 'brute force' solution of problems, and might be intellectually unsatisfying. Currently, more and more researchers started to be attracted by weakly-, semi-, un-, or self-supervised learning, where machines look like acquiring knowledge by themselves. Those learning methods might not be practically useful so soon, but they would be the basis for deeper understanding of intelligence. Our novelty-tolerant detection framework also can be a step toward more intellectual and intellectually interesting learning machines that perform self-motivated learning, since knowing the existence of unknowns is the source of intellectual modesty and curiosity [318, 319].

謝辞

本研究を進め，博士学位論文として纏めるにあたり，大変多くの方々にご指導とご支援を賜りました．この場をお借りしまして，私の研究を支えてくださった皆さまに，改めて深く御礼申し上げます．

指導教員の苗村 健 教授には，研究の方向性や進め方を懇切丁寧にご指導いただいたのみならず，時には研究への姿勢や社会人としての振舞まで，数えきれないほど多くの事を御指導いただきました．心より御礼申し上げます．先端科学技術研究センター　飯田 誠 特任准教授には野鳥に関するテーマの提供および日ごろから研究のご指導・助言をいただきました．心より御礼申し上げます．佐藤洋一教授，佐藤真一教授，山崎俊彦准教授には学位論文審査に置きまして貴重な御指導・御助言をいただきまして，心より御礼申し上げます．

川上 玲 講師，尤　少迪　講師（現 Data61-CSIRO, オーストラリア国立大学）には，研究の方針や評価の手法，実装，論文の添削，発表の仕方に至るまで懇切丁寧な御指導を頂きました．心より御礼申し上げます．会田大也 特任助教，小泉 直也 准教授（現 電気通信大学），福嶋 政期 助教，アリ ハウタサーリ 助教，阪口 紗季 研究員 には，ミーティングの場などで的確なアドバイスをいただき，研究を進めていく上で大変参考になりました．心より御礼申し上げます．

野鳥データセットの作成にあたり，日本気象協会からのご支援，NPO 法人バードリサーチ　植田睦之代表からのご協力をいただいています．心より御礼申し上げます．．また植田代表の呼びかけに応え，データの作成作業に参加してくださった皆様に深く感謝申し上げます．

秘書 土田 有里 氏には，大変素晴らしい研究環境をご提供いただきました．お陰様で，安心して研究に取り組むことができました．心より御礼申し上げます．

同じ研究グループのメンバーとして，日ごろから研究に関して議論やアドバイスをしていただいた，Trinh Tuan Tu 氏，Tran Hoang Ba 氏，竹木 章人 氏，福田 誠一郎 氏，王 晋 氏, 森脇 健太 氏，　王 亦楠 氏，　邵 文 氏，　加地 佑季 氏，　児玉 悠斗 氏，　田口 航平 氏には心より御礼申し上げます．卒業された先輩方である中島 諒 氏，谷合 竜典 氏，キム ボヨン 氏，樋爪　真子，長谷川 大起 氏からも様々なアドバイスをいただいており，心より御礼申し上げます．またハン チャンギョ 氏，上久保 竜輝 氏には当博士論文に関する助言をいただき, 心より御礼申し上げます．

研究室の同期として共に過ごしてきた高橋 一成氏，長徳 将希氏，中村 鮎葉氏，平木 剛史氏，山本 紘暉氏，吉田 夏子氏に心より御礼申し上げます．

またいかなる時にも私に支援，激励を下さった家族の皆さまに，心より御礼申し上げます．

最後に，博士課程での研究をご支援くださった日本学術振興会（JSPS 科研費 JP16J04552）に心より御礼申し上げます．

2018 年 12 月 7 日

吉橋 亮太

Publications

7.3 Publications related to the thesis

Journal Papers

[A] Ryota Yoshihashi, Trinh Tu Tuan, Rei Kawakami, Shaodi You, Makoto Iida, Takeshi Naemura, "Pedestrian detection with motion features via two-stream ConvNets," IPSJ Transactions on Computer Vision and Applications (CVA) 2018 10:12

[B] Ryota Yoshihashi, Rei Kawakami, Makoto Iida, Takeshi Naemura, " Bird detection and species classification with time-lapse images around a wind farm: Dataset construction and evaluation," Wind Energy, Wiley and Sons, Ltd, 2017, 20.12: pp. 1983–1995..

International conferences

[C] Ryota Yoshihashi, Rei Kawakami, Makoto Iida, Takeshi Naemura, "Construction of a Bird Image Dataset for Ecological Investigations," IEEE International Conference on Image Processing (ICIP2015), pp. 4248 – 4252 (2015.9).

[D] Ryota Yoshihashi, Rei Kawakami, Makoto Iida, Takeshi Naemura, "Evaluation of Bird Detection using Time-lapse Images around a Wind Farm," European Wind Energy Accosiation Conference (EWEA) 2015, oral.

[E] Trinh Tu Tuan, Ryota Yoshihashi, Rei Kawakami, Shaodi You, Makoto Iida, Takeshi Naemura, "Bird detection near wind turbines from high-resolution video using LSTM networks," World Wind Energy Conference (WWEC), 2016.10.

[F] Ryota Yoshihashi, Wen Shao, Rei Kawakami, You Shaodi, Makoto Iida, Takeshi Naemura, "Classification-Reconstruction Learning for Open-Set Recognition," under review.

Domestic conferences and technical reports

[G] 吉橋　亮太，チン　トゥー　トゥアン，川上　玲, 尤 少迪, 飯田　誠, 苗村　健，"動き表現の深層学習に基づく小物体の検出と追跡"，画像の理解・認識シンポジウム（MIRU2018）

[H] Ryota Yoshihashi, Trinh Tu Tuan, Rei Kawakami, Shaodi You, Makoto Iida, Takeshi Naemura, "Differentiating Objects by Motion: Joint Detection and Tracking of Small Flying Objects," arXiv preprint:1709.04666 (2017)

[I] チン　トゥー　トゥアン，吉橋　亮太，川上　玲, 尤 少迪, 飯田　誠, 苗村　健，“動物体検出のための LSTM ネットワークによる静止画と動き情報の統合”, 電子情報通信学会技術研究報告 PRMU 研究会，Vol.116, No.528, p.221-226，2017.3.

[J] 吉橋亮太，川上玲，飯田誠，苗村健: “野鳥の生態調査のための画像データセットの構築,” 画像センシングシンポジウム (SSII2014), IS1-12.

7.4 Publications not related to the thesis

Journal Papers

[K] Akito Takeki, Tu Trinh Tuan, Ryota Yoshihashi, Rei Kawakami, Makoto Iida, Takeshi Naemura, “Combining Deep Features for Object Detection at Various Scales: Finding Small Birds in Landscape Images,” IPSJ Transactions on Computer Vision and Application (CVA) Vol. 8, No. 5, 2016.

[L] Wen Shao, Rei Kawakami, Ryota Yoshihashi, Shaodi You, Hidemichi Kawase, Takeshi Naemura: “Cattle detection and counting in UAV images based on convolutional neural networks,” International Journal of Remote Sensing (under review).

International conferences

[M] Akito Takeki, Tu Trinh Tuan, Ryota Yoshihashi, Rei Kawakami, Makoto Iida, Takeshi Naemura: “Detection of Small Birds in Large Images by Combining a Deep Detector with Semantic Segmentation,” IEEE International Conference on Image Processing (ICIP2016), pp. 3977–3981 (2016.9).

[N] Kenta Moriwaki, Ryota Yoshihashi, Rei Kawakami, You Shaodi, Takeshi Naemura, “Hybrid Loss for Learning Single-Image-based HDR Reconstruction,” under review.

Domestic conferences and technical reports

[O] 王 亦楠, 吉橋 亮太, 川上 玲, 尤 少迪, 原野 徹, 伊藤 昌彦, 駒込 桂, 飯田 誠, 苗村 健, “風車ブレードのドローンによる空撮画像における異常検知手法の検討,” , 第 40 回風力エネルギー利用シンポジウム (2018.12).

[P] Seiichiro Fukuda, Ryota Yoshihashi, Rei Kawakami, Shaodi You, Makoto Iida, Takeshi Naemura: “Cross-connected Networks for Multi-task Learning of Detection and Segmentation,” arXiv preprint:1805.05569 (2018)

[Q] 邵 文, 福田 誠一郎, 吉橋 亮太, 川上 玲, 尤 少迪, 川瀬 英路, 苗村 健, “放牧支援のための空撮画像における CNN に基づく牛検出,” 映像メディア処理シンポジウム (IMPS2017), P-1-6 (2017.11).

[R] 王 晋, 福田 誠一郎, 吉橋 亮太, 川上 玲, 川瀬 英路, 苗村 健, “畜産業支援に向けたドローンによる空撮画像の撮影と牛検出への応用,” 第 23 回画像センシングシンポジウム (SSII2017), IS1-07 (2017.6).

[S] チャン　ホァン　バ, 吉橋 亮太, 川上 玲, 尤 少迪, 川瀬 英路, 苗村 健，“複雑な背景や変形を考慮したロバストな鳥の自動追跡,” 第 23 回画像センシングシンポジウム (SSII2016), IS2-35， 2016.6.

[T] 福田 誠一郎, 吉橋 亮太, 川上 玲, 飯田 誠, 苗村 健, “鳥検出のための矩形領域を教師情報とした CNN による領域分割,” 映像メディア学会 年次大会, 24D-4 (2016.09).

[U] 竹木 章人, チン トゥアン トゥー, 吉橋 亮太, 川上 玲, 飯田 誠, 苗村 健, “生態調査に向けた領域分割と検出器の組み合わせによる鳥画像検出,” 映像情報メディア学会年次大会 (ITE2015), 13C-2.

Bibliography

[1] R. Szeliski, *Computer vision: algorithms and applications*, Springer Science & Business Media, 2010.

[2] D. Park, C. Zitnick, D. Ramanan, and P. Dollar, "Exploring Weak Stabilization for Motion Feature Extraction," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[3] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[4] M. J. Wilber, W. J. Scheirer, P. Leitner, B. Heflin, J. Zott, D. Reinke, D. K. Delaney, and T. E. Boult, "Animal recognition in the mojave desert: Vision tools for field biologists," in *WACV*. 2013, pp. 206–213, IEEE.

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[6] G. Pasquale, C. Ciliberto, F. Odone, L. Rosasco, and L. Natale, "Are we Done with Object Recognition? The iCub robot's Perspective," *arXiv preprint arXiv:1709.09882*, 2017.

[7] A. Dantcheva, P. Elia, and A. Ross, "What else does your biometric data reveal? A survey on soft biometrics," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 3, pp. 441–467, 2016.

[8] Y. LeCun and G. Bengio, Y.and Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[9] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[10] D. G. Lowe, "Object recognition from local scale-invariant features," in *International Conference on Computer Vision (ICCV)*, 1999, vol. 2, pp. 1150–1157.

[11] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[12] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[13] N. Sunderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, and others, "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[14] 竹内啓 and others, **統計学辞典**, 東洋経済新聞社, 1989.

[15] "flickr," *http://www.flickr.com/*, p. 2016/02/03 アクセス.

[16] H. Nakayama, "Linear Distance Metric Learning for Large-scale Generic Image Recognition," *Ph. D dissertation, The University of Tokyo*.

[17] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACMMM*, 2014, pp. 675–678.

[18] "ImageNet: Cellular telephone, cellular phone, cellphone, cell, mobile phone, url: http://image-net.org/synset?wnid=n02992529, accesses 2018-11-27.

[19] E. Brynjolfsson, Y. Hu, and D. Simester, "Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales," *Management Science*, vol. 57, no. 8, pp. 1373–1386, 2011.

[20] "写真に写った食べ物を自動認識してカロリーを表示するアプリがザル過ぎて面白い「声優があげた写真にカロリーを伝える人の方が正確そう」, togetter, url: https://togetter.com/li/1263789, accessed 2018-10-02.

[21] 米谷竜, 斎藤英雄, 池畑諭, 牛久祥孝, 内山英昭, 内海ゆづ子, 小野峻佑, 片岡裕雄, 金崎朝子, 川西康友, 齋藤真樹, 櫻田健, 高橋康輔, and 松井勇佑, コンピュータビジョン一広がる要素技術と応用一, 共立出版.

[22] G. Dinneen, "Programming pattern recognition," in *Proceedings of the March 1-3, 1955, western joint computer conference*. 1955, pp. 94–100, ACM.

[23] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962.

[24] Y. LeCun, "The MNIST database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*.

[25] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "FVC2000: Fingerprint verification competition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, , no. 3, pp. 402–412, 2002.

[26] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid, "BlitzNet: A real-time deep network for scene understanding," in *International Conference on Computer Vision (ICCV)*, 2017, p. 11.

[27] I. Kokkinos, "UberNet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory.," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 2, p. 8.

[28] S. Fukuda, R. Yoshihashi, R. Kawakami, S. You, M. Iida, and T. Naemura, "Cross-connected Networks for Multi-task Learning of Detection and Segmentation," *arXiv preprint arXiv:1805.05569*, 2018.

[29] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, "Simultaneous detection and segmentation," in *European Conference on Computer Vision (ECCV)*. 2014, pp. 297–312, Springer.

[30] Y. Li, H. Qi, J. Dai, X. Ji, and Y. Wei, "Fully convolutional instance-aware semantic segmentation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[31] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2411–2418.

[32] J. M. M. F. R. P. L. e. T. V. G. H. A. L. Matej Kristan, Aleš Leonardis and G. Fernandez, "The Visual Object Tracking VOT2016 challenge results," *ECCVW*, 2016.

[33] A. Milan, L. Leal-Taixe, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *arXiv preprint arXiv:1603.00831*, 2016.

[34] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang, and S. Lyu, "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *arXiv preprint arXiv:1511.04136*, 2015.

[35] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua, "Multiple object tracking using k-shortest paths optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, no. 9, pp. 1806–1819, 2011.

[36] A. A. Butt and R. T. Collins, "Multi-target tracking by lagrangian relaxation to min-cost network flow," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1846–1853.

[37] D. Lowe, "Distinctive image features from scaleinvariant keypoints," *International Journal on Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[38] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *International Conference on Computer Vision (ICCV)*. 2003, p. 1470, IEEE.

[39] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual Categorization with Bags of Keypoints," *Workshop on statistical learning in computer vision, European Conference on Computer Vision*, 2004.

[40] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, vol. 1, pp. 511–518.

[41] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, vol. 1, pp. 886–893.

[42] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *International Conference on Computer Vision (ICCV)*, 2005, vol. 1, pp. 90–97.

[43] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New features and insights for pedestrian detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, pp. 1030–1037.

[44] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 36, no. 8, pp. 1532–1545, 2014.

[45] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on riemannian manifolds," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 10, pp. 1713–1727, 2008.

[46] H. Ren and Z.-N. Li, "Object Detection Using Generalization and Efficiency Balanced Co-Occurrence Features," in *International Conference on Computer Vision (ICCV)*, pp. 46–54.

[47] T. Ojala, M. Pietikainen, and T. Maenpaa, "Gray scale and rotation invariant texture classification with local binary patterns," in *European Conference on Computer Vision (ECCV)*, 2000, pp. 404–420.

[48] X. Wang, T. Han, and S. Yan, "An HOG-LBP human detector with partial occlusion handling," in *International Conference on Computer Vision (ICCV)*, 2009, pp. 32–39.

[49] E. Ohn-Bar and M. M. Trivedi, "To boost or not to boost? On the limits of boosted trees for object detection," in *International Conference on Pattern Recognition (ICPR)*. 2016, pp. 3350–3355, IEEE.

[50] S. Zhang, R. Benenson, and B. Schiele, "Filtered Feature Channels for Pedestrian Detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[51] I. Binford, "Visual perception by computer," in *IEEE Conference of Systems and Control*, 1971.

[52] S. A. Shafer and T. Kanade, "The theory of straight homogeneous generalized cylinders," Tech. Rep., CARNEGIE INST OF TECH PITTSBURGH PA DEPT OF COMPUTER SCIENCE, 1983.

[53] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[54] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[55] S. McCann and D. G. Lowe, "Local naive bayes nearest neighbor for image classification," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3650–3656, IEEE.

[56] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 891–923, 1998.

[57] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *The thirtieth annual ACM symposium on Theory of computing*. 1998, pp. 604–613, ACM.

[58] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*. 2006, pp. 459–468, IEEE.

[59] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 33, no. 1, pp. 117–128, 2011.

[60] Y. Matsui, T. Yamasaki, and K. Aizawa, "Pqtable: Fast exact asymmetric distance neighbor search for product quantization using hash tables," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1940–1948.

[61] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[62] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, pp. 386, 1958.

[63] C. Corinna and V. Vapnik, "Support-Vector Networks," *Machine Learning*, vol. 20, no. 3, pp. 421–436, 1995.

[64] D. Dheeru and E. Karra Taniskidou, *UCI Machine Learning Repository*, University of California, Irvine, School of Information and Computer Sciences, 2017.

[65] B. Scholkopf, "The kernel trick for distances," in *Advances in Neural Information Processing Systems (NIPS)*, 2001, pp. 301–307.

[66] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[67] J. R. Quinlan, *C4. 5: programs for machine learning*, Elsevier, 2014.

[68] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[69] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine learning*, vol. 63, no. 1, pp. 3–42, 2006.

[70] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of On-Line Learing and an Application to Boosting," *Computational Learning Theory*, vol. 904, pp. 23–37, 1995.

[71] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[72] T. Chen, T. He, M. Benesty, and others, "Xgboost: extreme gradient boosting," *R package version 0.4-2*, pp. 1–4, 2015.

[73] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *KDD*. 2016, pp. 785–794, ACM.

[74] R. Appel, T. Fuchs, P. Dollar, and P. Perona, "Quickly boosting decision trees-pruning underachieving features early," in *International Conference on Machine Learning (ICML)*, 2013, vol. 28, pp. 594–602.

[75] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[76] G. E. Hinton, J. L. McClelland, D. E. Rumelhart, and others, *Distributed representations*.

[77] B. Yang, J. Yan, Z. Lei, and S. Li, "Convolutional Channel Features For Pedestrian, Face and Edge Detection," in *International Conference on Computer Vision (ICCV)*, 2015.

[78] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulo, "Deep neural decision forests," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1467–1475.

[79] P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural networks*, vol. 1, no. 4, pp. 339–356, 1988.

[80] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 20, no. 1, pp. 23–38, 1998.

[81] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," *Proceedings of the IEEE*, 1998.

[82] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, pp. 267–285. Springer, 1982.

[83] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[84] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 2, p. 7.

[85] A. Roy and S. Todorovic, "Monocular depth estimation using neural regression forest," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 5506–5514.

[86] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 2758–2766.

[87] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "FlowNet 2.0: Evolution of optical flow estimation with deep networks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[88] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification," *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, vol. 35, no. 4, pp. 110:1–110:11, 2016.

[89] C. Dong, C. C. Loy, K. He, and X. Tang, "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 38, no. 2, pp. 295–307, Feb. 2016.

[90] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European Conference on Computer Vision (ECCV)*. 2016, pp. 391–407, Springer.

[91] K. Lu, S. You, and N. Barnes, "Deep Texture and Structure Aware Filtering Network for Image Smoothing," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 217–233.

[92] R. Furuta, N. Inoue, and a. T. Yamasaki, "Fully Convolutional Network with Multi-Step Reinforcement Learning for Image Processing," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[93] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[94] M. I. Jordan, "Serial order: A parallel distributed processing approach," in *Advances in psychology*, vol. 121, pp. 471–495. Elsevier, 1997.

[95] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, and others, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*, A field guide to dynamical recurrent neural networks. IEEE Press, 2001.

[96] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[97] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[98] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2015, pp. 2067–2075.

[99] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[100] S. Sukhbaatar, J. Weston, R. Fergus, and others, "End-to-end memory networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2440–2448.

[101] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, and others, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471, 2016.

[102] G. Melis, C. Dyer, and P. Blunsom, "On the state of the art of evaluation in neural language models," *arXiv preprint arXiv:1707.05589*, 2017.

[103] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," Tech. Rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[104] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller, "Efficient backprop," in *Neural networks: Tricks of the trade*, pp. 9–48. Springer, 2012.

[105] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," 2011, pp. 315–323.

[106] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International Conference on Learning Representations (ICLR)*, 2013.

[107] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *International Conference on Learning Representations (ICLR)*, 2016.

[108] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 971–980.

[109] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[110] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper With Convolutions," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[111] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2377–2385.

[112] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks.," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 1, p. 3.

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

[114] P. Baldi and P. J. Sadowski, "Understanding dropout," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 2814–2822.

[115] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using DropConnect," in *International Conference on Machine Learning (ICML)*, 2013, pp. 1058–1066.

[116] T. Yamashita, M. Tanaka, Y. Yamauchi, and H. Fujiyoshi, "SWAP-NODE: A regularization approach for deep convolutional neural networks," in *IEEE International Conference on Image Processing (ICIP)*. 2015, pp. 2475–2479, IEEE.

[117] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with Cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[118] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[119] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[120] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 901–909.

[121] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: the missing ingredient for fast stylization," *arXiv preprint:1607.08022*.

[122] Y. Wu and K. He, "Group normalization," 2018.

[123] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[124] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal on Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[125] P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 328–335.

[126] C. L. Zitnick and P. Dollar, "Edge boxes: Locating object proposals from edges," in *European Conference on Computer Vision (ECCV)*. 2014, pp. 391–405, Springer.

[127] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Susstrunk, and others, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 11, pp. 2274–2282, 2012.

[128] J. Hosang, M. Omran, R. Benenson, and B. Schiele, "Taking a deeper look at pedestrians," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[129] R. Girshick, "Fast R-CNN," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[130] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[131] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[132] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[133] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv*, 2018.

[134] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*. 2016, pp. 21–37, Springer.

[135] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and others, "Speed/accuracy trade-offs for modern convolutional object detectors," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 4.

[136] O. Ronneberger, P.Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. 2015, vol. 9351 of *LNCS*, pp. 234–241, Springer.

[137] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD: Deconvolutional single shot detector," *arXiv preprint arXiv:1701.06659*, 2017.

[138] T.-Y. Lin, P. Dollar, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature Pyramid Networks for Object Detection.," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, vol. 1, p. 4.

[139] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the effective receptive field in deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4898–4906.

[140] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2881–2890.

[141] X. Zeng, W. Ouyang, J. Yan, H. Li, T. Xiao, K. Wang, Y. Liu, Y. Zhou, B. Yang, Z. Wang, and others, "Crafting GBD-net for object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 40, no. 9, pp. 2109–2123, 2018.

[142] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy, and others, "Deepid-net: Deformable deep convolutional neural networks for object detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 2403–2412.

[143] M. Jaderberg, K. Simonyan, A. Zisserman, and others, "Spatial transformer networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2017–2025.

[144] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *International Conference on Computer Vision (ICCV)*, 2017, vol. 1, p. 3.

[145] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object Detection with Discriminatively Trained Part-based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[146] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 761–769.

[147] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2018.

[148] Z. Cai and N. Vasconcelos, "Cascade R-CNN: Delving into High Quality Object Detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[149] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun, "MegDet: A large mini-batch object detector," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[150] "Megvii, url: https://megvii.com/, accessed 2018-12-06.

[151] T. Akiba, T. Kerola, Y. Niitani, T. Ogawa, S. Sano, and S. Suzuki, "PFDet: 2nd Place Solution to Open Images Challenge 2018 Object Detection Track," *arXiv preprint arXiv:1809.00778*, 2018.

[152] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. Duerig, and others, "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale," *arXiv preprint arXiv:1811.00982*, 2018.

[153] "株式会社 preferred networks, url: https://www.preferred-networks.jp/, acessed 2018-12-06.

[154] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[155] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in *NIPSW*, 2015, vol. 5, pp. 1–6.

[156] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4293–4302.

[157] K. Li, Y. Kong, and Y. Fu, "Multi-Stream Deep Similarity Learning Networks for Visual Tracking," *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[158] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. Torr, "Fully-convolutional siamese networks for object tracking," in *ECCVW*, 2016, pp. 850–865.

[159] J. Valmadre, L. Bertinetto, J. F. Henriques, A. Vedaldi, and P. H. Torr, "End-to-end representation learning for Correlation Filter based tracking," *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[160] G. Ning, Z. Zhang, C. Huang, Z. He, X. Ren, and H. Wang, "Spatially supervised recurrent convolutional neural networks for visual object tracking," in *IEEE International Symposium on Circuits and Systems*, 2017.

[161] D. Gordon, A. Farhadi, and D. Fox, "Re $^ 3$: Re al-Time Recurrent Regression Networks for Visual Tracking of Generic Objects," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 788–795, 2018.

[162] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler, "Online Multi-Target Tracking Using Recurrent Neural Networks.," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2017, pp. 4225–4232.

[163] L. Wang, L. Zhang, and Z. Yi, "Trajectory Predictor by Using Recurrent Neural Networks in Visual Tracking," *IEEE Transactions on Cybernetics*, 2017.

[164] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[165] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 7, pp. 1409–1422, 2010.

[166] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[167] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses," in *European Conference on Computer Vision (ECCV)*. 2008, pp. 788–801, Springer.

[168] J. Han, L. Shao, D. Xu, and J. Shotton, "Enhanced computer vision with microsoft kinect sensor: A review," *IEEE transactions on cybernetics*, vol. 43, no. 5, pp. 1318–1334, 2013.

[169] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," in *ICRA*. 1991, pp. 1088–1095, IEEE.

[170] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, no. 7, pp. 787–800, 2003.

[171] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005, vol. 2, pp. 807–814, IEEE.

[172] T. Taniai, Y. Matsushita, Y. Sato, and T. Naemura, "Continuous 3d label stereo matching using local expansion moves," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 40, no. 11, pp. 2725–2739, 2018.

[173] C. Dai, Y. Zheng, and X. Li, "Pedestrian detection and tracking in infrared imagery using shape and appearance," *CVIU*, vol. 106, no. 2-3, pp. 288–299, 2007.

[174] J. Herweg, J. Kerekes, and M. Eismann, "Hyperspectral imaging phenomenology for the detection and tracking of pedestrians," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2012, pp. 5482–5485, IEEE.

[175] S. Hwang, J. Park, N. Kim, Y. Choi, and I. So Kweon, "Multispectral pedestrian detection: Benchmark dataset and baseline," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1037–1045.

[176] B. Leibe, E. Seemann, and B. Schiele, "Pedestrian detection in crowded scenes," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005, pp. 878–885, IEEE.

[177] F. Jurie, M. Dhome, and others, "Real Time Robust Template Matching.," in *British Machine Vision Conference (BMVC)*, 2002.

[178] T. Zhang, K. Jia, C. Xu, Y. Ma, and N. Ahuja, "Partial occlusion handling for visual tracking via robust part matching," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1258–1265.

[179] J. Tighe, M. Niethammer, and S. Lazebnik, "Scene parsing with object instances and occlusion ordering," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3748–3755.

[180] P. Baque, F. Fleuret, and P. Fua, "Deep occlusion reasoning for multi-camera multi-target detection," in *International Conference on Computer Vision (ICCV)*, 2017, vol. 2.

[181] P. Sundberg, T. Brox, M. Maire, P. Arbelaez, and J. Malik, "Occlusion boundary detection and figure/ground assignment from optical flow," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 2233–2240, IEEE.

[182] H. Fu, C. Wang, D. Tao, and M. J. Black, "Occlusion boundary detection via deep exploration of context," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 241–250.

[183] P. Dollar and C. L. Zitnick, "Structured forests for fast edge detection," in *International Conference on Computer Vision (ICCV)*, 2013, pp. 1841–1848.

[184] P. Dollar and C. L. Zitnick, "Fast edge detection using structured forests," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 8, pp. 1558–1570, 2015.

[185] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," 2011.

[186] Z. Yu, C. Feng, M.-Y. Liu, and S. Ramalingam, "Casenet: Deep category-aware semantic edge detection," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 21–26.

[187] A. Oliva and A. Torralba, "The role of context in object recognition," *Trends in cognitive sciences*, vol. 11, no. 12, pp. 520–527, 2007.

[188] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3668–3678.

[189] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, and others, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *International Journal on Computer Vision*, vol. 123, no. 1, pp. 32–73, 2017.

[190] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, , no. 6, pp. 721–741, 1984.

[191] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *Departmental Papers (CIS), Department of Computer & Information Science, University of Pennsylvania*, 2001.

[192] X. Cao, X. Wei, Y. Han, and X. Chen, "An object-level high-order contextual descriptor based on semantic, spatial, and scale cues," *IEEE Transactions on Cybernetics*, vol. 45, no. 7, pp. 1327–1339, 2015.

[193] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Object detection refinement using Markov random field based pruning and learning based rescoring," in *ICASSP*, 2017, pp. 1652–1656.

[194] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *International Conference on Learning Representations (ICLR)*, 2015.

[195] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NIP*, 2015, pp. 577–585.

[196] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 1529–1537.

[197] Z. Wu, D. Lin, and X. Tang, "Deep markov random field for image modeling," in *European Conference on Computer Vision (ECCV)*. 2016, pp. 295–312, Springer.

[198] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *European Conference on Computer Vision (ECCV)*, 2006, pp. 428–441.

[199] J. Mao, T. Xiao, Y. Jiang, and Z. Cao, "What Can Help Pedestrian Detection?," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[200] M. Jones and D. Snow, "Pedestrian detection using boosted features over many frames," in *International Conference on Pattern Recognition (ICPR)*, 2008, pp. 1–4.

[201] P. Viola, M. J. Jones, and D. Snow, "Detecting Pedestrians Using Patterns of Motion and Appearance," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[202] P. Massimo, "Background Subtraction Techniques: A Review," *IEEE International Conference on Systems, Man and Cybernetics*, 2004.

[203] L. Wang, Y. Qiao, and X. Tang, "Action recognition with Trajectory-Pooled Deep-Convolutional Descriptors," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[204] G. Gkioxari and J. Malik, "Finding Action Tubes," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[205] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 1, pp. 221–231, 2013.

[206] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[207] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 568–576.

[208] J. Shao, K. Kang, C. C. Loy, and X. Wang, "Deeply Learned Attributes for Crowded Scene Understanding," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[209] G. Cheron, I. Laptev, and C. Schmid, "P-CNN: Pose-Based CNN Features for Action Recognition," in *International Conference on Computer Vision (ICCV)*, 2015.

[210] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge,," *arXiv:1409.0575*, 2014.

[211] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[212] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-Guided Feature Aggregation for Video Object Detection," in *International Conference on Computer Vision (ICCV)*, 2017.

[213] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to Track and Track to Detect," in *International Conference on Computer Vision (ICCV)*, 2017.

[214] T. T. Trinh, R. Yoshihashi, R. Kawakami, M. Iida, and T. Naemura, "BIRD DETECTION NEAR WIND TURBINES FROM HIGH-RESOLUTION VIDEO USING LSTM NETWORKS," in *World Wind Energy Conference (WWEC)*, 2016.

[215] R. Yoshihashi, T. T. Trinh, R. Kawakami, S. You, M. Iida, and T. Naemura, "Differentiating Objects by Motion: Joint Detection and Tracking of Small Flying Objects," *arXiv preprint arXiv:1709.04666*, 2017.

[216] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2013, pp. 1310–1318.

[217] G. J. McLachlan, "Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis," *Journal of the American Statistical Association*, vol. 70, no. 350, pp. 365–369, 1975.

[218] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-Supervised Self-Training of Object Detection Models.," in *WACV*, 2005, pp. 29–36.

[219] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *ICMLf the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[220] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 3581–3589.

[221] L. Maaloe, C. K. Sonderby, S. K. Sonderby, and O. Winther, "Auxiliary deep generative models," in *International Conference on Machine Learning (ICML)*, 2016.

[222] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," *International Conference on Learning Representations (ICLR)*, 2016.

[223] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 3546–3554.

[224] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research (JAIR)*, vol. 26, pp. 101–126, 2006.

[225] A. Margolis, "A literature review of domain adaptation with unlabeled data," 2011.

[226] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation," in *European Conference on Computer Vision (ECCV)*. 2016, pp. 597–613, Springer.

[227] K. Saito, Y. Ushiku, and T. Harada, "Asymmetric tri-training for unsupervised domain adaptation," *International Conference on Machine Learning (ICML)*, 2017.

[228] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[229] K. Saito, K. Watanabe, Y. Ushiku, and T. Harada, "Maximum classifier discrepancy for unsupervised domain adaptation," *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 3, 2018.

[230] L. M. Manevitz and M. Yousef, "One-class SVMs for document classification," *Journal of Machine Learning Research (JMLR)*, vol. 2, no. Dec, pp. 139–154, 2001.

[231] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[232] D. M. Tax and R. P. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004.

[233] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *International Conference on Data Mining (ICDM)*. 2008, pp. 413–422, IEEE.

[234] S. Roberts and L. Tarassenko, "A probabilistic resource allocating network for novelty detection," *Neural Computation*, vol. 6, no. 2, pp. 270–284, 1994.

[235] H. Ringberg, A. Soule, J. Rexford, and C. Diot, "Sensitivity of PCA for traffic anomaly detection," *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 1, pp. 109–120, 2007.

[236] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe, "Learning deep representations of appearance and motion for anomalous event detection," *British Machine Vision Conference (BMVC)*, 2015.

[237] R. Hinami, T. Mei, and S. Satoh, "Joint Detection and Recounting of Abnormal Events by Learning Deep Generic Knowledge.," in *International Conference on Computer Vision (ICCV)*, 2017, pp. 3639–3647.

[238] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. 2017, pp. 665–674, ACM.

[239] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," 2018.

[240] C. Aytekin, X. Ni, F. Cricri, and E. Aksu, "Clustering and Unsupervised Anomaly Detection with L2 Normalized Deep Auto-Encoder Representations," in *IJCNN*, 2018.

[241] P. Perera and V. M. Patel, "Learning Deep Features for One-Class Classification," *arXiv preprint arXiv:1801.05365*, 2018.

[242] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2672–2680.

[243] T. Schlegl, P. Seebock, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International Conference on Information Processing in Medical Imaging*. 2017, pp. 146–157, Springer.

[244] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[245] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *International Conference on Learning Representations (ICLR)*, 2018.

[246] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," in *International Conference on Learning Representations (ICLR)*, 2015.

[247] W. J. Scheirer, L. P. Jain, and T. E. Boult, "Probability models for open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 36, no. 11, pp. 2317–2324, 2014.

[248] G. Fei and B. Liu, "Breaking the Closed World Assumption in Text Classification," in *NAACLHLT*, 2016.

[249] E. Rudd, L. P. Jain, W. J. Scheirer, and T. Boult, "The Extreme Value Machine," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 40, no. 3, Mar. 2017.

[250] P. R. M. Junior, R. M. de Souza, R. d. O. Werneck, B. V. Stein, D. V. Pazinato, W. R. de Almeida, O. A. Penatti, R. d. S. Torres, and A. Rocha, "Nearest neighbors distance ratio open-set classifier," *Machine Learning*, vol. 106, no. 3, pp. 359–386, 2017.

[251] H. Zhang and V. M. Patel, "Sparse representation-based open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 39, no. 8, pp. 1690–1696, 2017.

[252] A. Bendale and T. E. Boult, "Towards open set deep networks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1563–1572.

[253] Z. Ge, S. Demyanov, Z. Chen, and R. Garnavi, "Generative OpenMax for multi-class open set classification," *British Machine Vision Conference (BMVC)*, 2017.

[254] L. Shu, H. Xu, and B. Liu, "DOC: Deep open classification of text documents," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[255] A. L. Drewitt and R. H. W. Langston, "Assessing the Impacts of Wind Farms on Birds," *Ibis–the International Journal of Avian Science*, vol. 148, pp. 29–42, 2006.

[256] W. P. Kuvlesky, L. A. Brennan, M. L. Morrison, B. M. B. K. K. Boydston, and F. C. Bryant, "Wind Energy Development and Wildlife Conservation: Challenges and Opportunities," *The Journal of Wildlife Management*, vol. 71, no. 8, pp. 2487–2498, 2007.

[257] K. S. Smallwood, L. Rugge, and M. L. Morrison, "Influence of Behavior on Bird Mortality in Wind Energy Developments," *The Journal of Wildlife Management*, vol. 73, no. 7, pp. 1082–1098, 2009.

[258] A. L. Drewitt and R. H. W. Langston, "Collision Effects of Wind-power Generators and Other Obstacles on Birds," *Annals of the New York Academy of Sciences*, 2008.

[259] A. L. Drewitt and R. H. W. Langston, "Risk Evaluation for Federally Listed (Roseate Tern, Piping Plover) or Candidate (Red Knot) Bird Species in Offshore Waters: A First Step for Managing the Potential Impacts of Wind Facility Development on the Atlantic Outer Continental Shelf," *Renewable Energy*, 2011.

[260] A. Rioperez and M. d. l. Puente, "DTBird: A Self-working System to Reduce Bird Mortality in Wind Farms," *European Wind Energy Association Conference*, 2010.

[261] M. Desholm, A. D. Fox, P. D. L. Beasley, and J. Kahlert, "Remote techniques for counting and estimating the number of bird-wind turbine collisions at sea: a review," *Ibis - the International Journal of Avian Science*, vol. 148, no. s1, pp. 76–89, 2006.

[262] S. Bassi, A. Bowen, and S. Fankhauser, "The case for and against onshore wind energy in the UK," *Grantham Research Institute on Climate Change and Environment Policy Brief, London*, 2012.

[263] E. A. Masden, D. T. Haydon, A. D. Fox, R. W. Furness, R. Bullman, and M. Desholm, "Barriers to movement: impacts of wind farms on migrating birds," *ICES Journal of Marine Science, 66: 746?753*, 2009.

[264] S. C. Clough, S. McGovern, D. Campbell, and M. M. Rehfisch, "Aerial Survey Techniques for Assessing Offshore Wind Farms," *International Council for the Exploration of the Sea, Conference and Meeting (CM) Documents*, 2012.

[265] E. Wiggelinkhuizen, S. Barhorst, L. Rademakers, H. d. Boon, and S. Dirksen, "WT-BIRD: BIRD COLLISION MONITORING SYSTEM FOR MULTI-MEGAWATT WIND TURBINES," *European Wind Energy Association Conference*, 2007.

[266] "NEC、鳥位置検出ソリューションを国内外で販売開始, 日本電気株式会社, url: https://jpn.nec.com/press/201210/20121011_01.html, acessed 2018-10-02.

[267] "１０億円かけた羽田の鳥衝突防止装置、機能せず.," 読売新聞 オンライン版, *url: https://archive.fo/Pn9or*, 元記事削除のためアーカイブサイトより, Jan. 2015.

[268] S. C. Clough and A. N. Banks, "A 21st Century Approach to Aerial Bird and Mammal Surveys at Offshore Wind Farm Sites," *European Wind Energy Association Conference*, 2011.

[269] V. Jain and E. Learned-Miller, "FDDB: A benchmark for face detection in unconstrained settings," Tech. Rep.

[270] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *CVIU*, vol. 106, no. 1, pp. 59–70, 2007.

[271] B. Russell, A. Torralba, K. Murphy, and W. Freeman, "LabelMe: a Database and Web-based Tool for Image Annotation," *International Journal on Computer Vision*, vol. 77, no. 1-3, pp. 157–173, 2008.

[272] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[273] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," *url: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/*.

[274] P. Welinder, S. Branson, T. Mita, W. C, S. F, S. Belongie, and P. Perona, "Caltech-UCSD Birds 200," 2010.

[275] T. Berg, J. Liu, S. W. Lee, D. W. Alexander, M. L.and Jacobs, and P. N. Belhumeur, "Birdsnap: Large-scale Fine-grained Visual Categorization of Birds," *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[276] M. Everingham, S. M. A. Eslami, L. V. Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge: A Retrospective," *International Journal on Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[277] V. Jain and E. Learned-Miller, "FDDB: A Benchmark for Face Detection in Unconstrained Settings," Tech. Rep. UM-CS-2010-009, University of Massachusetts, Amherst, 2010.

[278] A. Rozantsev, V. Lepetit, and P. Fua, "Detecting flying objects using a single moving camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 39, no. 5, pp. 879–892, 2017.

[279] G. Van Horn, O. Mac Aodha, Y. Song, Y. Cui, C. Sun, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The iNaturalist species classification and detection dataset," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[280] W. Shao, R. Kawakami, Y. Ryota, Y. Shaodi, K. Hidemichi, and N. Takeshi, "Cattle detection and counting in UAV images based on convolutional neural networks," *International Journal of Remote Sensing (under review)*.

[281] 白石卓也, "生態調査・監視のための画像認識を用いた野鳥の検出," 東京大学情報理工学系研究科 修士学位論文, 2012.

[282] 長谷川大起, "生態調査のための画像認識を用いた野鳥出現数調査の基礎検討," 東京大学工学部 電子情報工学科 卒業論文, 2013.

[283] A. Takeki, T. T. Trinh, R. Yoshihashi, R. Kawakami, M. Iida, and T. Naemura, "Detection of small birds in large images by combining a deep detector with semantic segmentation," *IEEE International Conference on Image Processing*, 2016.

[284] A. Takeki, T. T. Trinh, R. Yoshihashi, R. Kawakami, M. Iida, and T. Naemura, "Combining deep features for object detection at various scales: finding small birds in landscape images," *IPSJ transactions on computer vision and applications*, vol. 8, no. 1, pp. 5, 2016.

[285] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[286] R. Yoshihashi, R. Kawakami, M. Iida, and T. Naemura, "Construction of a Bird Image Dataset for Ecological Investigation," *IEEE International Conference on Image Processing*, 2015.

[287] T. Suzuki, H. Kataoka, Y. Aoki, and Y. Satoh, "Anticipating traffic accidents with adaptive loss and large-scale incident db," 2018.

[288] N. Kruger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. J. Rodriguez-Sanchez, and L. Wiskott, "Deep Hierarchies in the Primate Visual Cortex: What Can We Learn For Computer Vision?," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 8, pp. 1847–1871, 2013.

[289] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *International Conference on Computer Vision (ICCV)*, 2007.

[290] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler, "Convolutional learning of spatio-temporal features," in *European Conference on Computer Vision (ECCV)*, 2010.

[291] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[292] M. Hasan and A. K. Roy-Chowdhury, "Continuous learning of human activity models using deep nets," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 705–720.

[293] Y. Tian, P. Luo, X. Wang, and X. Tang, "Pedestrian Detection Aided by Deep Learning Semantic Tasks," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[294] Y. Tian, P. Luo, X. Wang, and X. Tang, "Deep Learning Strong Parts for Pedestrian Detection," in *International Conference on Computer Vision (ICCV)*, 2015.

[295] M. A. Goodale and A. D. Milner, "Separate visual pathways for perception and action," *Trends in neurosciences*, vol. 15, no. 1, pp. 20–25, 1992.

[296] S. Gladh, M. Danelljan, F. S. Khan, and M. Felsberg, "Deep Motion Features for Visual Tracking," in *International Conference on Pattern Recognition (ICPR)*, 2016.

[297] G. Johansson, "Visual perception of biological motion and a model for its analysis," *Perception & psychophysics*, vol. 14, no. 2, pp. 201–211, 1973.

[298] L. Bottou, "Stochastic Gradient Descent Tricks," *Neural Networks: Tricks of the Trade*, pp. 421–436, 2012.

[299] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition," in *International Conference on Machine Learning (ICML)*, 2014.

[300] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision.," in *International Joint Conference on Artificial Intelligence*, 1981, vol. 81, pp. 674–679.

[301] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. T. Freeman, "SIFT flow: Dense correspondence across different scenes," in *European Conference on Computer Vision (ECCV)*, 2008, pp. 28–42.

[302] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *International Conference on Computer Vision (ICCV)*, 2013, pp. 1385–1392.

[303] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 346–361.

[304] C. Dubout and F. Fleuret, "Exact acceleration of linear object detectors," in *European Conference on Computer Vision (ECCV)*, 2012, pp. 301–311.

[305] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 304–311.

[306] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.

[307] W. Nam, P. Dollar, and J. H. Han, "Local decorrelation for improved pedestrian detection," in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 424–432.

[308] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3626–3633.

[309] J. Marin, D. Vazquez, A. M. Lopez, J. Amores, and B. Leibe, "Random forests of local experts for pedestrian detection," in *International Conference on Computer Vision (ICCV)*, 2013, pp. 2592–2599.

[310] W. Nam, B. Han, and J. H. Han, "Improving object localization using macrofeature layout selection," in *ICCVW*. 2011, pp. 1801–1808, IEEE.

[311] Z. Cai, M. Saberian, and V. N, "Learning Complexity-Aware Cascades for Deep Pedestrian Detection," in *International Conference on Computer Vision (ICCV)*, 2015.

[312] J. Li, X. Liang, S. Shen, T. Xu, and S. Yan, "Scale-aware Fast R-CNN for Pedestrian Detection," *arXiv preprint arXiv:1510.08160*, 2015.

[313] L. Zhang, L. Lin, X. Liang, and K. He, "Is faster R-CNN doing well for pedestrian detection?," in *European Conference on Computer Vision*. 2016, pp. 443–457, Springer.

[314] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *European Conference on Computer Vision (ECCV)*. 2016, pp. 354–370, Springer.

[315] "環境省鳥類等に関する風力発電施設立地適正化のための手引き, url: https://www.env.go.jp/nature/yasei/sg_windplant/guide/post_91.html, accessed 2018-12-06.

[316] G. Van Horn, O. Mac Aodha, Y. Song, A. Shepard, H. Adam, P. Perona, and S. Belongie, "The inaturalist challenge 2017 dataset," in *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[317] A. Schumann, L. Sommer, J. Klatte, T. Schuchert, and J. Beyerer, "Deep cross-domain flying object classification for robust UAV detection," in *Advanced Video and Signal Based Surveillance (AVSS)*. 2017, pp. 1–6, IEEE.

[318] プラトン, **ソクラテスの弁明・クリトン**, 三嶋 輝夫, 田中 享英 訳, 講談社学術文庫, 1998.

[319] **論語**, 金谷 治 訳, 岩波文庫, 1999.