Chapter 6

Recognition

© Springer Nature Switzerland AG 2022
R. Szeliski, *Computer Vision*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-34372-9_6

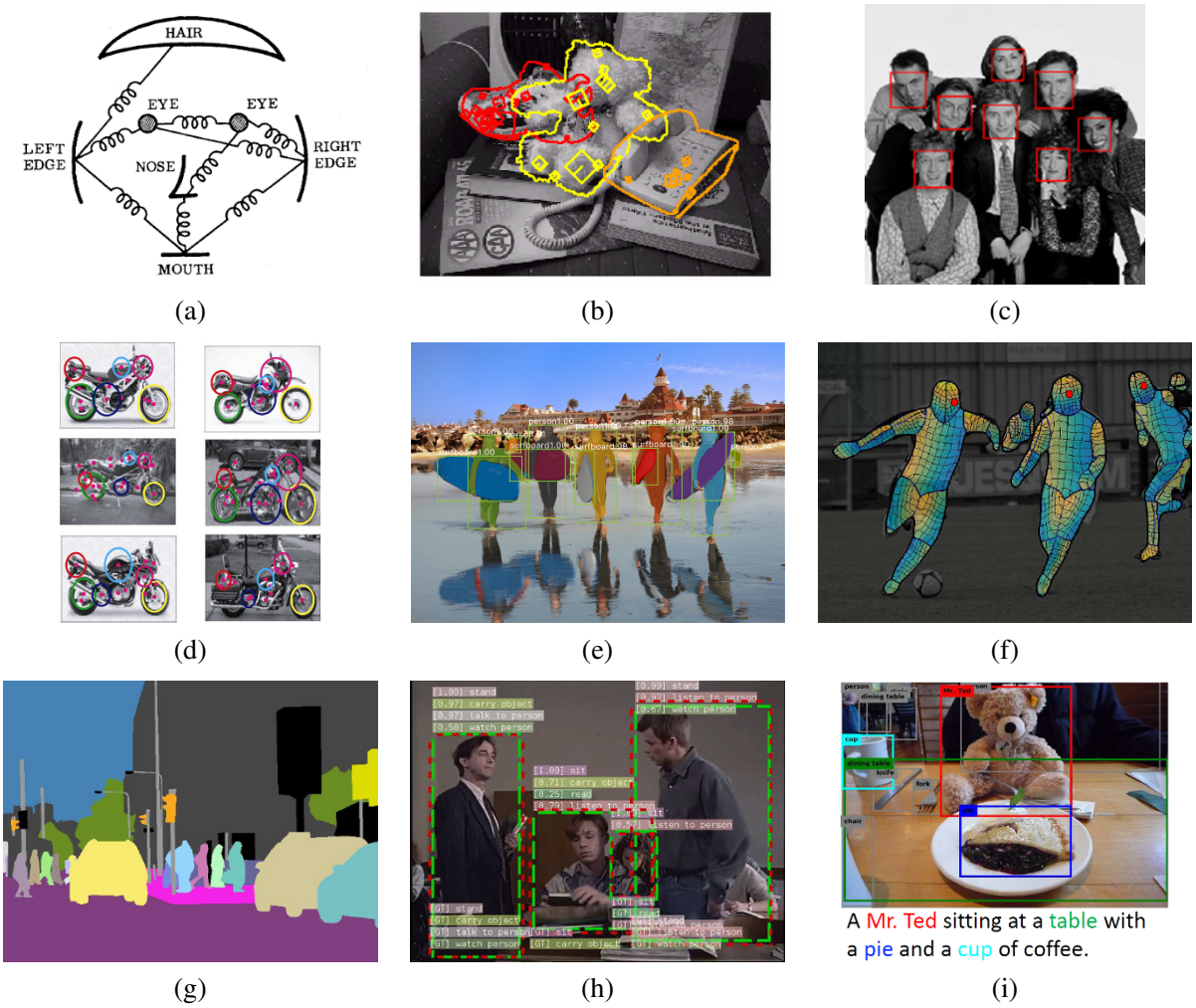


Figure 6.1 Various kinds of recognition: (a) face recognition with pictorial structures (Fischler and Elschlager 1973) © 1973 IEEE; (b) instance (known object) recognition (Lowe 1999) © 1999 IEEE; (c) real-time face detection (Viola and Jones 2004) © 2004 Springer; (d) feature-based recognition (Fergus, Perona, and Zisserman 2007) © 2007 Springer; (e) instance segmentation using Mask R-CNN (He, Gkioxari *et al.* 2017) © 2017 IEEE; (f) pose estimation (Güler, Neverova, and Kokkinos 2018) © 2018 IEEE; (g) panoptic segmentation (Kirillov, He *et al.* 2019) © 2019 IEEE; (h) video action recognition (Feichtenhofer, Fan *et al.* 2019); (i) image captioning (Lu, Yang *et al.* 2018) © 2018 IEEE.

Of all the computer vision topics covered in this book, visual recognition has undergone the largest changes and fastest development in the last decade, due in part to the availability of much larger labeled datasets as well as breakthroughs in deep learning (Figure 5.40). In the first edition of this book (Szeliski 2010), recognition was the last chapter, since it was considered a "high-level task" to be layered on top of lower-level components such as feature detection and matching. In fact, many introductory vision courses still teach recognition at the end, often covering "classic" (non-learning) vision algorithms and applications first, and then shifting to deep learning and recognition.

As I mentioned in the preface and introduction, I have now moved machine and deep learning to early in the book, since it is foundational technology widely used in other parts of computer vision. I also decided to move the recognition chapter right after deep learning, since most of the modern techniques for recognition are natural applications of deep neural networks. The majority of the old recognition chapter has been replaced with newer deep learning techniques, so you will sometimes find terse descriptions of classical recognition techniques along with pointers to the first edition and relevant surveys or seminal papers.

A good example of the classic approach is *instance recognition*, where we are trying to find exemplars of a particular manufactured object such as a stop sign or sneaker (Figure 6.1b). (An even earlier example is face recognition using relative feature locations, as shown in Figure 6.1a.) The general approach of finding distinctive features while dealing with local appearance variation (Section 7.1.2), and then checking for their co-occurrence and relative positions in an image, is still widely used for manufactured 3D object detection (Figure 6.3), 3D structure and pose recovery (Chapter 11), and location recognition (Section 11.2.3). Highly accurate and widely used feature-based approaches to instance recognition were developed in the 2000s (Figure 7.27) and, despite more recent deep learning-based alternatives, are often still the preferred method (Sattler, Zhou *et al.* 2019). We review instance recognition in Section 6.1, although some of the needed components, such as feature detection, description, and matching (Chapter 7), as well as 3D pose estimation and verification (Chapter 11), will not be introduced until later.

The more difficult problem of *category* or *class recognition* (e.g., recognizing members of highly variable categories such as cats, dogs, or motorcycles) was also initially attacked using feature-based approaches and relative locations (*part-based models*), such as the one depicted in Figure 6.1d. We begin our discussion of *image classification* (another name for whole-image category recognition) in Section 6.2 with a review of such "classic" (though now rarely used) techniques. We then show how the deep neural networks described in the previous chapter are ideally suited to these kinds of classification problems. Next, we cover visual similarity search, where instead of categorizing an image into a predefined number of categories, we retrieve other images that are semantically similar. Finally, we focus on face recognition, which is one of the longest studied topics in computer vision.

In Section 6.3, we turn to the topic of *object detection*, where we categorize not just whole images but delineate (with bounding boxes) where various objects are located. This topic includes more specialized variants such as *face detection* and *pedestrian detection*, as well as the detection of objects in generic categories. In Section 6.4, we study *semantic segmentation*, where the task is now to delineate various objects and materials in a pixel-accurate manner, i.e., to label each pixel with an object identity and class. Variants on this include *instance segmentation*, where each separate object gets a unique label, *panoptic segmentation*, where both objects and stuff (e.g., grass, sky) get labeled, and *pose estimation*, where pixels get labeled with people's body parts and orientations. The last two sections of this chapter briefly touch on *video understanding* (Section 6.5) and *vision and language* (Section 6.6).

Before starting to describe individual recognition algorithms and variants, I should briefly mention the critical role that large-scale datasets and benchmarks have played in the rapid advancement

Figure 6.2 Recognizing objects in a cluttered scene (Lowe 2004) © 2004 Springer. Two of the training images in the database are shown on the left. They are matched to the cluttered scene in the middle using SIFT features, shown as small squares in the right image. The affine warp of each recognized database image onto the scene is shown as a larger parallelogram in the right image.

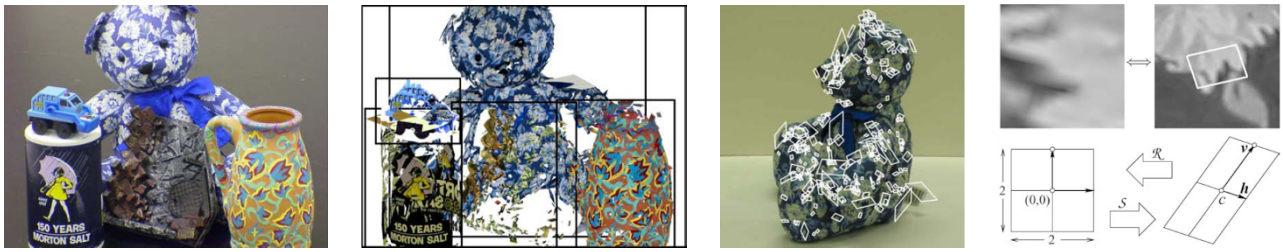
of recognition systems. While small datasets such as Xerox 10 (Csurka, Dance *et al.* 2006) and Caltech-101 (Fei-Fei, Fergus, and Perona 2006) played an early role in evaluating object recognition systems, the PASCAL Visual Object Class (VOC) challenge (Everingham, Van Gool *et al.* 2010; Everingham, Eslami *et al.* 2015) was the first dataset large and challenging enough to significantly propel the field forward. However, PASCAL VOC only contained 20 classes. The introduction of the ImageNet dataset (Deng, Dong *et al.* 2009; Russakovsky, Deng *et al.* 2015), which had 1,000 classes and over one million labeled images, finally provided enough data to enable end-to-end learning systems to break through. The Microsoft COCO (Common Objects in Context) dataset spurred further development (Lin, Maire *et al.* 2014), especially in accurate per-object segmentation, which we study in Section 6.4. A nice review of crowdsourcing methods to construct such datasets is presented in (Kovashka, Russakovsky *et al.* 2016). We will mention additional, sometimes more specialized, datasets throughout this chapter. A listing of the most popular and active datasets and benchmarks is provided in Tables 6.1–6.4.

6.1 Instance recognition

General object recognition falls into two broad categories, namely *instance recognition* and *class recognition*. The former involves re-recognizing a known 2D or 3D rigid object, potentially being viewed from a novel viewpoint, against a cluttered background, and with partial occlusions.[1] The latter, which is also known as *category-level* or *generic* object recognition (Ponce, Hebert *et al.* 2006), is the much more challenging problem of recognizing any instance of a particular general class, such as "cat", "car", or "bicycle".

Over the years, many different algorithms have been developed for instance recognition. Mundy (2006) surveys earlier approaches, which focused on extracting lines, contours, or 3D surfaces from images and matching them to known 3D object models. Another popular approach was to acquire images from a large set of viewpoints and illuminations and to represent them using an eigenspace

[1] The Microsoft COCO dataset paper (Lin, Maire *et al.* 2014) introduced the newer concept of *instance segmentation*, which is the pixel-accurate delineation of different objects drawn from a set of generic classes (Section 6.4.2). This now sometimes leads to confusion, unless you look at these two terms (instance recognition vs. segmentation) carefully.



(a) (b) (c) (d)

Figure 6.3 3D object recognition with affine regions (Rothganger, Lazebnik *et al.* 2006) © 2006 Springer: (a) sample input image; (b) five of the recognized (reprojected) objects along with their bounding boxes; (c) a few of the local affine regions; (d) local affine region (patch) reprojected into a canonical (square) frame, along with its geometric affine transformations.

decomposition (Murase and Nayar 1995). More recent approaches (Lowe 2004; Lepetit and Fua 2005; Rothganger, Lazebnik *et al.* 2006; Ferrari, Tuytelaars, and Van Gool 2006b; Gordon and Lowe 2006; Obdržálek and Matas 2006; Sivic and Zisserman 2009; Zheng, Yang, and Tian 2018) tend to use viewpoint-invariant 2D features, such as those we will discuss in Section 7.1.2. After extracting informative sparse 2D features from both the new image and the images in the database, image features are matched against the object database, using one of the sparse feature matching strategies described in Section 7.1.3. Whenever a sufficient number of matches have been found, they are verified by finding a geometric transformation that aligns the two sets of features (Figure 6.2).

Geometric alignment

To recognize one or more instances of some known objects, such as those shown in the left column of Figure 6.2, the recognition system first extracts a set of interest points in each database image and stores the associated descriptors (and original positions) in an indexing structure such as a search tree (Section 7.1.3). At recognition time, features are extracted from the new image and compared against the stored object features. Whenever a sufficient number of matching features (say, three or more) are found for a given object, the system then invokes a *match verification* stage, whose job is to determine whether the spatial arrangement of matching features is consistent with those in the database image.

Because images can be highly cluttered and similar features may belong to several objects, the original set of feature matches can have a large number of outliers. For this reason, Lowe (2004) suggests using a Hough transform (Section 7.4.2) to accumulate votes for likely geometric transformations. In his system, he uses an affine transformation between the database object and the collection of scene features, which works well for objects that are mostly planar, or where at least several corresponding features share a quasi-planar geometry.[2]

Another system that uses local affine frames is the one developed by Rothganger, Lazebnik *et al.* (2006). In their system, the affine region detector of Mikolajczyk and Schmid (2004) is used to rectify local image patches (Figure 6.3d), from which both a SIFT descriptor and a 10×10 UV color histogram are computed and used for matching and recognition. Corresponding patches in different views of the same object, along with their local affine deformations, are used to compute a 3D affine

[2] When a larger number of features is available, a full fundamental matrix can be used (Brown and Lowe 2002; Gordon and Lowe 2006). When image stitching is being performed (Brown and Lowe 2007), the motion models discussed in Section 8.2.1 can be used instead.

model for the object using an extension of the factorization algorithm of Section 11.4.1, which can then be upgraded to a Euclidean reconstruction (Tomasi and Kanade 1992). At recognition time, local Euclidean neighborhood constraints are used to filter potential matches, in a manner analogous to the affine geometric constraints used by Lowe (2004) and Obdržálek and Matas (2006). Figure 6.3 shows the results of recognizing five objects in a cluttered scene using this approach.

While feature-based approaches are normally used to detect and localize known objects in scenes, it is also possible to get pixel-level segmentations of the scene based on such matches. Ferrari, Tuytelaars, and Van Gool (2006b) describe such a system for simultaneously recognizing objects and segmenting scenes, while Kannala, Rahtu *et al.* (2008) extend this approach to non-rigid deformations. Section 6.4 re-visits this topic of joint recognition and segmentation in the context of generic class (category) recognition.

While instance recognition in the early to mid-2000s focused on the problem of locating a known 3D object in an image, as shown in Figures 6.2–6.3, attention shifted to the more challenging problem of *instance retrieval* (also known as *content-based image retrieval*), in which the number of images being searched can be very large. Section 7.1.4 reviews such techniques, a snapshot of which can be seen in Figure 7.27 and the survey by Zheng, Yang, and Tian (2018). This topic is also related to visual similarity search (Section 6.2.3 and 3D pose estimation (Section 11.2).

6.2 Image classification

While instance recognition techniques are relatively mature and are used in commercial applications such as traffic sign recognition (Stallkamp, Schlipsing *et al.* 2012), generic category (class) recognition is still a rapidly evolving research area. Consider for example the set of photographs in Figure 6.4a, which shows objects taken from 10 different visual categories. (I'll leave it up to you to name each of the categories.) How would you go about writing a program to categorize each of these images into the appropriate class, especially if you were also given the choice "none of the above"?

As you can tell from this example, visual category recognition is an extremely challenging problem. However, the progress in the field has been quite dramatic, if judged by how much better today's algorithms are compared to those of a decade ago.

In this section, we review the main classes of algorithms used for whole-image classification. We begin with classic feature-based approaches that rely on handcrafted features and their statistics, optionally using machine learning to do the final classification (Figure 5.2b). Since such techniques are no longer widely used, we present a fairly terse description of the most important techniques. More details can be found in the first edition of this book (Szeliski 2010, Chapter 14) and in the cited journal papers and surveys. Next, we describe modern image classification systems, which are based on the deep neural networks we introduced in the previous chapter. We then describe visual similarity search, where the task is to find visually and semantically similar images, rather than classification into a fixed set of categories. Finally, we look at face recognition, since this topic has its own long history and set of techniques.

6.2.1 Feature-based methods

In this section, we review "classic" feature-based approaches to category recognition (image classification). While, historically, *part-based* representations and recognition algorithms (Section 6.2.1) were the preferred approach (Fischler and Elschlager 1973; Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2007), we begin by describing simpler *bag-of-features* approaches

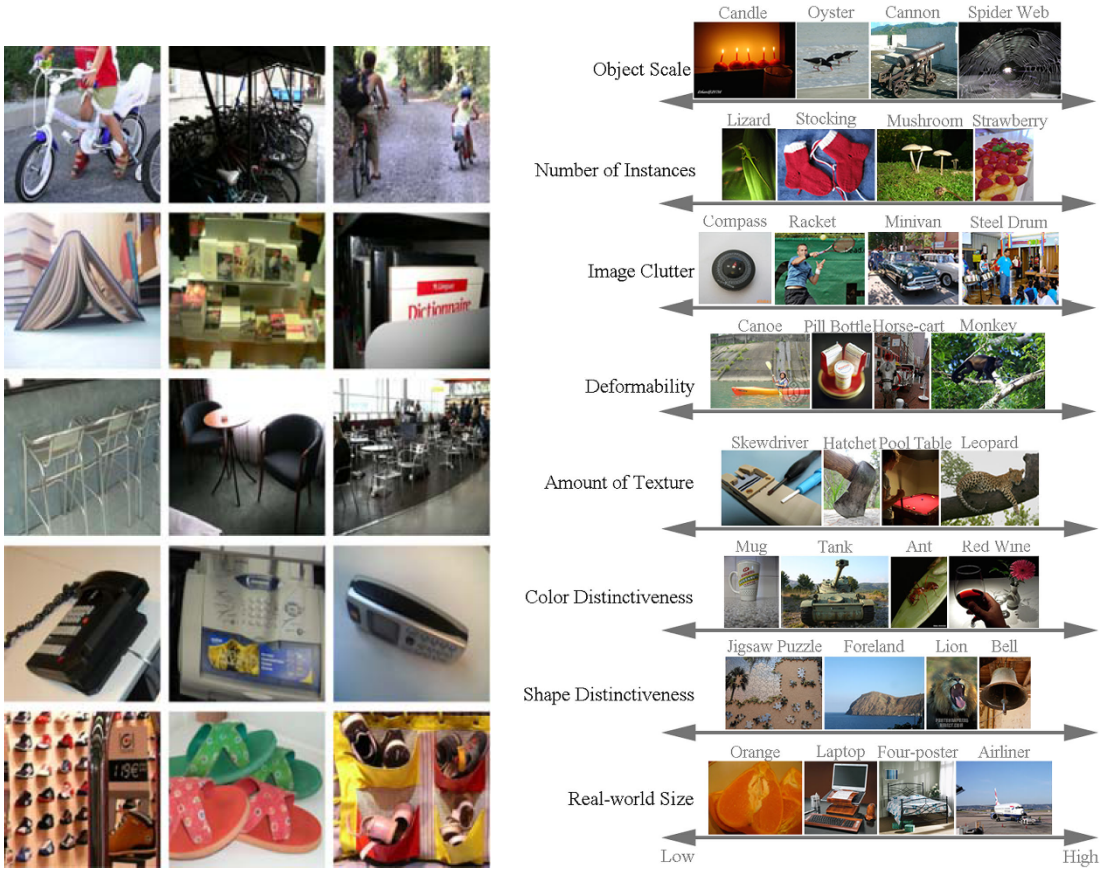
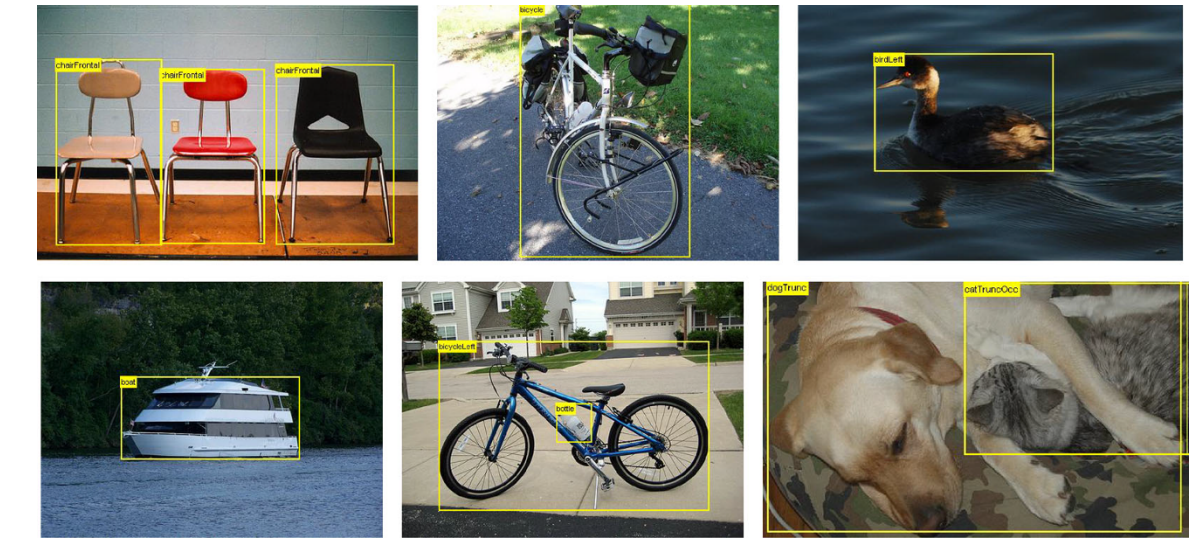


Figure 6.4 Challenges in image recognition: (a) sample images from the Xerox 10 class dataset (Csurka, Dance *et al.* 2006) © 2007 Springer; (b) axes of difficulty and variation from the ImageNet dataset (Russakovsky, Deng *et al.* 2015) © 2015 Springer.

that represent objects and images as unordered collections of feature descriptors. We then review more complex systems constructed with part-based models, and then look at how context and scene understanding, as well as machine learning, can improve overall recognition results. Additional details on the techniques presented in this section can be found in older survey articles, paper collections, and courses (Pinz 2005; Ponce, Hebert *et al.* 2006; Dickinson, Leonardis *et al.* 2007; Fei-Fei, Fergus, and Torralba 2009), as well as two review articles on the PASCAL and ImageNet recognition challenges (Everingham, Van Gool *et al.* 2010; Everingham, Eslami *et al.* 2015; Russakovsky, Deng *et al.* 2015) and the first edition of this book (Szeliski 2010, Chapter 14).

Bag of words

One of the simplest algorithms for category recognition is the *bag of words* (also known as *bag of features* or *bag of keypoints*) approach (Csurka, Dance *et al.* 2004; Lazebnik, Schmid, and Ponce 2006; Csurka, Dance *et al.* 2006; Zhang, Marszalek *et al.* 2007). As shown in Figure 6.6, this algorithm simply computes the distribution (histogram) of visual words found in the query image and compares this distribution to those found in the training images. We will give more details of this approach in Section 7.1.4. The biggest difference from instance recognition is the absence of



(a)

(b)

Figure 6.5 Sample images from two widely used image classification datasets: (a) Pascal Visual Object Categories (VOC) (Everingham, Eslami *et al.* 2015) © 2015 Springer; (b) ImageNet (Russakovsky, Deng *et al.* 2015) © 2015 Springer.

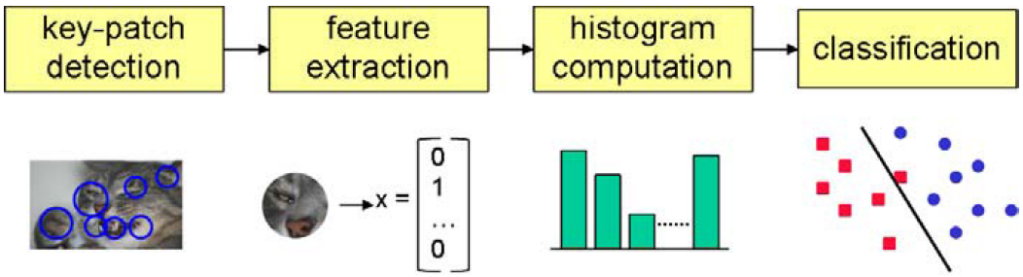


Figure 6.6 A typical processing pipeline for a bag-of-words category recognition system (Csurka, Dance *et al.* 2006) © 2007 Springer. Features are first extracted at keypoints and then quantized to get a distribution (histogram) over the learned *visual words* (feature cluster centers). The feature distribution histogram is used to learn a decision surface using a classification algorithm, such as a support vector machine.

a geometric verification stage (Section 6.1), since individual instances of generic visual categories, such as those shown in Figure 6.4a, have relatively little spatial coherence to their features (but see the work by Lazebnik, Schmid, and Ponce (2006)).

Csurka, Dance *et al.* (2004) were the first to use the term *bag of keypoints* to describe such approaches and among the first to demonstrate the utility of frequency-based techniques for category recognition. Their original system used affine covariant regions and SIFT descriptors, k-means visual vocabulary construction, and both a naïve Bayesian classifier and support vector machines for classification. (The latter was found to perform better.) Their newer system (Csurka, Dance *et al.* 2006) uses regular (non-affine) SIFT patches and boosting instead of SVMs and incorporates a small amount of geometric consistency information.

Zhang, Marszalek *et al.* (2007) perform a more detailed study of such bag of features systems. They compare a number of feature detectors (Harris–Laplace (Mikolajczyk and Schmid 2004) and Laplacian (Lindeberg 1998b)), descriptors (SIFT, RIFT, and SPIN (Lazebnik, Schmid, and Ponce 2005)), and SVM kernel functions.

Instead of quantizing feature vectors to visual words, Grauman and Darrell (2007b) develop a technique for directly computing an approximate distance between two variably sized collections of feature vectors. Their approach is to bin the feature vectors into a multi-resolution pyramid defined in feature space and count the number of features that land in corresponding bins B_{il} and B'_{il}. The distance between the two sets of feature vectors (which can be thought of as points in a high-dimensional space) is computed using histogram intersection between corresponding bins, while discounting matches already found at finer levels and weighting finer matches more heavily. In follow-on work, Grauman and Darrell (2007a) show how an explicit construction of the pyramid can be avoided using hashing techniques.

Inspired by this work, Lazebnik, Schmid, and Ponce (2006) show how a similar idea can be employed to augment bags of keypoints with loose notions of 2D spatial location analogous to the pooling performed by SIFT (Lowe 2004) and "gist" (Torralba, Murphy *et al.* 2003). In their work, they extract affine region descriptors (Lazebnik, Schmid, and Ponce 2005) and quantize them into visual words. (Based on previous results by Fei-Fei and Perona (2005), the feature descriptors are extracted densely (on a regular grid) over the image, which can be helpful in describing texture-less regions such as the sky.) They then form a spatial pyramid of bins containing word counts (histograms) and use a similar pyramid match kernel to combine histogram intersection counts in a hierarchical fashion.

The debate about whether to use quantized feature descriptors or continuous descriptors and

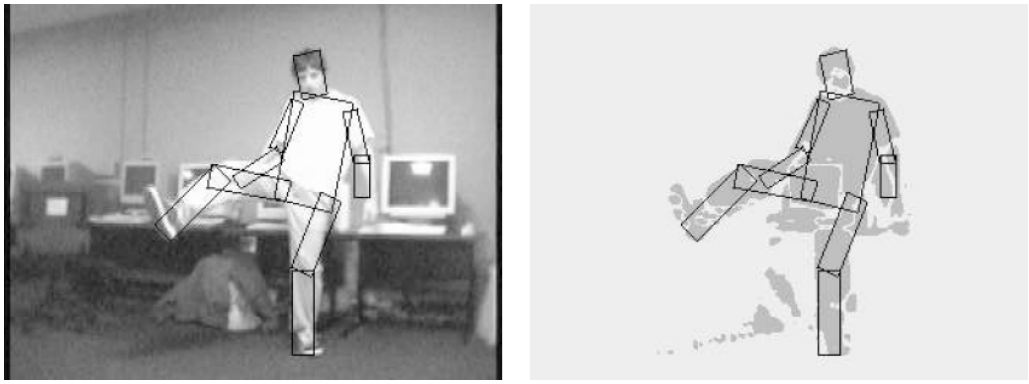


Figure 6.7 Using pictorial structures to locate and track a person (Felzenszwalb and Huttenlocher 2005) © 2005 Springer. The structure consists of articulated rectangular body parts (torso, head, and limbs) connected in a tree topology that encodes relative part positions and orientations. To fit a pictorial structure model, a binary silhouette image is first computed using background subtraction.

also whether to use sparse or dense features went on for many years. Boiman, Shechtman, and Irani (2008) show that if query images are compared to *all* the features representing a given class, rather than just each class image individually, nearest-neighbor matching followed by a naïve Bayes classifier outperforms quantized visual words. Instead of using generic feature detectors and descriptors, some authors have been investigating *learning* class-specific features (Ferencz, Learned-Miller, and Malik 2008), often using randomized forests (Philbin, Chum *et al.* 2007; Moosmann, Nowak, and Jurie 2008; Shotton, Johnson, and Cipolla 2008) or combining the feature generation and image classification stages (Yang, Jin *et al.* 2008). Others, such as Serre, Wolf, and Poggio (2005) and Mutch and Lowe (2008) use hierarchies of dense feature transforms inspired by biological (visual cortical) processing combined with SVMs for final classification.

Part-based models

Recognizing an object by finding its constituent parts and measuring their geometric relationships is one of the oldest approaches to object recognition (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991). Part-based approaches were often used for face recognition (Moghaddam and Pentland 1997; Heisele, Ho *et al.* 2003; Heisele, Serre, and Poggio 2007) and continue being used for pedestrian detection (Figure 6.24) (Felzenszwalb, McAllester, and Ramanan 2008) and pose estimation (Güler, Neverova, and Kokkinos 2018).

In this overview, we discuss some of the central issues in part-based recognition, namely, the representation of geometric relationships, the representation of individual parts, and algorithms for learning such descriptions and recognizing them at run time. More details on part-based models for recognition can be found in the course notes by Fergus (2009).

The earliest approaches to representing geometric relationships were dubbed *pictorial structures* by Fischler and Elschlager (1973) and consisted of spring-like connections between different feature locations (Figure 6.1a). To fit a pictorial structure to an image, an energy function of the form

$$E = \sum_i V_i(\mathbf{l}_i) + \sum_{ij \in E} V_{ij}(\mathbf{l}_i, \mathbf{l}_j) \tag{6.1}$$

is minimized over all potential part locations or poses $\{\mathbf{l}_i\}$ and pairs of parts (i, j) for which an edge (geometric relationship) exists in E. Note how this energy is closely related to that used with

Markov random fields (4.35–4.38), which can be used to embed pictorial structures in a probabilistic framework that makes parameter learning easier (Felzenszwalb and Huttenlocher 2005).

Part-based models can have different topologies for the geometric connections between the parts (Carneiro and Lowe 2006). For example, Felzenszwalb and Huttenlocher (2005) restrict the connections to a tree, which makes learning and inference more tractable. A tree topology enables the use of a recursive Viterbi (dynamic programming) algorithm (Pearl 1988; Bishop 2006), in which leaf nodes are first optimized as a function of their parents, and the resulting values are then plugged in and eliminated from the energy function, To further increase the efficiency of the inference algorithm, Felzenszwalb and Huttenlocher (2005) restrict the pairwise energy functions $V_{ij}(\mathbf{l}_i, \mathbf{l}_j)$ to be Mahalanobis distances on functions of location variables and then use fast distance transform algorithms to minimize each pairwise interaction in time that is closer to linear in N.

Figure 6.7 shows the results of using their pictorial structures algorithm to fit an articulated body model to a binary image obtained by background segmentation. In this application of pictorial structures, parts are parameterized by the locations, sizes, and orientations of their approximating rectangles. Unary matching potentials $V_i(\mathbf{l}_i)$ are determined by counting the percentage of foreground and background pixels inside and just outside the tilted rectangle representing each part.

A large number of different graphical models have been proposed for part-based recognition. Carneiro and Lowe (2006) discuss a number of these models and propose one of their own, which they call a *sparse flexible model*; it involves ordering the parts and having each part's location depend on at most k of its ancestor locations.

The simplest models are bags of words, where there are no geometric relationships between different parts or features. While such models can be very efficient, they have a very limited capacity to express the spatial arrangement of parts. Trees and stars (a special case of trees where all leaf nodes are directly connected to a common root) are the most efficient in terms of inference and hence also learning (Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2005; Felzenszwalb, McAllester, and Ramanan 2008). Directed acyclic graphs come next in terms of complexity and can still support efficient inference, although at the cost of imposing a causal structure on the part model (Bouchard and Triggs 2005; Carneiro and Lowe 2006). k-fans, in which a clique of size k forms the root of a star-shaped model have inference complexity $O(N^{k+1})$, although with distance transforms and Gaussian priors, this can be lowered to $O(N^k)$ (Crandall, Felzenszwalb, and Huttenlocher 2005; Crandall and Huttenlocher 2006). Finally, fully connected *constellation* models are the most general, but the assignment of features to parts becomes intractable for moderate numbers of parts P, since the complexity of such an assignment is $O(N^P)$ (Fergus, Perona, and Zisserman 2007).

The original constellation model was developed by Burl, Weber, and Perona (1998) and consists of a number of parts whose relative positions are encoded by their mean locations and a full covariance matrix, which is used to denote not only positional uncertainty but also potential correlations between different parts. Weber, Welling, and Perona (2000) extended this technique to a weakly supervised setting, where both the appearance of each part and its locations are automatically learned given whole image labels. Fergus, Perona, and Zisserman (2007) further extend this approach to simultaneous learning of appearance and shape models from scale-invariant keypoint detections.

The part-based approach to recognition has also been extended to learning new categories from small numbers of examples, building on recognition components developed for other classes (Fei-Fei, Fergus, and Perona 2006). More complex hierarchical part-based models can be developed using the concept of grammars (Bouchard and Triggs 2005; Zhu and Mumford 2006). A simpler way to use parts is to have keypoints that are recognized as being part of a class vote for the estimated part locations (Leibe, Leonardis, and Schiele 2008). Parts can also be a useful component of fine-grained category recognition systems, as shown in Figure 6.9.

(a) (b) (c) (d) (e)

Figure 6.8 The importance of context (images courtesy of Antonio Torralba). Can you name all of the objects in images (a–b), especially those that are circled in (c–d). Look carefully at the circled objects. Did you notice that they all have the same shape (after being rotated), as shown in column (e)?

Context and scene understanding

Thus far, we have mostly considered the task of recognizing and localizing objects in isolation from that of understanding the scene (context) in which the object occur. This is a big limitation, as context plays a very important role in human object recognition (Oliva and Torralba 2007). Context can greatly improve the performance of object recognition algorithms (Divvala, Hoiem *et al.* 2009), as well as providing useful semantic clues for general scene understanding (Torralba 2008).

Consider the two photographs in Figure 6.8a–b. Can you name all of the objects, especially those circled in images (c–d)? Now have a closer look at the circled objects. Do see any similarity in their shapes? In fact, if you rotate them by 90°, they are all the same as the "blob" shown in Figure 6.8e. So much for our ability to recognize object by their shape!

Even though we have not addressed context explicitly earlier in this chapter, we have already seen several instances of this general idea being used. A simple way to incorporate spatial information into a recognition algorithm is to compute feature statistics over different regions, as in the spatial pyramid system of Lazebnik, Schmid, and Ponce (2006). Part-based models (Figure 6.7) use a kind of local context, where various parts need to be arranged in a proper geometric relationship to constitute an object.

The biggest difference between part-based and context models is that the latter combine objects into scenes and the number of constituent objects from each class is not known in advance. In fact, it is possible to combine part-based and context models into the same recognition architecture (Murphy, Torralba, and Freeman 2003; Sudderth, Torralba *et al.* 2008; Crandall and Huttenlocher 2007).

Consider an image database consisting of street and office scenes. If we have enough training images with labeled regions, such as buildings, cars, and roads, or monitors, keyboards, and mice, we can develop a geometric model for describing their relative positions. Sudderth, Torralba *et al.* (2008) develop such a model, which can be thought of as a two-level constellation model. At the top level, the distributions of objects relative to each other (say, buildings with respect to cars) is modeled as a Gaussian. At the bottom level, the distribution of parts (affine covariant features) with respect to the object center is modeled using a mixture of Gaussians. However, since the number of objects in the scene and parts in each object are unknown, a *latent Dirichlet process* (LDP) is used to model object and part creation in a generative framework. The distributions for all of the objects and parts are learned from a large labeled database and then later used during inference (recognition) to label the elements of a scene.

Another example of context is in simultaneous segmentation and recognition (Section 6.4 and Figure 6.33), where the arrangements of various objects in a scene are used as part of the labeling process. Torralba, Murphy, and Freeman (2004) describe a conditional random field where the estimated locations of building and roads influence the detection of cars, and where boosting is used to learn the structure of the CRF. Rabinovich, Vedaldi *et al.* (2007) use context to improve the results of CRF segmentation by noting that certain adjacencies (relationships) are more likely than others, e.g., a person is more likely to be on a horse than on a dog. Galleguillos and Belongie (2010) review various approaches proposed for adding context to object categorization, while Yao and Fei-Fei (2012) study human-object interactions. (For a more recent take on this problem, see Gkioxari, Girshick *et al.* (2018).)

Context also plays an important role in 3D inference from single images (Figure 6.41), using computer vision techniques for labeling pixels as belonging to the ground, vertical surfaces, or sky (Hoiem, Efros, and Hebert 2005a). This line of work has been extended to a more holistic approach that simultaneously reasons about object identity, location, surface orientations, occlusions, and camera viewing parameters (Hoiem, Efros, and Hebert 2008).

A number of approaches use the *gist* of a scene (Torralba 2003; Torralba, Murphy *et al.* 2003) to determine where instances of particular objects are likely to occur. For example, Murphy, Torralba, and Freeman (2003) train a regressor to predict the vertical locations of objects such as pedestrians, cars, and buildings (or screens and keyboards for indoor office scenes) based on the gist of an image. These location distributions are then used with classic object detectors to improve the performance of the detectors. Gists can also be used to directly match complete images, as we saw in the scene completion work of Hays and Efros (2007).

Finally, some of the work in scene understanding exploits the existence of large numbers of labeled (or even unlabeled) images to perform matching directly against whole images, where the images themselves implicitly encode the expected relationships between objects (Russell, Torralba *et al.* 2007; Malisiewicz and Efros 2008; Galleguillos and Belongie 2010). This, of course, is one of the central benefits of using deep neural networks, which we discuss in the next section.

6.2.2 Deep networks

As we saw in Section 5.4.3, deep networks started outperforming "shallow" learning-based approaches on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with the introduction of the "AlexNet" SuperVision system of Krizhevsky, Sutskever, and Hinton (2012). Since that time, recognition accuracy has continued to improve dramatically (Figure 5.40) driven to a large degree by deeper networks and better training algorithms. More recently, more efficient networks have become the focus of research (Figure 5.45) as well as larger (unlabeled) training datasets (Section 5.4.7). There are now open-source frameworks such as Classy Vision[3] for training and fine tuning your own image and video classification models. Users can also upload custom images on the web to the Computer Vision Explorer[4] to see how well many popular computer vision models perform on their own images.

In addition to recognizing commonly occurring categories such as those found in the ImageNet and COCO datasets, researchers have studied the problem of *fine-grained* category recognition (Duan, Parikh *et al.* 2012; Zhang, Donahue *et al.* 2014; Krause, Jin *et al.* 2015), where the differences between sub-categories can be subtle and the number of exemplars is quite low (Figure 6.9). Examples of categories with fine-grained sub-classes include flowers (Nilsback and Zisserman 2006), cats and dogs (Parkhi, Vedaldi *et al.* 2012), birds (Wah, Branson *et al.* 2011; Van Horn,

[3] https://classyvision.ai
[4] https://vision-explorer.allenai.org

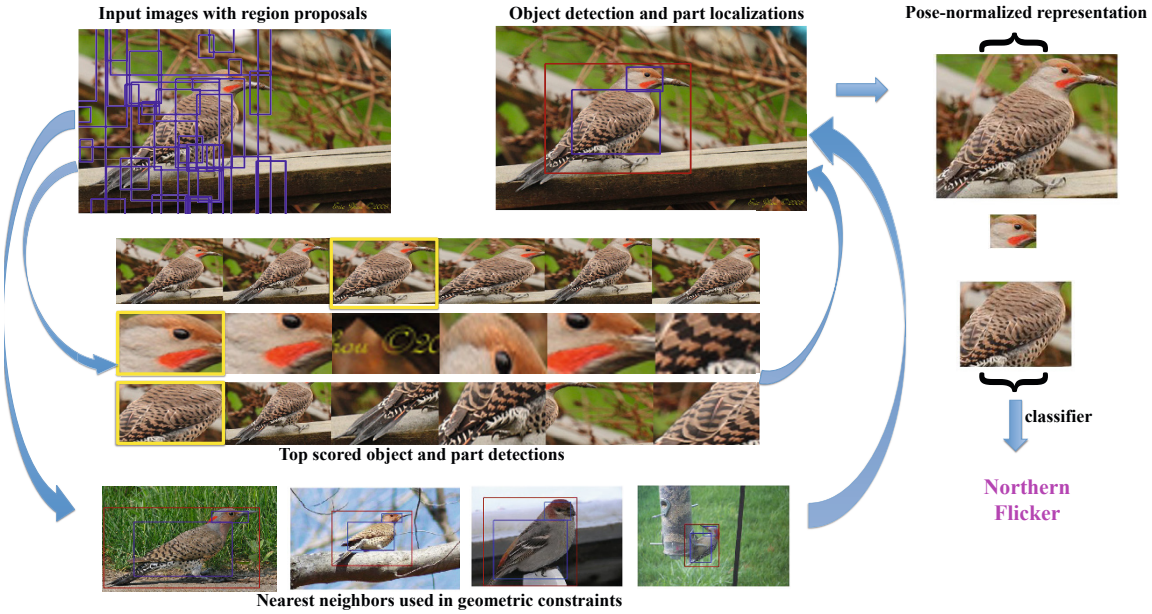


Figure 6.9 Fine-grained category recognition using parts (Zhang, Donahue *et al.* 2014) © 2014 Springer. Deep neural network object and part detectors are trained and their outputs are combined using geometric constraints. A classifier trained on features from the extracted parts is used for the final categorization.

(a) (b)

Figure 6.10 Fine-grained category recognition. (a) The iNaturalist website and app allows citizen scientists to collect and classify images on their phones (Van Horn, Mac Aodha *et al.* 2018) © 2018 IEEE. (b) Attributes can be used for fine-grained categorization and zero-shot learning (Lampert, Nickisch, and Harmeling 2014) © 2014 Springer. These images are part of the *Animals with Attributes* dataset.

Branson *et al.* 2015), and cars (Yang, Luo *et al.* 2015). A recent example of fine-grained categorization is the iNaturalist system (Van Horn, Mac Aodha *et al.* 2018),[5] which allows both specialists and citizen scientists to photograph and label biological species, using a fine-grained category recognition system to label new images (Figure 6.10a).

Fine-grained categorization is often attacked using *attributes* of images and classes (Lampert, Nickisch, and Harmeling 2009; Parikh and Grauman 2011; Lampert, Nickisch, and Harmeling 2014), as shown in Figure 6.10b. Extracting attributes can enable *zero-shot learning* (Xian, Lampert *et al.* 2019), where previously unseen categories can be described using combinations of such attributes. However, some caution must be used in order not to learn spurious correlations between different attributes (Jayaraman, Sha, and Grauman 2014) or between objects and their common contexts (Singh, Mahajan *et al.* 2020). Fine-grained recognition can also be tackled using metric learning (Wu, Manmatha *et al.* 2017) or nearest-neighbor visual similarity search (Touvron, Sablayrolles *et al.* 2020), which we discuss next.

6.2.3 *Application*: Visual similarity search

Automatically classifying images into categories and tagging them with attributes using computer vision algorithms makes it easier to find them in catalogues and on the web. This is commonly used in *image search* or *image retrieval* engines, which find likely images based on keywords, just as regular web search engines find relevant documents and pages.

Sometimes, however, it's easier to find the information you need from an image, i.e., using *visual search*. Examples of this include fine-grained categorization, which we have just seen, as well as instance retrieval, i.e., finding the exact same object (Section 6.1) or location (Section 11.2.3). Another variant is finding visually similar images (often called *visual similarity search* or *reverse image search*), which is useful when the search intent cannot be succinctly captured in words.[6]

The topic of searching by visual similarity has a long history and goes by a variety of names, including query by image content (QBIC) (Flickner, Sawhney *et al.* 1995) and content-based image retrieval (CBIR) (Smeulders, Worring *et al.* 2000; Lew, Sebe *et al.* 2006; Vasconcelos 2007; Datta, Joshi *et al.* 2008). Early publications in these fields were based primarily on simple whole-image similarity metrics, such as color and texture (Swain and Ballard 1991; Jacobs, Finkelstein, and Salesin 1995; Manjunathi and Ma 1996).

Later architectures, such as that by Fergus, Perona, and Zisserman (2004), use a feature-based learning and recognition algorithm to re-rank the outputs from a traditional keyword-based image search engine. In follow-on work, Fergus, Fei-Fei *et al.* (2005) cluster the results returned by image search using an extension of probabilistic latest semantic analysis (PLSA) (Hofmann 1999) and then select the clusters associated with the highest ranked results as the representative images for that category. Other approaches rely on carefully annotated image databases such as LabelMe (Russell, Torralba *et al.* 2008). For example, Malisiewicz and Efros (2008) describe a system that, given a query image, can find similar LabelMe images, whereas Liu, Yuen, and Torralba (2009) combine feature-based correspondence algorithms with the labeled database to perform simultaneous recognition and segmentation.

Newer approaches to visual similarity search use whole-image descriptors such as Fisher kernels and the Vector of Locally Aggregated Descriptors (VLAD) (Jégou, Perronnin *et al.* 2012) or pooled CNN activations (Babenko and Lempitsky 2015a; Tolias, Sicre, and Jégou 2016; Cao, Araujo, and Sim 2020; Ng, Balntas *et al.* 2020; Tolias, Jenicek, and Chum 2020) combined with metric learning

[5] https://www.inaturalist.org

[6] Some authors use the term image retrieval to denote visual similarity search, (e.g., Jégou, Perronnin *et al.* 2012; Radenović, Tolias, and Chum 2019).

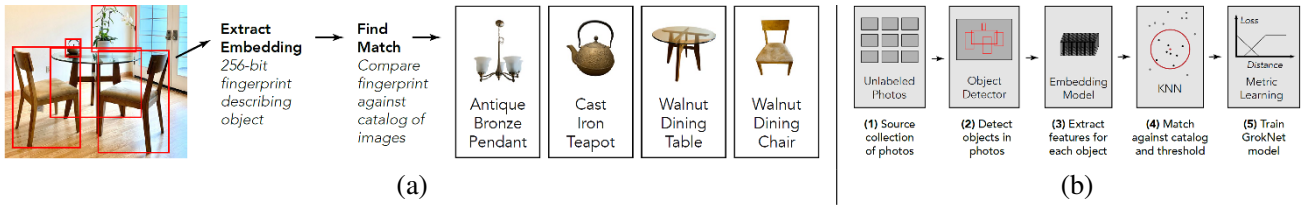


Figure 6.11 The GrokNet product recognition service is used for product tagging, visual search, and recommendations © Bell, Liu *et al.* (2020): (a) recognizing all the products in a photo; (b) automatically sourcing data for metric learning using weakly supervised data augmentation.

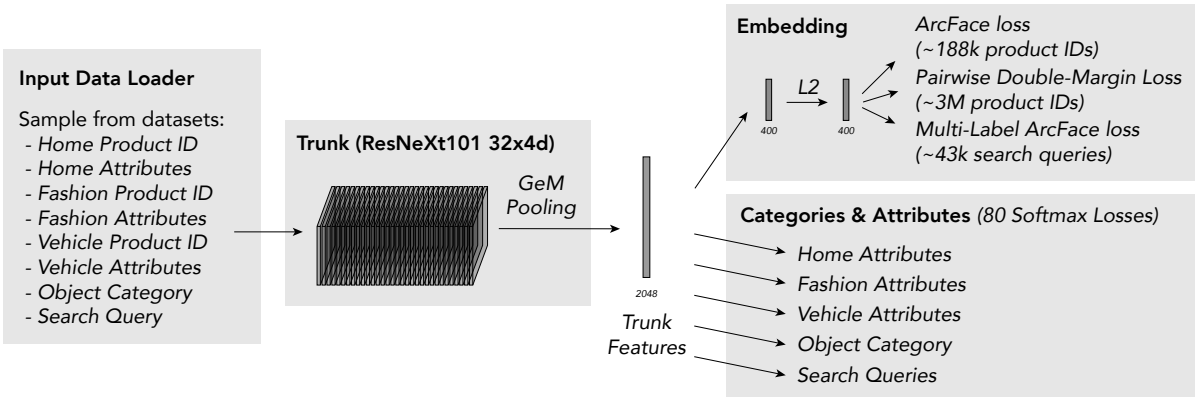


Figure 6.12 The GrokNet training architecture uses seven datasets, a common DNN trunk, two branches, and 83 loss functions (80 categorical losses + 3 embedding losses) © Bell, Liu *et al.* (2020).

(Bell and Bala 2015; Song, Xiang *et al.* 2016; Gordo, Almazán *et al.* 2017; Wu, Manmatha *et al.* 2017; Berman, Jégou *et al.* 2019) to represent each image with a compact descriptor that can be used to measure similarity in large databases (Johnson, Douze, and Jégou 2021). It is also possible to combine several techniques, such as deep networks with VLAD (Arandjelovic, Gronat *et al.* 2016), generalized mean (GeM) pooling (Radenović, Tolias, and Chum 2019), or dynamic mean (DAME) pooling (Yang, Kien Nguyen *et al.* 2019) into complete systems that are end-to-end tunable. Gordo, Almazán *et al.* (2017) provide a comprehensive review and experimental comparison of many of these techniques, which we also discuss in Section 7.1.4 on large-scale matching and retrieval. Some of the latest techniques for image retrieval use combinations of local and global descriptors to obtain state-of-the art performance on the landmark recognition tasks (Cao, Araujo, and Sim 2020; Ng, Balntas *et al.* 2020; Tolias, Jenicek, and Chum 2020). The ECCV 2020 Workshop on Instance-Level Recognition[7] has pointers to some of the latest work in this area, while the upcoming NeurIPS'21 Image Similarity Challenge[8] has new datasets for detecting content manipulation.

A recent example of a commercial system that uses visual similarity search, in addition to category recognition, is the GrokNet product recognition service described by Bell, Liu *et al.* (2020). GrokNet takes as input user images and shopping queries and returns indexed items similar to the ones in the query image (Figure 6.11a). The reason for needing a similarity search component is that the world contains too many "long-tail" items such as "a fur sink, an electric dog polisher, or a

[7] https://ilr-workshop.github.io/ECCVW2020
[8] https://www.drivendata.org/competitions/79/

Figure 6.13 Humans can recognize low-resolution faces of familiar people (Sinha, Balas *et al.* 2006) © 2006 IEEE.

gasoline powered turtleneck sweater",[9] to make full categorization practical.

At training time, GrokNet takes both weakly labeled images, with category and/or attribute labels, and unlabeled images, where features in objects are detected and then used for metric learning, using a modification of ArcFace loss (Deng, Guo *et al.* 2019) and a novel pairwise margin loss (Figure 6.11b). The overall system takes in large collections of unlabeled and weakly labeled images and trains a ResNeXt101 trunk using a combination of category and attribute softmax losses and three different metric losses on the embeddings (Figure 6.12). GrokNet is just one example of a large number of commercial visual product search systems that have recently been developed. Others include systems from Amazon (Wu, Manmatha *et al.* 2017), Pinterest (Zhai, Wu *et al.* 2019), and Facebook (Tang, Borisyuk *et al.* 2019). In addition to helping people find items they may with to purchase, large-scale similarity search can also speed the search for harmful content on the web, as exemplified in Facebook's SimSearchNet.[10]

6.2.4 Face recognition

Among the various recognition tasks that computers are asked to perform, face recognition is the one where they have arguably had the most success.[11] While even people cannot readily distinguish between similar people with whom they are not familiar (O'Toole, Jiang *et al.* 2006; O'Toole, Phillips *et al.* 2009), computers' ability to distinguish among a small number of family members and friends has found its way into consumer-level photo applications. Face recognition can be used in a variety of additional applications, including human–computer interaction (HCI), identity verification (Kirovski, Jojic, and Jancke 2004), desktop login, parental controls, and patient monitoring (Zhao, Chellappa *et al.* 2003), but it also has the potential for misuse (Chokshi 2019; Ovide 2020).

Face recognizers work best when they are given images of faces under a wide variety of pose, illumination, and expression (PIE) conditions (Phillips, Moon *et al.* 2000; Sim, Baker, and Bsat 2003; Gross, Shi, and Cohn 2005; Huang, Ramesh *et al.* 2007; Phillips, Scruggs *et al.* 2010). More recent widely used datasets include labeled Faces in the Wild (LFW) (Huang, Ramesh *et al.* 2007; Learned-Miller, Huang *et al.* 2016), YouTube Faces (YTF) (Wolf, Hassner, and Maoz 2011), MegaFace (Kemelmacher-Shlizerman, Seitz *et al.* 2016; Nech and Kemelmacher-Shlizerman 2017), and the IARPA Janus Benchmark (IJB) (Klare, Klein *et al.* 2015; Maze, Adams *et al.* 2018), as tabulated in

[9] https://www.google.com/search?q=gasoline+powered+turtleneck+sweater

[10] https://ai.facebook.com/blog/using-ai-to-detect-covid-19-misinformation-and-exploitative-content

[11] Instance recognition, i.e., the re-recognition of known objects such as locations or planar objects, is the other most successful application of general image recognition. In the general domain of *biometrics*, i.e., identity recognition, specialized images such as irises and fingerprints perform even better (Jain, Bolle, and Pankanti 1999; Daugman 2004).

Name/URL	Contents/Reference
CMU Multi-PIE database http://www.cs.cmu.edu/afs/cs/project/PIE/MultiPie	337 people's faces in various poses Gross, Matthews *et al.* (2010)
Faces in the Wild http://vis-www.cs.umass.edu/lfw	5,749 internet celebrities Huang, Ramesh *et al.* (2007)
YouTube Faces (YTF) https://www.cs.tau.ac.il/~wolf/ytfaces	1,595 people in 3,425 YouTube videos Wolf, Hassner, and Maoz (2011)
MegaFace https://megaface.cs.washington.edu	1M internet faces Nech and Kemelmacher-Shlizerman (2017)
IARPA Janus Benchmark (IJB) https://www.nist.gov/programs-projects/face-challenges	31,334 faces of 3,531 people in videos Maze, Adams *et al.* (2018)
WIDER FACE http://shuoyang1213.me/WIDERFACE	32,203 images for face *detection* Yang, Luo *et al.* (2016)

Table 6.1 Face recognition and detection datasets, adapted from Maze, Adams *et al.* (2018).

Table 6.1. (See Masi, Wu *et al.* (2018) for additional datasets used for training.)

Some of the earliest approaches to face recognition involved finding the locations of distinctive image features, such as the eyes, nose, and mouth, and measuring the distances between these feature locations (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991). Other approaches relied on comparing gray-level images projected onto lower dimensional subspaces called *eigenfaces* (Section 5.2.3) and jointly modeling shape and appearance variations (while discounting pose variations) using *active appearance models* (Section 6.2.4). Descriptions of "classic" (pre-DNN) face recognition systems can be found in a number of surveys and books on this topic (Chellappa, Wilson, and Sirohey 1995; Zhao, Chellappa *et al.* 2003; Li and Jain 2005) as well as the Face Recognition website.[12] The survey on face recognition by humans by Sinha, Balas *et al.* (2006) is also well worth reading; it includes a number of surprising results, such as humans' ability to recognize low-resolution images of familiar faces (Figure 6.13) and the importance of eyebrows in recognition. Researchers have also studied the automatic recognition of facial expressions. See Chang, Hu *et al.* (2006), Shan, Gong, and McOwan (2009), and Li and Deng (2020) for some representative papers.

Active appearance and 3D shape models

The need to use modular or view-based eigenspaces for face recognition, which we discussed in Section 5.2.3, is symptomatic of a more general observation, i.e., that facial appearance and identifiability depend as much on *shape* as they do on color or texture (which is what eigenfaces capture). Furthermore, when dealing with 3D head rotations, the *pose* of a person's head should be discounted when performing recognition.

In fact, the earliest face recognition systems, such as those by Fischler and Elschlager (1973), Kanade (1977), and Yuille (1991), found distinctive feature points on facial images and performed recognition on the basis of their relative positions or distances. Later techniques such as *local feature analysis* (Penev and Atick 1996) and *elastic bunch graph matching* (Wiskott, Fellous *et al.* 1997) combined local filter responses (jets) at distinctive feature locations together with shape models to perform recognition.

A visually compelling example of why both shape and texture are important is the work of Rowland and Perrett (1995), who manually traced the contours of facial features and then used

[12] https://www.face-rec.org

(a) (b) (c) (d) (e)

Figure 6.14 Manipulating facial appearance through shape and color (Rowland and Perrett 1995) © 1995 IEEE. By adding or subtracting gender-specific shape and color characteristics to an input image (b), different amounts of gender variation can be induced. The amounts added (from the mean) are: (a) +50% (gender enhancement), (b) 0% (original image), (c) –50% (near "androgyny"), (d) –100% (gender switched), and (e) –150% (opposite gender attributes enhanced).

these contours to normalize (warp) each image to a canonical shape. After analyzing both the shape and color images for deviations from the mean, they were able to associate certain shape and color deformations with personal characteristics such as age and gender (Figure 6.14). Their work demonstrates that both shape and color have an important influence on the perception of such characteristics.

Around the same time, researchers in computer vision were beginning to use simultaneous shape deformations and texture interpolation to model the variability in facial appearance caused by identity or expression (Beymer 1996; Vetter and Poggio 1997), developing techniques such as Active Shape Models (Lanitis, Taylor, and Cootes 1997), 3D Morphable Models (Blanz and Vetter 1999; Egger, Smith *et al.* 2020), and Elastic Bunch Graph Matching (Wiskott, Fellous *et al.* 1997).[13]

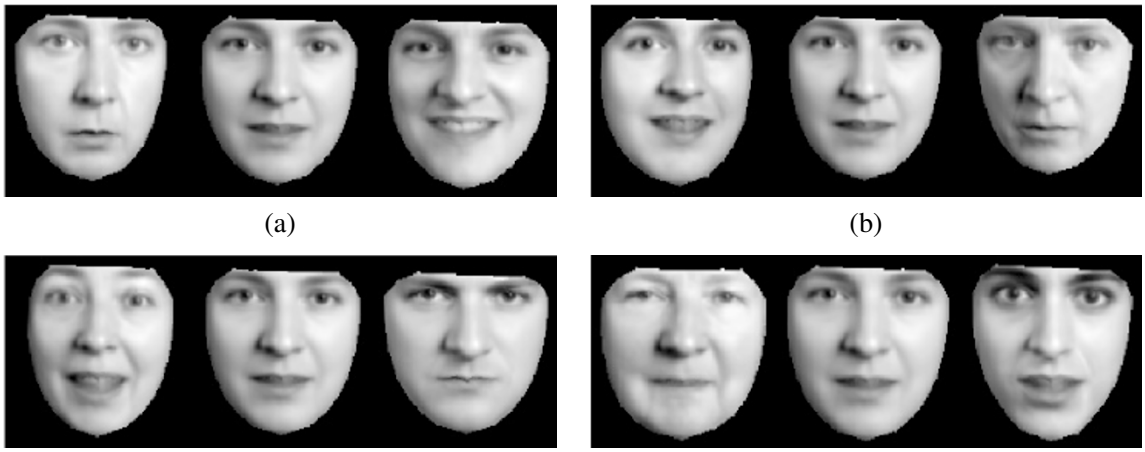
The *active appearance models* (AAMs) of Cootes, Edwards, and Taylor (2001) model both the variation in the shape of an image $\mathbf{s}$, which is normally encoded by the location of key feature points on the image, as well as the variation in texture $\mathbf{t}$, which is normalized to a canonical shape before being analyzed. Both shape and texture are represented as deviations from a mean shape $\bar{\mathbf{s}}$ and texture $\bar{\mathbf{t}}$,

$$\mathbf{s} = \bar{\mathbf{s}} + \mathbf{U}_s \mathbf{a} \tag{6.2}$$

$$\mathbf{t} = \bar{\mathbf{t}} + \mathbf{U}_t \mathbf{a}, \tag{6.3}$$

where the eigenvectors in $\mathbf{U}_s$ and $\mathbf{U}_t$ have been pre-scaled (whitened) so that unit vectors in $\mathbf{a}$ represent one standard deviation of variation observed in the training data. In addition to these principal deformations, the shape parameters are transformed by a global similarity to match the location, size, and orientation of a given face. Similarly, the texture image contains a scale and offset to best match novel illumination conditions.

[13] We will look at the application of PCA to 3D head and face modeling and animation in Section 13.6.3.



(a) (b) (c) (d)

Figure 6.15 Principal modes of variation in active appearance models (Cootes, Edwards, and Taylor 2001) © 2001 IEEE. The four images show the effects of simultaneously changing the first four modes of variation in both shape and texture by $\pm\sigma$ from the mean. You can clearly see how the shape of the face and the shading are simultaneously affected.

As you can see, the same appearance parameters $\mathbf{a}$ in (6.2–6.3) simultaneously control both the shape and texture deformations from the mean, which makes sense if we believe them to be correlated. Figure 6.15 shows how moving three standard deviations along each of the first four principal directions ends up changing several correlated factors in a person's appearance, including expression, gender, age, and identity.

Although active appearance models are primarily designed to accurately capture the variability in appearance and deformation that are characteristic of faces, they can be adapted to face recognition by computing an identity subspace that separates variation in identity from other sources of variability such as lighting, pose, and expression (Costen, Cootes *et al.* 1999). The basic idea, which is modeled after similar work in eigenfaces (Belhumeur, Hespanha, and Kriegman 1997; Moghaddam, Jebara, and Pentland 2000), is to compute separate statistics for intrapersonal and extrapersonal variation and then find discriminating directions in these subspaces. While AAMs have sometimes been used directly for recognition (Blanz and Vetter 2003), their main use in the context of recognition is to align faces into a canonical pose (Liang, Xiao *et al.* 2008; Ren, Cao *et al.* 2014) so that more traditional methods of face recognition (Penev and Atick 1996; Wiskott, Fellous *et al.* 1997; Ahonen, Hadid, and Pietikäinen 2006; Zhao and Pietikäinen 2007; Cao, Yin *et al.* 2010) can be used.

Active appearance models have been extended to deal with illumination and viewpoint variation (Gross, Baker *et al.* 2005) as well as occlusions (Gross, Matthews, and Baker 2006). One of the most significant extensions is to construct 3D models of shape (Matthews, Xiao, and Baker 2007), which are much better at capturing and explaining the full variability of facial appearance across wide changes in pose. Such models can be constructed either from monocular video sequences (Matthews, Xiao, and Baker 2007), as shown in Figure 6.16a, or from multi-view video sequences (Ramnath, Koterba *et al.* 2008), which provide even greater reliability and accuracy in reconstruction and tracking (Murphy-Chutorian and Trivedi 2009).

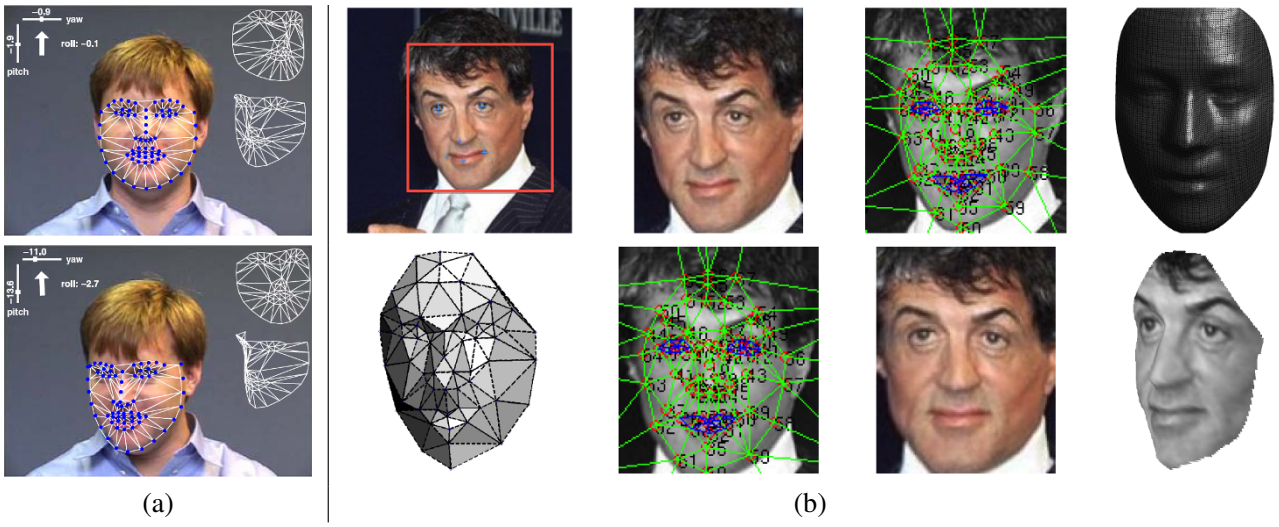


Figure 6.16 Head tracking and frontalization: (a) using 3D active appearance models (AAMs) (Matthews, Xiao, and Baker 2007) © 2007 Springer, showing video frames along with the estimated yaw, pitch, and roll parameters and the fitted 3D deformable mesh; (b) using six and then 67 fiducial points in the DeepFace system (Taigman, Yang *et al.* 2014) © 2014 IEEE, used to frontalize the face image (bottom row).

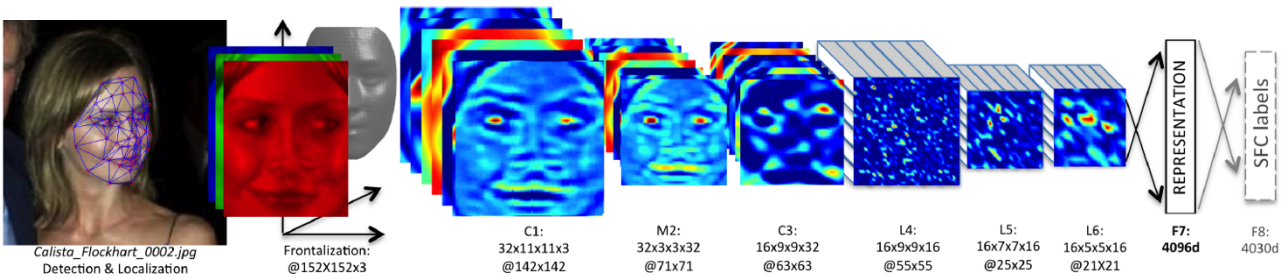


Figure 6.17 The DeepFace architecture (Taigman, Yang *et al.* 2014) © 2014 IEEE, starts with a frontalization stage, followed by several locally connected (non-convolutional) layers, and then two fully connected layers with a K-class softmax.

Facial recognition using deep learning

Prompted by the dramatic success of deep networks in whole-image categorization, face recognition researchers started using deep neural network backbones as part of their systems. Figures 6.16b–6.17 shows two stages in the DeepFace system of Taigman, Yang *et al.* (2014), which was one of the first systems to realize large gains using deep networks. In their system, a landmark-based pre-processing *frontalization* step is used to convert the original color image into a well-cropped front-looking face. Then, a deep locally connected network (where the convolution kernels can vary spatially) is fed into two final fully connected layers before classification.

Some of the more recent deep face recognizers omit the frontalization stage and instead use data augmentation (Section 5.3.3) to create synthetic inputs with a larger variety of poses (Schroff, Kalenichenko, and Philbin 2015; Parkhi, Vedaldi, and Zisserman 2015). Masi, Wu *et al.* (2018) provide an excellent tutorial and survey on deep face recognition, including a list of widely used training and testing datasets, a discussion of frontalization and dataset augmentation, and a section on training losses (Figure 6.18). This last topic is central to the ability to scale to larger and larger

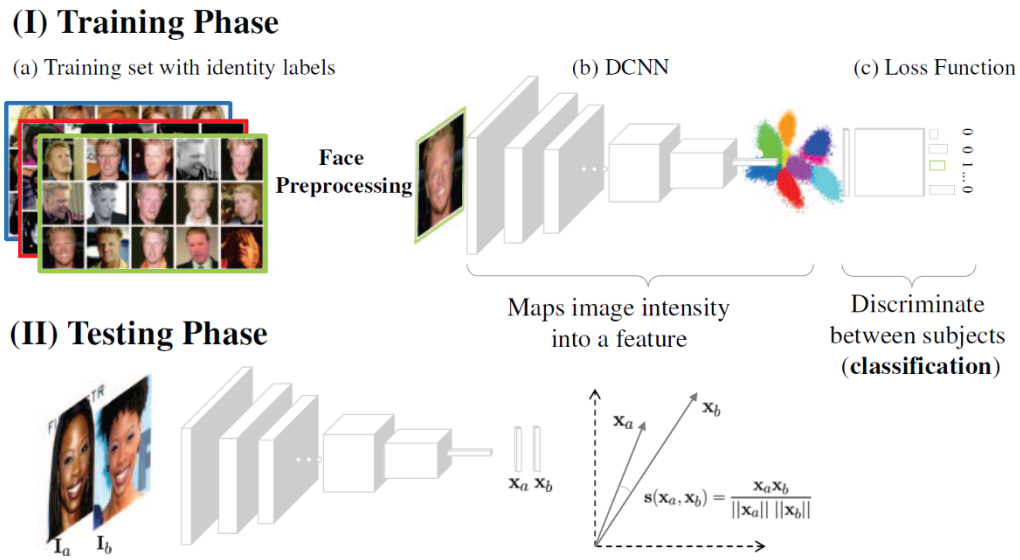


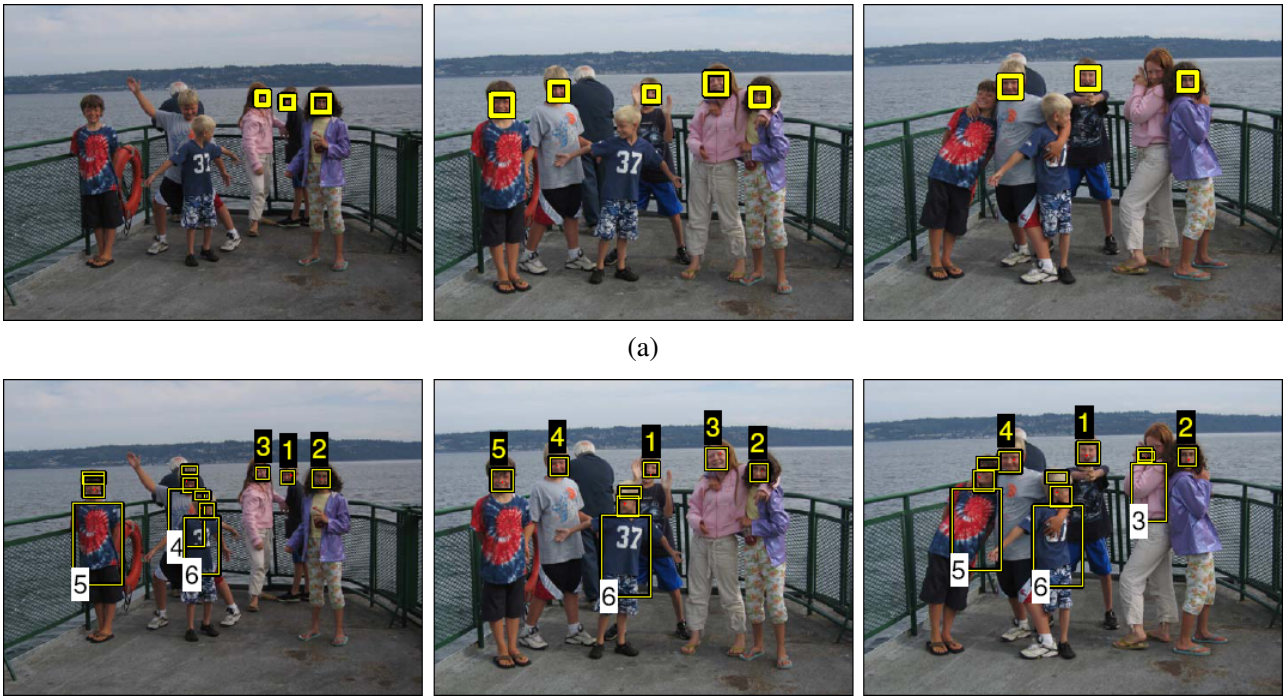
Figure 6.18 A typical modern deep face recognition architecture, from the survey by Masi, Wu *et al.* (2018) © 2018 IEEE. At training time, a huge labeled face set (a) is used to constrain the weights of a DCNN (b), optimizing a loss function (c) for a classification task. At test time, the classification layer is often discarded, and the DCNN is used as a feature extractor for comparing face descriptors.

numbers of people. Schroff, Kalenichenko, and Philbin (2015) and Parkhi, Vedaldi, and Zisserman (2015) use triplet losses to construct a low-dimensional embedding space that is independent of the number of subjects. More recent systems use contrastive losses inspired by the softmax function, which we discussed in Section 5.3.4. For example, the ArcFace paper by Deng, Guo *et al.* (2019) measures angular distances on the unit hypersphere in the embedding space and adds an extra margin to get identities to clump together. This idea has been further extended for visual similarity search (Bell, Liu *et al.* 2020) and face recognition (Huang, Shen *et al.* 2020; Deng, Guo *et al.* 2020a).

Personal photo collections

In addition to digital cameras automatically finding faces to aid in auto-focusing and video cameras finding faces in video conferencing to center on the speaker (either mechanically or digitally), face detection has found its way into most consumer-level photo organization packages and photo sharing sites. Finding faces and allowing users to tag them makes it easier to find photos of selected people at a later date or to automatically share them with friends. In fact, the ability to tag friends in photos is one of the more popular features on Facebook.

Sometimes, however, faces can be hard to find and recognize, especially if they are small, turned away from the camera, or otherwise occluded. In such cases, combining face recognition with person detection and clothes recognition can be very effective, as illustrated in Figure 6.19 (Sivic, Zitnick, and Szeliski 2006). Combining person recognition with other kinds of context, such as location recognition (Section 11.2.3) or activity or event recognition, can also help boost performance (Lin, Kapoor *et al.* 2010).



(a)

(b)

Figure 6.19 Person detection and re-recognition using a combined face, hair, and torso model (Sivic, Zitnick, and Szeliski 2006) © 2006 Springer. (a) Using face detection alone, several of the heads are missed. (b) The combined face and clothing model successfully re-finds all the people.

6.3 Object detection

If we are given an image to analyze, such as the group portrait in Figure 6.20, we could try to apply a recognition algorithm to every possible sub-window in this image. Such algorithms are likely to be both slow and error-prone. Instead, it is more effective to construct special-purpose *detectors*, whose job it is to rapidly find likely regions where particular objects might occur.

We begin this section with face detectors, which were some of the earliest successful examples of recognition. Such algorithms are built into most of today's digital cameras to enhance auto-focus and into video conferencing systems to control panning and zooming. We then look at pedestrian detectors, as an example of more general methods for object detection. Finally, we turn to the problem of multi-class object detection, which today is solved using deep neural networks.

6.3.1 Face detection

Before face recognition can be applied to a general image, the locations and sizes of any faces must first be found (Figures 6.1c and 6.20). In principle, we could apply a face recognition algorithm at every pixel and scale (Moghaddam and Pentland 1997) but such a process would be too slow in practice.

Over the last four decades, a wide variety of fast face detection algorithms have been developed. Yang, Kriegman, and Ahuja (2002) and Zhao, Chellappa *et al.* (2003) provide comprehensive surveys of earlier work in this field. According to their taxonomy, face detection techniques can be classified as feature-based, template-based, or appearance-based. Feature-based techniques attempt

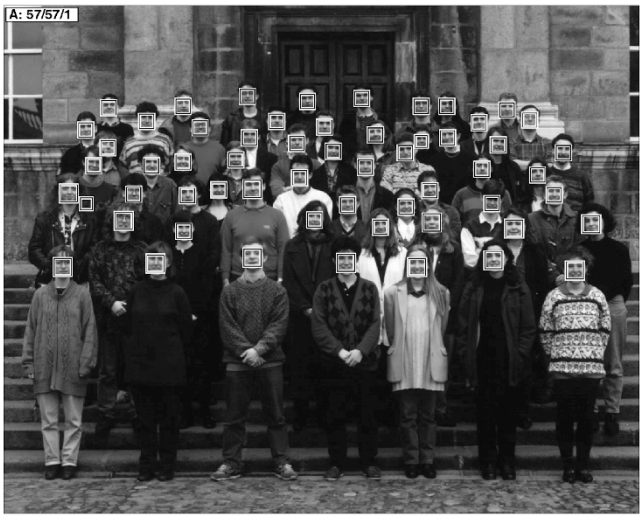


Figure 6.20 Face detection results produced by Rowley, Baluja, and Kanade (1998) © 1998 IEEE. Can you find the one false positive (a box around a non-face) among the 57 true positive results?

to find the locations of distinctive image features such as the eyes, nose, and mouth, and then verify whether these features are in a plausible geometrical arrangement. These techniques include some of the early approaches to face recognition (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991), as well as later approaches based on modular eigenspaces (Moghaddam and Pentland 1997), local filter jets (Leung, Burl, and Perona 1995; Penev and Atick 1996; Wiskott, Fellous *et al.* 1997), support vector machines (Heisele, Ho *et al.* 2003; Heisele, Serre, and Poggio 2007), and boosting (Schneiderman and Kanade 2004).

Template-based approaches, such as active appearance models (AAMs) (Section 6.2.4), can deal with a wide range of pose and expression variability. Typically, they require good initialization near a real face and are therefore not suitable as fast face detectors.

Appearance-based approaches scan over small overlapping rectangular patches of the image searching for likely face candidates, which can then be refined using a *cascade* of more expensive but selective detection algorithms (Sung and Poggio 1998; Rowley, Baluja, and Kanade 1998; Romdhani, Torr *et al.* 2001; Fleuret and Geman 2001; Viola and Jones 2004). To deal with scale variation, the image is usually converted into a sub-octave pyramid and a separate scan is performed on each level. Most appearance-based approaches rely heavily on training classifiers using sets of labeled face and non-face patches.

Sung and Poggio (1998) and Rowley, Baluja, and Kanade (1998) present two of the earliest appearance-based face detectors and introduce a number of innovations that are widely used in later work by others. To start with, both systems collect a set of labeled face patches (Figure 6.20) as well as a set of patches taken from images that are known not to contain faces, such as aerial images or vegetation. The collected face images are augmented by artificially mirroring, rotating, scaling, and translating the images by small amounts to make the face detectors less sensitive to such effects.

The next few paragraphs provide quick reviews of a number of early appearance-based face detectors, keyed by the machine algorithms they are based on. These systems provide an interesting glimpse into the gradual adoption and evolution of machine learning in computer vision. More detailed descriptions can be found in the original papers, as well as the first edition of this book (Szeliski 2010).

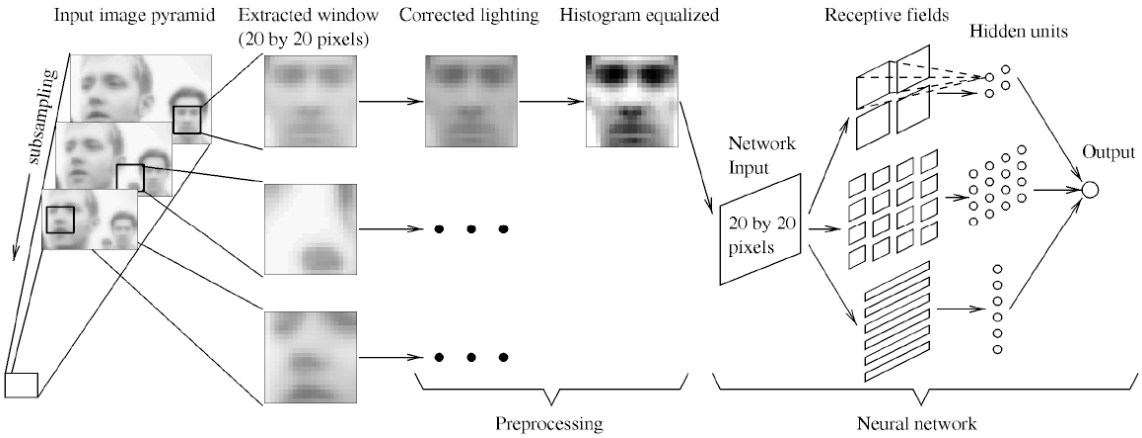


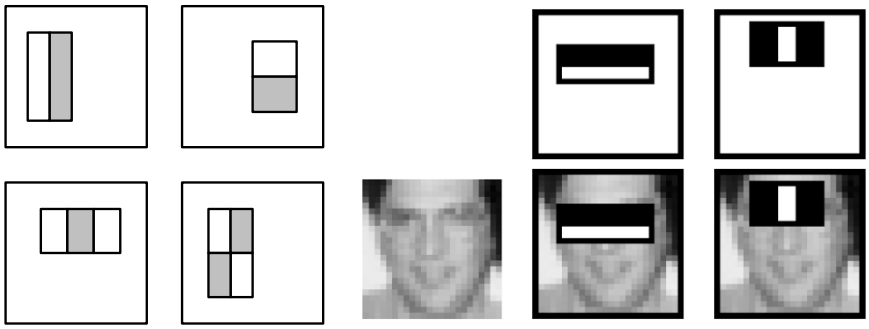
Figure 6.21 A neural network for face detection (Rowley, Baluja, and Kanade 1998) © 1998 IEEE. Overlapping patches are extracted from different levels of a pyramid and then pre-processed. A three-layer neural network is then used to detect likely face locations.

Clustering and PCA. Once the face and non-face patterns have been pre-processed, Sung and Poggio (1998) cluster each of these datasets into six separate clusters using k-means and then fit PCA subspaces to each of the resulting 12 clusters. At detection time, the DIFS and DFFS metrics first developed by Moghaddam and Pentland (1997) are used to produce 24 Mahalanobis distance measurements (two per cluster). The resulting 24 measurements are input to a multi-layer perceptron (MLP), i.e., a fully connected neural network.

Neural networks. Instead of first clustering the data and computing Mahalanobis distances to the cluster centers, Rowley, Baluja, and Kanade (1998) apply a neural network (MLP) directly to the 20×20 pixel patches of gray-level intensities, using a variety of differently sized hand-crafted "receptive fields" to capture both large-scale and smaller scale structure (Figure 6.21). The resulting neural network directly outputs the likelihood of a face at the center of every overlapping patch in a multi-resolution pyramid. Since several overlapping patches (in both space and resolution) may fire near a face, an additional merging network is used to merge overlapping detections. The authors also experiment with training several networks and merging their outputs. Figure 6.20 shows a sample result from their face detector.

Support vector machines. Instead of using a neural network to classify patches, Osuna, Freund, and Girosi (1997) use *support vector machines* (SVMs), which we discussed in Section 5.1.4, to classify the same preprocessed patches as Sung and Poggio (1998). An SVM searches for a series of *maximum margin* separating planes in feature space between different classes (in this case, face and non-face patches). In those cases where linear classification boundaries are insufficient, the feature space can be lifted into higher-dimensional features using *kernels* (5.29). SVMs have been used by other researchers for both face detection and face recognition (Heisele, Ho *et al.* 2003; Heisele, Serre, and Poggio 2007) as well as general object recognition (Lampert 2008).

Boosting. Of all the face detectors developed in the 2000s, the one introduced by Viola and Jones (2004) is probably the best known. Their technique was the first to introduce the concept of *boosting* to the computer vision community, which involves training a series of increasingly discriminating



(a) (b)

Figure 6.22 Simple features used in boosting-based face detector (Viola and Jones 2004) © 2004 Springer: (a) difference of rectangle feature composed of 2–4 different rectangles (pixels inside the white rectangles are subtracted from the gray ones); (b) the first and second features selected by AdaBoost. The first feature measures the differences in intensity between the eyes and the cheeks, the second one between the eyes and the bridge of the nose.

simple classifiers and then blending their outputs (Bishop 2006, Section 14.3; Hastie, Tibshirani, and Friedman 2009, Chapter 10; Murphy 2012, Section 16.4; Glassner 2018, Section 14.7).

In more detail, boosting involves constructing a *classifier* $h(\mathbf{x})$ as a sum of simple *weak learners*,

$$h(\mathbf{x}) = \text{sign}\left[\sum_{j=0}^{m-1} \alpha_j h_j(\mathbf{x})\right], \tag{6.4}$$

where each of the weak learners $h_j(\mathbf{x})$ is an extremely simple function of the input, and hence is not expected to contribute much (in isolation) to the classification performance.

In most variants of boosting, the weak learners are threshold functions,

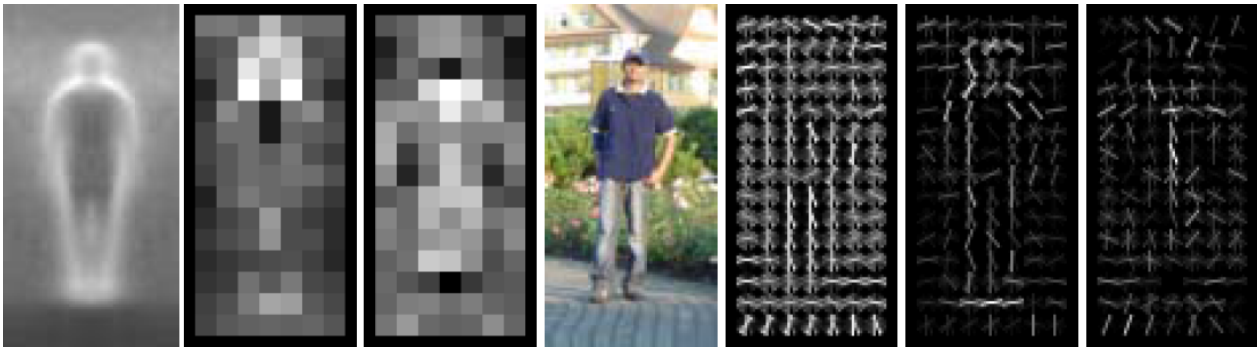
$$h_j(\mathbf{x}) = a_j[f_j < \theta_j] + b_j[f_j \geq \theta_j] = \begin{cases} a_j & \text{if} \quad f_j < \theta_j \\ b_j & \text{otherwise,} \end{cases} \tag{6.5}$$

which are also known as *decision stumps* (basically, the simplest possible version of *decision trees*). In most cases, it is also traditional (and simpler) to set a_j and b_j to ± 1, i.e., $a_j = -s_j$, $b_j = +s_j$, so that only the feature f_j, the threshold value θ_j, and the polarity of the threshold $s_j \in \pm 1$ need to be selected.[14]

In many applications of boosting, the features are simply coordinate axes x_k, i.e., the boosting algorithm selects one of the input vector components as the best one to threshold. In Viola and Jones' face detector, the features are differences of rectangular regions in the input patch, as shown in Figure 6.22. The advantage of using these features is that, while they are more discriminating than single pixels, they are extremely fast to compute once a summed area table has been precomputed, as described in Section 3.2.3 (3.31–3.32). Essentially, for the cost of an $O(N)$ precomputation phase (where N is the number of pixels in the image), subsequent differences of rectangles can be computed in $4r$ additions or subtractions, where $r \in \{2, 3, 4\}$ is the number of rectangles in the feature.

The key to the success of boosting is the method for incrementally selecting the weak learners and for re-weighting the training examples after each stage. The AdaBoost (Adaptive Boosting)

[14] Some variants, such as that of Viola and Jones (2004), use $(a_j, b_j) \in [0, 1]$ and adjust the learning algorithm accordingly.



(a) (b) (c) (d) (e) (f) (g)

Figure 6.23 Pedestrian detection using histograms of oriented gradients (Dalal and Triggs 2005) © 2005 IEEE: (a) the average gradient image over the training examples; (b) each "pixel" shows the maximum positive SVM weight in the block centered on the pixel; (c) likewise, for the negative SVM weights; (d) a test image; (e) the computed R-HOG (rectangular histogram of gradients) descriptor; (f) the R-HOG descriptor weighted by the positive SVM weights; (g) the R-HOG descriptor weighted by the negative SVM weights.

algorithm (Bishop 2006; Hastie, Tibshirani, and Friedman 2009; Murphy 2012) does this by re-weighting each sample as a function of whether it is correctly classified at each stage, and using the stage-wise average classification error to determine the final weightings α_j among the weak classifiers.

To further increase the speed of the detector, it is possible to create a *cascade* of classifiers, where each classifier uses a small number of tests (say, a two-term AdaBoost classifier) to reject a large fraction of non-faces while trying to pass through all potential face candidates (Fleuret and Geman 2001; Viola and Jones 2004; Brubaker, Wu *et al.* 2008).

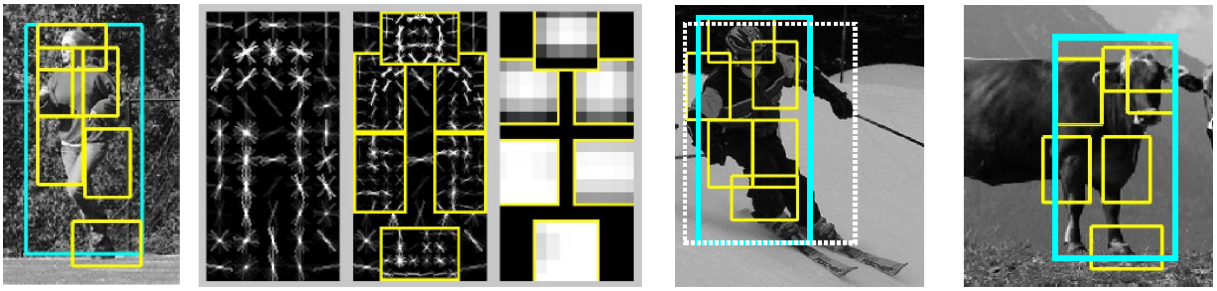
Deep networks. Since the initial burst of face detection research in the early 2000s, face detection algorithms have continued to evolve and improve (Zafeiriou, Zhang, and Zhang 2015). Researchers have proposed using cascades of features (Li and Zhang 2013), deformable parts models (Mathias, Benenson *et al.* 2014), aggregated channel features (Yang, Yan *et al.* 2014), and neural networks (Li, Lin *et al.* 2015; Yang, Luo *et al.* 2015). The WIDER FACE benchmark[15,16] (Yang, Luo *et al.* 2016) contains results from, and pointers to, more recent papers, including RetinaFace (Deng, Guo *et al.* 2020b), which combines ideas from other recent neural networks and object detectors such as Feature Pyramid Networks (Lin, Dollár *et al.* 2017) and RetinaNet (Lin, Goyal *et al.* 2017), and also has a nice review of other recent face detectors.

6.3.2 Pedestrian detection

While a lot of the early research on object detection focused on faces, the detection of other objects, such as pedestrians and cars, has also received widespread attention (Gavrila and Philomin 1999; Gavrila 1999; Papageorgiou and Poggio 2000; Mohan, Papageorgiou, and Poggio 2001; Schneiderman and Kanade 2004). Some of these techniques maintained the same focus as face detection on speed and efficiency. Others, however, focused on accuracy, viewing detection as a more challenging

[15] http://shuoyang1213.me/WIDERFACE

[16] The WIDER FACE benchmark has expanded to a larger set of detection challenges and workshops: https://wider-challenge.org/2019.html.



(a) (b) (c) (d)

Figure 6.24 Part-based object detection (Felzenszwalb, McAllester, and Ramanan 2008) © 2008 IEEE: (a) An input photograph and its associated person (blue) and part (yellow) detection results. (b) The detection model is defined by a coarse template, several higher resolution part templates, and a spatial model for the location of each part. (c) True positive detection of a skier and (d) false positive detection of a cow (labeled as a person).

variant of generic class recognition (Section 6.3.3) in which the locations and extents of objects are to be determined as accurately as possible (Everingham, Van Gool *et al.* 2010; Everingham, Eslami *et al.* 2015; Lin, Maire *et al.* 2014).

An example of a well-known pedestrian detector is the algorithm developed by Dalal and Triggs (2005), who use a set of overlapping *histogram of oriented gradients* (HOG) descriptors fed into a support vector machine (Figure 6.23). Each HOG has cells to accumulate magnitude-weighted votes for gradients at particular orientations, just as in the scale invariant feature transform (SIFT) developed by Lowe (2004), which we will describe in Section 7.1.2 and Figure 7.16. Unlike SIFT, however, which is only evaluated at interest point locations, HOGs are evaluated on a regular overlapping grid and their descriptor magnitudes are normalized using an even coarser grid; they are only computed at a single scale and a fixed orientation. To capture the subtle variations in orientation around a person's outline, a large number of orientation bins are used and no smoothing is performed in the central difference gradient computation—see Dalal and Triggs (2005) for more implementation details. Figure 6.23d shows a sample input image, while Figure 6.23e shows the associated HOG descriptors.

Once the descriptors have been computed, a support vector machine (SVM) is trained on the resulting high-dimensional continuous descriptor vectors. Figures 6.23b–c show a diagram of the (most) positive and negative SVM weights in each block, while Figures 6.23f–g show the corresponding weighted HOG responses for the central input image. As you can see, there are a fair number of positive responses around the head, torso, and feet of the person, and relatively few negative responses (mainly around the middle and the neck of the sweater).

Much like face detection, the fields of pedestrian and general object detection continued to advance rapidly in the 2000s (Belongie, Malik, and Puzicha 2002; Mikolajczyk, Schmid, and Zisserman 2004; Dalal and Triggs 2005; Leibe, Seemann, and Schiele 2005; Opelt, Pinz, and Zisserman 2006; Torralba 2007; Andriluka, Roth, and Schiele 2009; Maji and Berg 2009; Andriluka, Roth, and Schiele 2010; Dollár, Belongie, and Perona 2010).

A significant advance in the field of person detection was the work of Felzenszwalb, McAllester, and Ramanan (2008), who extend the histogram of oriented gradients person detector to incorporate flexible parts models (Section 6.2.1). Each part is trained and detected on HOGs evaluated at two pyramid levels below the overall object model and the locations of the parts relative to the parent node (the overall bounding box) are also learned and used during recognition (Figure 6.24b). To compensate for inaccuracies or inconsistencies in the training example bounding boxes (dashed

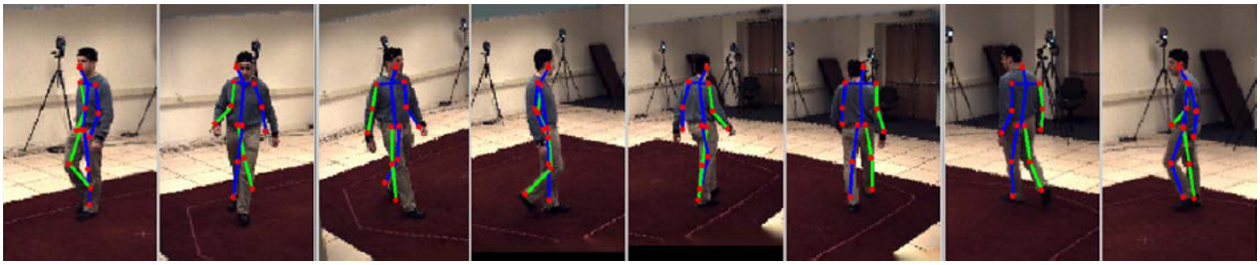


Figure 6.25 Pose detection using random forests (Rogez, Rihan *et al.* 2008) © 2008 IEEE. The estimated pose (state of the kinematic model) is drawn over each input frame.

white lines in Figure 6.24c), the "true" location of the parent (blue) bounding box is considered a latent (hidden) variable and is inferred during both training and recognition. Since the locations of the parts are also latent, the system can be trained in a semi-supervised fashion, without needing part labels in the training data. An extension to this system (Felzenszwalb, Girshick *et al.* 2010), which includes among its improvements a simple contextual model, was among the two best object detection systems in the 2008 Visual Object Classes detection challenge (Everingham, Van Gool *et al.* 2010). Improvements to part-based person detection and pose estimation include work by Andriluka, Roth, and Schiele (2009) and Kumar, Zisserman, and Torr (2009).

An even more accurate estimate of a person's pose and location is presented by Rogez, Rihan *et al.* (2008), who compute both the phase of a person in a walk cycle and the locations of individual joints, using random forests built on top of HOGs (Figure 6.25). Since their system produces full 3D pose information, it is closer in its application domain to 3D person trackers (Sidenbladh, Black, and Fleet 2000; Andriluka, Roth, and Schiele 2010), which we will discussed in Section 13.6.4. When video sequences are available, the additional information present in the optical flow and motion discontinuities can greatly aid in the detection task, as discussed by Efros, Berg *et al.* (2003), Viola, Jones, and Snow (2003), and Dalal, Triggs, and Schmid (2006).

Since the 2000s, pedestrian and general person detection have continued to be actively developed, often in the context of more general multi-class object detection (Everingham, Van Gool *et al.* 2010; Everingham, Eslami *et al.* 2015; Lin, Maire *et al.* 2014). The Caltech pedestrian detection benchmark[17] and survey by Dollár, Belongie, and Perona (2010) introduces a new dataset and provides a nice review of algorithms through 2012, including Integral Channel Features (Dollár, Tu *et al.* 2009), the Fastest Pedestrian Detector in the West (Dollár, Belongie, and Perona 2010), and 3D pose estimation algorithms such as Poselets (Bourdev and Malik 2009). Since its original construction, this benchmark continues to tabulate and evaluate more recent detectors, including Dollár, Appel, and Kienzle (2012), Dollár, Appel *et al.* (2014), and more recent algorithms based on deep neural networks (Sermanet, Kavukcuoglu *et al.* 2013; Ouyang and Wang 2013; Tian, Luo *et al.* 2015; Zhang, Lin *et al.* 2016). The CityPersons dataset (Zhang, Benenson, and Schiele 2017) and WIDER Face and Person Challenge[18] also report results on recent algorithms.

6.3.3 General object detection

While face and pedestrian detection algorithms were the earliest to be extensively studied, computer vision has always been interested in solving the general object detection and labeling problem, in

[17] http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians
[18] https://wider-challenge.org/2019.html

Figure 6.26 Intersection over union (IoU): (a) schematic formula, (b) real-world example © 2020 Ross Girshick.

addition to whole-image classification. The PASCAL Visual Object Classes (VOC) Challenge (Everingham, Van Gool *et al.* 2010), which contained 20 classes, had both classification and detection challenges. Early entries that did well on the detection challenge include a feature-based detector and spatial pyramid matching SVM classifier by Chum and Zisserman (2007), a star-topology deformable part model by Felzenszwalb, McAllester, and Ramanan (2008), and a sliding window SVM classifier by Lampert, Blaschko, and Hofmann (2008). The competition was re-run annually, with the two top entries in the 2012 detection challenge (Everingham, Eslami *et al.* 2015) using a sliding window spatial pyramid matching (SPM) SVM (de Sande, Uijlings *et al.* 2011) and a University of Oxford re-implementation of a deformable parts model (Felzenszwalb, Girshick *et al.* 2010).

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC), released in 2010, scaled up the dataset from around 20 thousand images in PASCAL VOC 2010 to over 1.4 million in ILSVRC 2010, and from 20 object classes to 1,000 object classes (Russakovsky, Deng *et al.* 2015). Like PASCAL, it also had an object detection task, but it contained a much wider range of challenging images (Figure 6.4). The Microsoft COCO (Common Objects in Context) dataset (Lin, Maire *et al.* 2014) contained even more objects per image, as well as pixel-accurate segmentations of multiple objects, enabling the study of not only *semantic segmentation* (Section 6.4), but also individual object *instance segmentation* (Section 6.4.2). Table 6.2 list some of the datasets used for training and testing general object detection algorithms.

The release of COCO coincided with a wholescale shift to deep networks for image classification, object detection, and segmentation (Jiao, Zhang *et al.* 2019; Zhao, Zheng *et al.* 2019). Figure 6.29 shows the rapid improvements in average precision (AP) on the COCO object detection task, which correlates strongly with advances in deep neural network architectures (Figure 5.40).

Precision vs. recall

Before we describe the elements of modern object detectors, we should first discuss what metrics they are trying to optimize. The main task in object detection, as illustrated in Figures 6.5a and 6.26b, is to put accurate bounding boxes around all the objects of interest and to correctly label such objects. To measure the accuracy of each bounding box (not too small and not too big), the common metric is *intersection over union* (IoU), which is also known as the *Jaccard index* or *Jaccard similarity coefficient* (Rezatofighi, Tsoi *et al.* 2019). The IoU is computed by taking the predicted and ground truth bounding boxes B_{pr} and B_{gt} for an object and computing the ratio of their area of

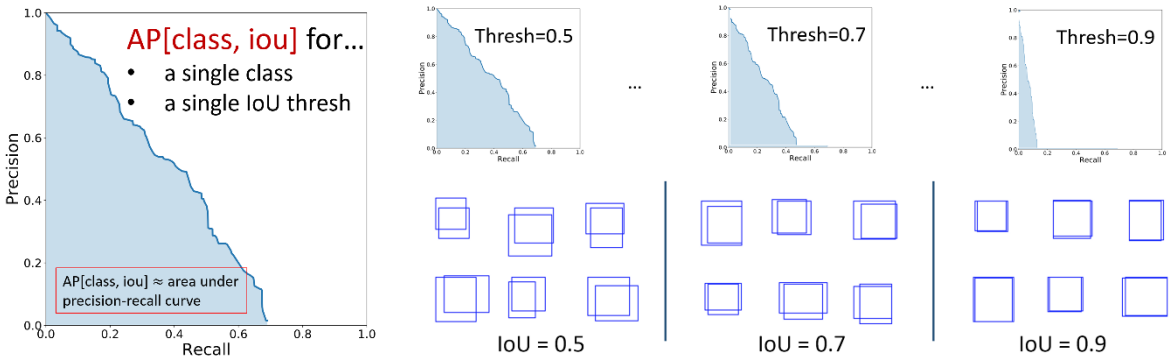


Figure 6.27 Object detector average precision © 2020 Ross Girshick: (a) a precision-recall curve for a single class and IoU threshold, with the AP being the area under the P-R curve; (b) average precision averaged over several IoU thresholds (from looser to tighter).

intersection and their area of union,

$$IoU = \frac{B_{\mathrm{pr}} \cap B_{\mathrm{gt}}}{B_{\mathrm{pr}} \cup B_{\mathrm{gt}}}, \tag{6.6}$$

as shown in Figure 6.26a.

As we will shortly see, object detectors operate by first proposing a number of plausible rectangular regions (detections) and then classifying each detection while also producing a confidence score (Figure 6.26b). These regions are then run through some kind of *non-maximal suppression* (NMS) stage, which removes weaker detections that have too much overlap with stronger detections, using a greedy most-confident-first algorithm.

To evaluate the performance of an object detector, we run through all of the detections, from most confident to least, and classify them as *true positive* TP (correct label and sufficiently high IoU) or *false positive* FP (incorrect label or ground truth object already matched). For each new decreasing confidence threshold, we can compute the *precision* and *recall* as

$$\text{precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{6.7}$$

$$\text{recall} = \frac{\text{TP}}{\text{P}}, \tag{6.8}$$

where P is the number of positive examples, i.e., the number of labeled ground truth detections in the test image.[19] (See Section 7.1.3 on feature matching for additional terms that are often used in measuring and describing error rates.)

Computing the precision and recall at every confidence threshold allows us to populate a precision-recall curve, such as the one in Figure 6.27a. The area under this curve is called *average precision* (AP). A separate AP score can be computed for each class being detected, and the results averaged to produce a *mean average precision* (mAP). Another widely used measure if the While earlier benchmarks such as PASCAL VOC determined the mAP using a single IoU threshold of 0.5 (Everingham, Eslami *et al.* 2015), the COCO benchmark (Lin, Maire *et al.* 2014) averages the mAP over a set of IoU thresholds, IoU $\in \{0.50, 0.55, \ldots, 0.95\}$, as shown in Figure 6.27a. While this AP score

[19] Another widely reported measure is the *F-score*, which is the harmonic mean of the precision and recall.

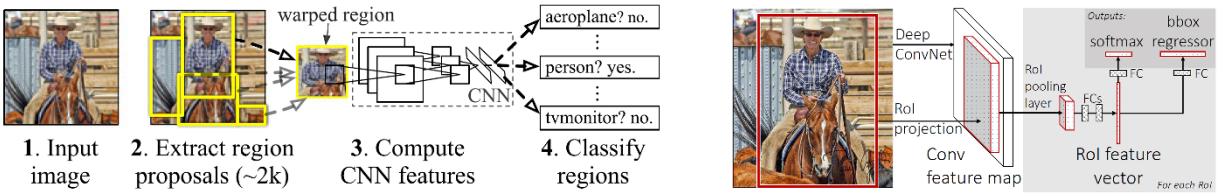


Figure 6.28 The R-CNN and Fast R-CNN object detectors. (a) R-CNN rescales pixels inside each proposal region and performs a CNN + SVM classification (Girshick, Donahue *et al.* 2015) © 2015 IEEE. (b) Fast R-CNN resamples convolutional features and uses fully connected layers to perform classification and bounding box regression (Girshick 2015) © 2015 IEEE.

continues to be widely used, an alternative *probability-based detection quality* (PDQ) score has recently been proposed (Hall, Dayoub *et al.* 2020). A smoother version of average precision called Smooth-AP has also been proposed and shown to have benefits on large-scale image retrieval tasks (Brown, Xie *et al.* 2020).

Modern object detectors

The first stage in detecting objects in an image is to propose a set of plausible rectangular regions in which to run a classifier. The development of such *region proposal* algorithms was an active research area in the early 2000s (Alexe, Deselaers, and Ferrari 2012; Uijlings, Van De Sande *et al.* 2013; Cheng, Zhang *et al.* 2014; Zitnick and Dollár 2014).

One of the earliest object detectors based on neural networks is R-CNN, the Region-based Convolutional Network developed by Girshick, Donahue *et al.* (2014). As illustrated in Figure 6.28a, this detector starts by extracting about 2,000 region proposals using the selective search algorithm of Uijlings, Van De Sande *et al.* (2013). Each proposed regions is then rescaled (warped) to a 224 square image and passed through an AlexNet or VGG neural network with a support vector machine (SVM) final classifier.

The follow-on Fast R-CNN paper by Girshick (2015) interchanges the convolutional neural network and region extraction stages and replaces the SVM with some fully connected (FC) layers, which compute both an object class and a bounding box refinement (Figure 6.28b). This reuses the CNN computations and leads to much faster training and test times, as well as dramatically better accuracy compared to previous networks (Figure 6.29). As you can see from Figure 6.28b, Fast R-CNN is an example of a deep network with a shared *backbone* and two separate *heads*, and hence two different loss functions, although these terms were not introduced until the Mask R-CNN paper by He, Gkioxari *et al.* (2017).

The Faster R-CNN system, introduced a few month later by Ren, He *et al.* (2015), replaces the relatively slow selective search stage with a convolutional region proposal network (RPN), resulting in much faster inference. After computing convolutional features, the RPN suggests at each coarse location a number of potential *anchor boxes*, which vary in shape and size to accommodate different potential objects. Each proposal is then classified and refined by an instance of the Fast R-CNN heads and the final detections are ranked and merged using non-maximal suppression.

R-CNN, Fast R-CNN, and Faster R-CNN all operate on a single resolution convolutional feature map (Figure 6.30b). To obtain better scale invariance, it would be preferable to operate on a range of resolutions, e.g, by computing a feature map at each image pyramid level, as shown in Figure 6.30a, but this is computationally expensive. We could, instead, simply start with the various

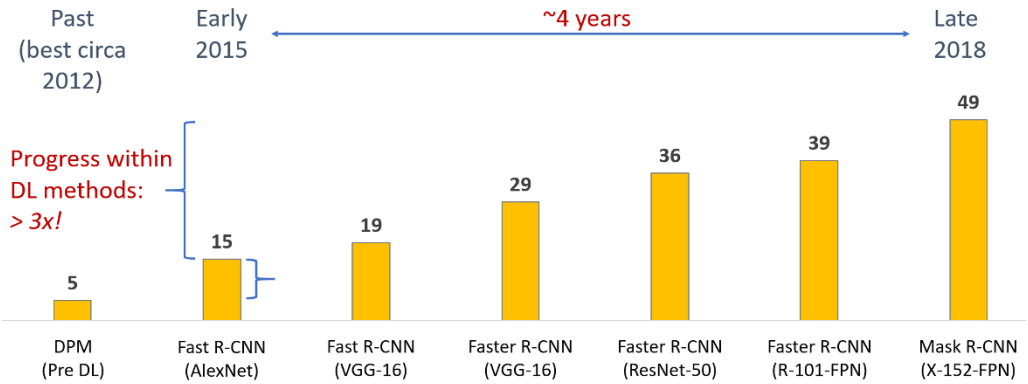


Figure 6.29 Best average precision (AP) results by year on the COCO object detection task (Lin, Maire *et al.* 2014) © 2020 Ross Girshick.

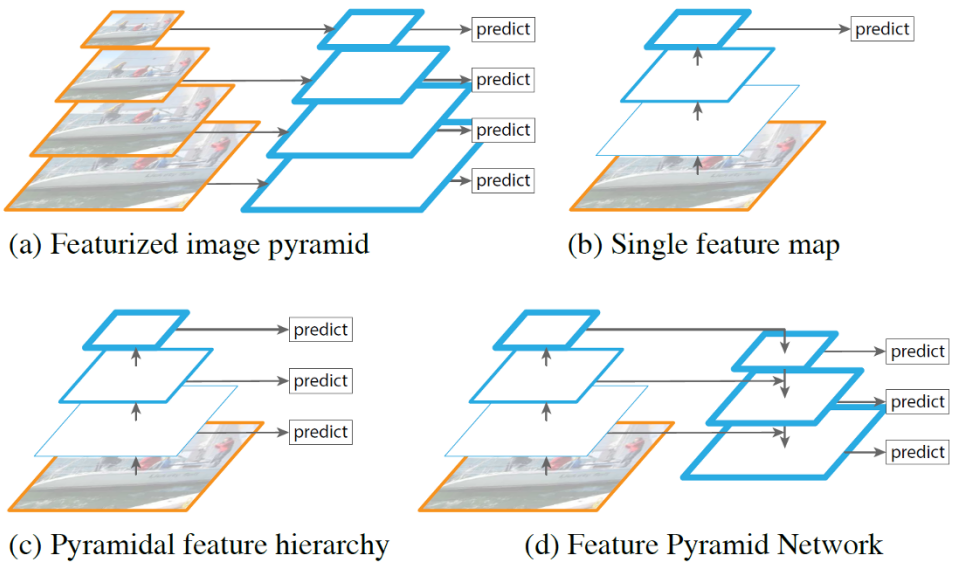


Figure 6.30 A Feature Pyramid Network and its precursors (Lin, Dollár *et al.* 2017) © 2017 IEEE: (a) deep features extracted at each level in an image pyramid; (b) a single low-resolution feature map; (c) a deep feature pyramid, with higher levels having greater abstraction; (d) a Feature Pyramid Network, with top-down context for all levels.

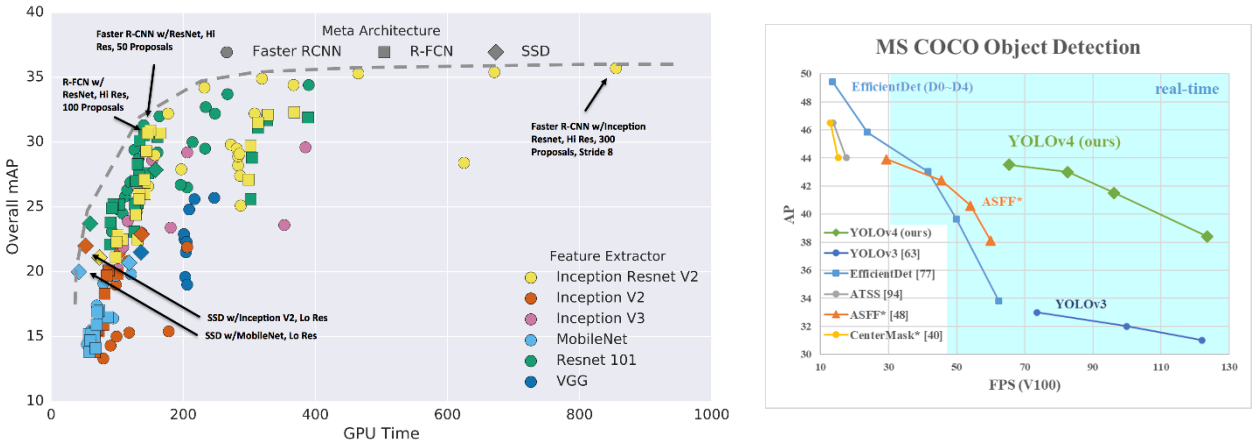


Figure 6.31 Speed/accuracy trade-offs for convolutional object detectors: (a) (Huang, Rathod *et al.* 2017) © 2017 IEEE; (b) YOLOv4 © Bochkovskiy, Wang, and Liao (2020).

levels inside the convolutional network (Figure 6.30c), but these levels have different degrees of semantic abstraction, i.e., higher/smaller levels are attuned to more abstract constructs. The best solution is to construct a *Feature Pyramid Network* (FPN), as shown in Figure 6.30d, where top-down connections are used to endow higher-resolution (lower) pyramid levels with the semantics inferred at higher levels (Lin, Dollár *et al.* 2017).[20] This additional information significantly enhances the performance of object detectors (and other downstream tasks) and makes their behavior much less sensitive to object size.

DETR (Carion, Massa *et al.* 2020) uses a simpler architecture that eliminates the use of non-maximum suppression and anchor generation. Their model consists of a ResNet backbone that feeds into a transformer encoder-decoder. At a high level, it makes N bounding box predictions, some of which may include the "no object class". The ground truth bounding boxes are also padded with "no object class" bounding boxes to obtain N total bounding boxes. During training, *bipartite matching* is then used to build a one-to-one mapping from every predicted bounding box to a ground truth bounding box, with the chosen mapping leading to the lowest possible cost. The overall training loss is then the sum of the losses between the matched bounding boxes. They find that their approach is competitive with state-of-the-art object detection performance on COCO.

Single-stage networks

In the architectures we've looked at so far, a region proposal algorithm or network selects the locations and shapes of the detections to be considered, and a second network is then used to classify and regress the pixels or features inside each region. An alternative is to use a *single-stage network*, which uses a single neural network to output detections at a variety of locations. Two examples of such detectors are SSD (Single Shot MultiBox Detector) from Liu, Anguelov *et al.* (2016) and the family of YOLO (You Only Look Once) detectors described in Redmon, Divvala *et al.* (2016),Redmon and Farhadi (2017), and Redmon and Farhadi (2018). RetinaNet (Lin, Goyal *et al.* 2017) is also a single-stage detector built on top of a feature pyramid network. It uses a *focal loss* to focus the training on hard examples by downweighting the loss on well-classified samples, thus preventing

[20] It's interesting to note that the human visual system is full of such re-entrant or feedback pathways (Gilbert and Li 2013), although the extent to which cognition influences perception is still being debated (Firestone and Scholl 2016).

(a) (b) (c) (d)

Figure 6.32 Examples of image segmentation (Kirillov, He *et al.* 2019) © 2019 IEEE: (a) original image; (b) semantic segmentation (per-pixel classification); (c) instance segmentation (delineate each object); (d) panoptic segmentation (label all things and stuff).

the larger number of easy negatives from overwhelming the training. These and more recent convolutional object detectors are described in the recent survey by Jiao, Zhang *et al.* (2019). Figure 6.31 shows the speed and accuracy of detectors published up through early 2017.

The latest in the family of YOLO detectors is YOLOv4 by Bochkovskiy, Wang, and Liao (2020). In addition to outperforming other recent fast detectors such as EfficientDet (Tan, Pang, and Le 2020), as shown in Figure 6.31b, the paper breaks the processing pipeline into several stages, including a *neck*, which performs the top-down feature enhancement found in the feature pyramid network. The paper also evaluates many different components, which they categorize into a "bag of freebies" that can be used during training and a "bag of specials" that can be used at detection time with minimal additional cost.

While most bounding box object detectors continue to evaluate their results on the COCO dataset (Lin, Maire *et al.* 2014),[21] newer datasets such as Open Images (Kuznetsova, Rom *et al.* 2020), and LVIS: Large Vocabulary Instance Segmentation (Gupta, Dollár, and Girshick 2019) are now also being used (see Table 6.2). Two recent workshops that highlight the latest results using these datasets are Zendel *et al.* (2020) and Kirillov, Lin *et al.* (2020) and also have challenges related to instance segmentation, panoptic segmentation, keypoint estimation, and dense pose estimation, which are topics we discuss later in this chapter. Open-source frameworks for training and fine-tuning object detectors include the TensorFlow Object Detection API[22] and PyTorch's Detectron2.[23]

6.4 Semantic segmentation

A challenging version of general object recognition and scene understanding is to simultaneously perform recognition and accurate boundary segmentation (Fergus 2007). In this section, we examine a number of related problems, namely *semantic segmentation* (per-pixel class labeling), *instance segmentation* (accurately delineating each separate object), *panoptic segmentation* (labeling both objects and stuff), and dense pose estimation (labeling pixels belonging to people and their body parts). Figures 6.32 and 6.43 show some of these kinds of segmentations.

The basic approach to simultaneous recognition and segmentation is to formulate the problem as one of labeling every pixel in an image with its class membership. Older approaches often did this using energy minimization or Bayesian inference techniques, i.e., conditional random fields (Section 4.3.1). The TextonBoost system of Shotton, Winn *et al.* (2009) uses unary (pixel-wise) potentials based on image-specific color distributions (Section 4.3.2), location information (e.g.,

[21] See https://codalab.org for the latest competitions and leaderboards.
[22] https://github.com/tensorflow/models/tree/master/research/object_detection
[23] https://github.com/facebookresearch/detectron2

Name/URL	Extents	Contents/Reference
Object recognition		
Oxford buildings dataset	Pictures of buildings	5,062 images
https://www.robots.ox.ac.uk/~vgg/data/oxbuildings		Philbin, Chum *et al.* (2007)
INRIA Holidays	Holiday scenes	1,491 images
https://lear.inrialpes.fr/people/jegou/data.php		Jégou, Douze, and Schmid (2008)
PASCAL	Segmentations, boxes	11k images (2.9k with segmentations)
http://host.robots.ox.ac.uk/pascal/VOC		Everingham, Eslami *et al.* (2015)
ImageNet	Complete images	21k (WordNet) classes, 14M images
https://www.image-net.org		Deng, Dong *et al.* (2009)
Fashion MNIST	Complete images	70k fashion products
https://github.com/zalandoresearch/fashion-mnist		Xiao, Rasul, and Vollgraf (2017)
Object detection and segmentation		
Caltech Pedestrian Dataset	Bounding boxes	Pedestrians
http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians		Dollár, Wojek *et al.* (2009)
MSR Cambridge	Per-pixel segmentations	23 classes
https://www.microsoft.com/en-us/research/project/image-understanding		Shotton, Winn *et al.* (2009)
LabelMe dataset	Polygonal boundaries	>500 categories
http://labelme.csail.mit.edu		Russell, Torralba *et al.* (2008)
Microsoft COCO	Segmentations, boxes	330k images
https://cocodataset.org		Lin, Maire *et al.* (2014)
Cityscapes	Polygonal boundaries	30 classes, 25,000 images
https://www.cityscapes-dataset.com		Cordts, Omran *et al.* (2016)
Broden	Segmentation masks	A variety of visual concepts
http://netdissect.csail.mit.edu		Bau, Zhou *et al.* (2017)
Broden+	Segmentation masks	A variety of visual concepts
https://github.com/CSAILVision/unifiedparsing		Xiao, Liu *et al.* (2018)
LVIS	Instance segmentations	1,000 categories, 2.2M images
https://www.lvisdataset.org		Gupta, Dollár, and Girshick (2019)
Open Images	Segs., relationships	478k images, 3M relationships
https://g.co/dataset/openimages		Kuznetsova, Rom *et al.* (2020)

Table 6.2 Image databases for classification, detection, and localization.

foreground objects are more likely to be in the middle of the image, sky is likely to be higher, and road is likely to be lower), and novel texture-layout classifiers trained using shared boosting. It also uses traditional pairwise potentials that look at image color gradients. The texton-layout features first filter the image with a series of 17 oriented filter banks and then cluster the responses to classify each pixel into 30 different texton classes (Malik, Belongie *et al.* 2001). The responses are then filtered using offset rectangular regions trained with joint boosting (Viola and Jones 2004) to produce the texton-layout features used as unary potentials. Figure 6.33 shows some examples of images successfully labeled and segmented using TextonBoost

The TextonBoost conditional random field framework has been extended to LayoutCRFs by Winn and Shotton (2006), who incorporate additional constraints to recognize multiple object instances and deal with occlusions, and by Hoiem, Rother, and Winn (2007) to incorporate full 3D models. Conditional random fields continued to be widely used and extended for simultaneous recognition and segmentation applications, as described in the first edition of this book (Szeliski

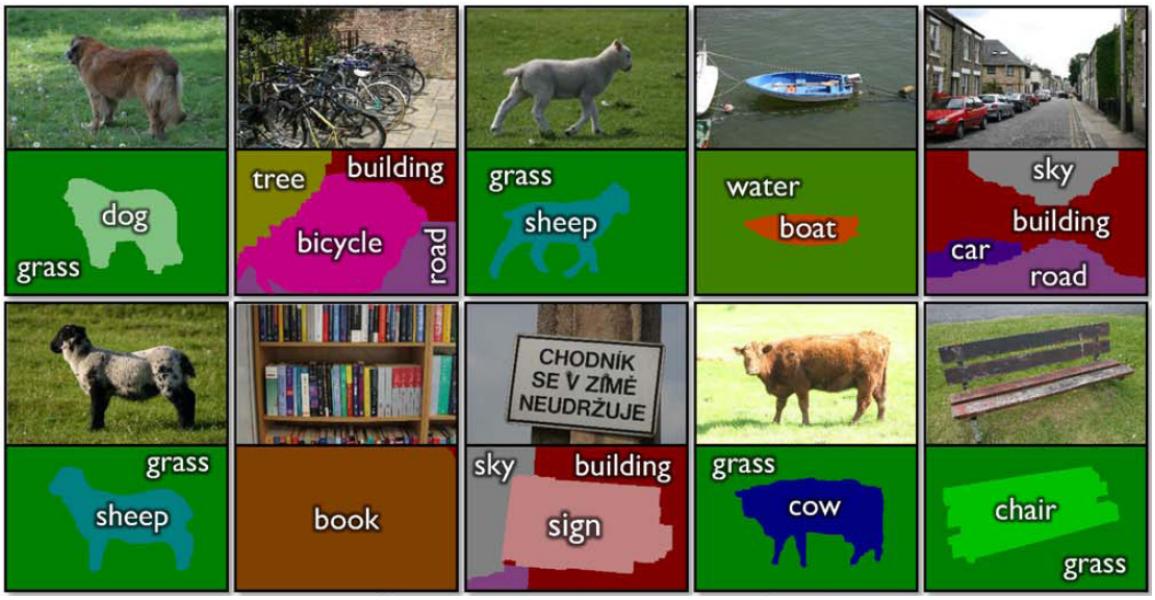


Figure 6.33 Simultaneous recognition and segmentation using TextonBoost (Shotton, Winn *et al.* 2009) © 2009 Springer.

2010, Section 14.4.3), along with approaches that first performed low-level or hierarchical segmentations (Section 7.5).

The development of fully convolutional networks (Long, Shelhamer, and Darrell 2015), which we described in Section 5.4.1, enabled per-pixel semantic labeling using a single neural network. While the first networks suffered from poor resolution (very loose boundaries), the addition of conditional random fields at a final stage (Chen, Papandreou *et al.* 2018; Zheng, Jayasumana *et al.* 2015), deconvolutional upsampling (Noh, Hong, and Han 2015), and fine-level connections in U-nets (Ronneberger, Fischer, and Brox 2015), all helped improve accuracy and resolution.

Modern semantic segmentation systems are often built on architectures such as the feature pyramid network (Lin, Dollár *et al.* 2017), which have top-down connections to help percolate semantic information down to higher-resolution maps. For example, the Pyramid Scene Parsing Network (PSPNet) of Zhao, Shi *et al.* (2017) uses spatial pyramid pooling (He, Zhang *et al.* 2015) to aggregate features at various resolution levels. The Unified Perceptual Parsing network (UPerNet) of Xiao, Liu *et al.* (2018) uses both a feature pyramid network and a pyramid pooling module to label image pixels not only with object categories but also materials, parts, and textures, as shown in Figure 6.34. HRNet (Wang, Sun *et al.* 2020) keeps high-resolution versions of feature maps throughout the pipeline with occasional interchange of information between channels at different resolution layers. Such networks can also be used to estimate surface normals and depths in an image (Huang, Zhou *et al.* 2019; Wang, Geraghty *et al.* 2020).

Semantic segmentation algorithms were initially trained and tested on datasets such as MSRC (Shotton, Winn *et al.* 2009) and PASCAL VOC (Everingham, Eslami *et al.* 2015). More recent datasets include the Cityscapes dataset for urban scene understanding (Cordts, Omran *et al.* 2016) and ADE20K (Zhou, Zhao *et al.* 2019), which labels pixels in a wider variety of indoor and outdoor scenes with 150 different category and part labels. The Broadly and Densely Labeled Dataset (Broden) created by Bau, Zhou *et al.* (2017) federates a number of such densely labeled datasets, including ADE20K, Pascal-Context, Pascal-Part, OpenSurfaces, and Describable Textures to obtain

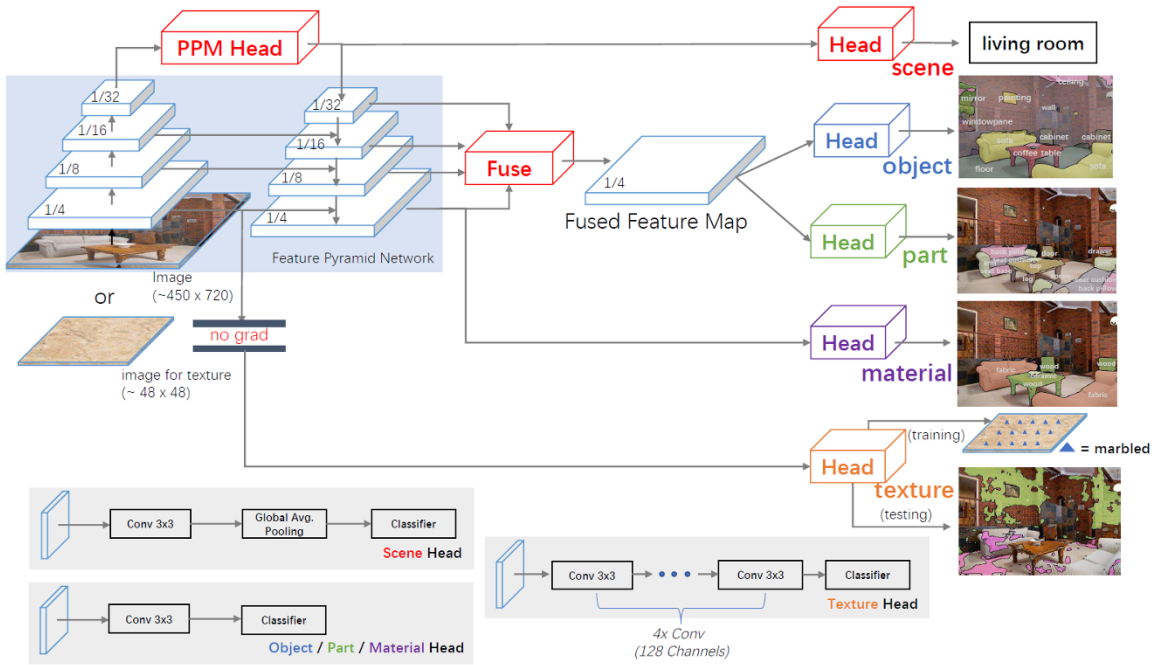


Figure 6.34 The UPerNet framework for Unified Perceptual Parsing (Xiao, Liu *et al.* 2018) © 2018 Springer. A Feature Pyramid Network (FPN) backbone is appended with a Pyramid Pooling Module (PPM) before feeding it into the top-down branch of the FPN. before feeding it into the top-down branch of the FPN. Various layers of the FPN and/or PPM are fed into different heads, including a scene head for image classification, object and part heads from the fused FPN features, a material head operating on the finest level of the FPN, and a texture head that does not participate in the FPN fine tuning. The bottom gray squares give more details into some of the heads.

a wide range of labels such as materials and textures in addition to basic object semantics. While this dataset was originally developed to aid in the interpretability of deep networks, it has also proven useful (with extensions) for training unified multi-task labeling systems such as UPerNet (Xiao, Liu *et al.* 2018). Table 6.2 list some of the datasets used for training and testing semantic segmentation algorithms.

One final note. While semantic image segmentation and labeling have widespread applications in image understanding, the converse problem of going from a semantic sketch or painting of a scene to a photorealistic image has also received widespread attention (Johnson, Gupta, and Fei-Fei 2018; Park, Liu *et al.* 2019; Bau, Strobelt *et al.* 2019; Ntavelis, Romero *et al.* 2020b). We look at this topic in more detail in Section 10.5.3 on semantic image synthesis.

6.4.1 *Application*: Medical image segmentation

One of the most promising applications of image segmentation is in the medical imaging domain, where it can be used to segment anatomical tissues for later quantitative analysis. Figure 4.21 shows a binary graph cut with directed edges being used to segment the liver tissue (light gray) from its surrounding bone (white) and muscle (dark gray) tissue. Figure 6.35 shows the segmentation of a brain scan for the detection of brain tumors. Before the development of the mature optimization and deep learning techniques used in modern image segmentation algorithms, such processing required much more laborious manual tracing of individual X-ray slices.

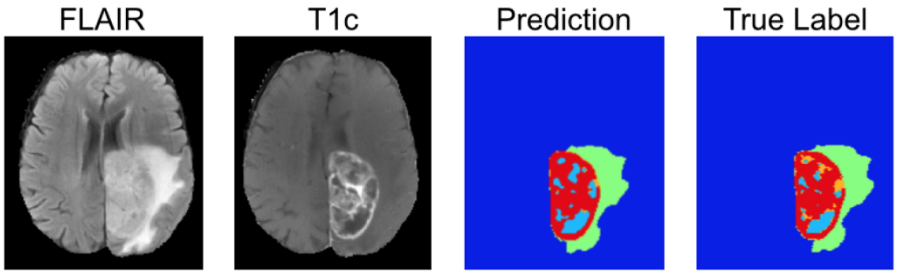


Figure 6.35 3D volumetric medical image segmentation using a deep network (Kamnitsas, Ferrante *et al.* 2016) © 2016 Springer.

Initially, optimization techniques such as Markov random fields (Section 4.3.2) and discriminative classifiers such as random forests (Section 5.1.5) were used for medical image segmentation (Criminisi, Robertson *et al.* 2013). More recently, the field has shifted to deep learning approaches (Kamnitsas, Ferrante *et al.* 2016; Kamnitsas, Ledig *et al.* 2017; Havaei, Davy *et al.* 2017).

The fields of medical image segmentation (McInerney and Terzopoulos 1996) and medical image registration (Kybic and Unser 2003) (Section 9.2.3) are rich research fields with their own specialized conferences, such as *Medical Imaging Computing and Computer Assisted Intervention (MICCAI)*, and journals, such as *Medical Image Analysis* and *IEEE Transactions on Medical Imaging*. These can be great sources of references and ideas for research in this area.

6.4.2 Instance segmentation

Instance segmentation is the task of finding all of the relevant objects in an image and producing pixel-accurate masks for their visible regions (Figure 6.36b). One potential approach to this task is to perform known object instance recognition (Section 6.1) and to then backproject the object model into the scene (Lowe 2004), as shown in Figure 6.1d, or matching portions of the new scene to pre-learned (segmented) object models (Ferrari, Tuytelaars, and Van Gool 2006b; Kannala, Rahtu *et al.* 2008). However, this approach only works for known rigid 3D models.

For more complex (flexible) object models, such as those for humans, a different approach is to pre-segment the image into larger or smaller pieces (Section 7.5) and to then match such pieces to portions of the model (Mori, Ren *et al.* 2004; Mori 2005; He, Zemel, and Ray 2006; Gu, Lim *et al.* 2009). For general highly variable classes, a related approach is to vote for potential object locations and scales based on feature correspondences and to then infer the object extents (Leibe, Leonardis, and Schiele 2008).

With the advent of deep learning, researchers started combining region proposals or image pre-segmentations with convolutional second stages to infer the final instance segmentations (Hariharan, Arbeláez *et al.* 2014; Hariharan, Arbeláez *et al.* 2015; Dai, He, and Sun 2015; Pinheiro, Lin *et al.* 2016; Dai, He, and Sun 2016; Li, Qi *et al.* 2017).

A breakthrough in instance segmentation came with the introduction of Mask R-CNN (He, Gkioxari *et al.* 2017). As shown in Figure 6.36a, Mask R-CNN uses the same region proposal network as Faster R-CNN (Ren, He *et al.* 2015), but then adds an additional *branch* for predicting the object *mask*, in addition to the existing branch for bounding box refinement and classification.[24] As with other networks that have multiple branches (or heads) and outputs, the training losses corre-

[24] Mask R-CNN was the first paper to introduce the terms *backbone* and *head* to describe the common deep convolutional feature extraction front end and the specialized back end branches.

(a) (b)

Figure 6.36 Instance segmentation using Mask R-CNN (He, Gkioxari *et al.* 2017) © 2017 IEEE: (a) system architecture, with an additional segmentation branch; (b) sample results.

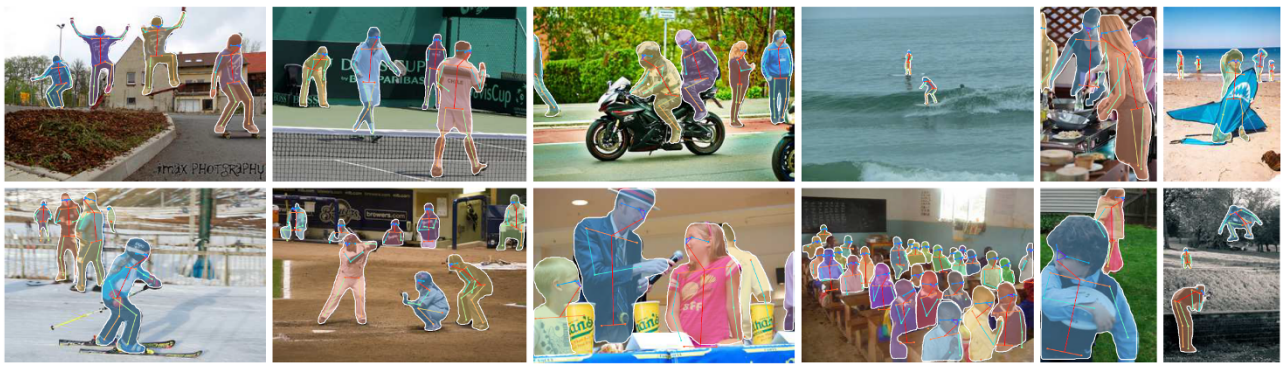


Figure 6.37 Person keypoint detection and segmentation using Mask R-CNN (He, Gkioxari *et al.* 2017) © 2017 IEEE

sponding to each supervised output need to be carefully balanced. It is also possible to add additional branches, e.g., branches trained to detect human keypoint locations (implemented as per-keypoint mask images), as shown in Figure 6.37.

Since its introduction, the performance of Mask R-CNN and its extensions has continued to improve with advances in backbone architectures (Liu, Qi *et al.* 2018; Chen, Pang *et al.* 2019). Two recent workshops that highlight the latest results in this area are the COCO + LVIS Joint Recognition Challenge (Kirillov, Lin *et al.* 2020) and the Robust Vision Challenge (Zendel *et al.* 2020).[25] It is also possible to replace the pixel masks produced by most instance segmentation techniques with time-evolving closed contours, i.e., "snakes" (Section 7.3.1), as in Peng, Jiang *et al.* (2020). In order to encourage higher-quality segmentation boundaries, Cheng, Girshick *et al.* (2021) propose a new Boundary Intersection-over-Union (Boundary IoU) metric to replace the commonly used Mask IoU metric.

6.4.3 Panoptic segmentation

As we have seen, semantic segmentation classifies each pixel in an image into its semantic category, i.e., what *stuff* does each pixel correspond to. Instance segmentation associates pixels with individ-

[25] You can find the leaderboards for instance segmentation and other COCO recognition tasks at https://cocodataset.org.

Figure 6.38 Panoptic segmentation results produced using a Panoptic Feature Pyramid Network (Kirillov, Girshick *et al.* 2019) © 2019 IEEE.

Figure 6.39 Detectron2 panoptic segmentation results on some of my personal photos. (Click on the "Colab Notebook" link at https://github.com/facebookresearch/detectron2 and then edit the input image URL to try your own.)

ual objects, i.e., how many *objects* are there and what are their extents (Figure 6.32). Putting both of these systems together has long been a goal of semantic scene understanding (Yao, Fidler, and Urtasun 2012; Tighe and Lazebnik 2013; Tu, Chen *et al.* 2005). Doing this on a per-pixel level results in a *panoptic segmentation* of the scene, where all of the objects are correctly segmented and the remaining stuff is correctly labeled (Kirillov, He *et al.* 2019). Producing a sensible *panoptic quality* (PQ) metric that simultaneously balances the accuracy on both tasks takes some careful design. In their paper, Kirillov, He *et al.* (2019) describe their proposed metric and analyze the performance of both humans (in terms of consistency) and recent algorithms on three different datasets.

The COCO dataset has now been extended to include a panoptic segmentation task, on which some recent results can be found in the ECCV 2020 workshop on this topic (Kirillov, Lin *et al.* 2020). Figure 6.38 show some segmentations produced by the panoptic feature pyramid network described by Kirillov, Girshick *et al.* (2019), which adds two branches for instance segmentation and semantic segmentation to a feature pyramid network.

(a) (b) (c) (d)

Figure 6.40 Scene completion using millions of photographs (Hays and Efros 2007) © 2007 ACM: (a) original image; (b) after unwanted foreground removal; (c) plausible scene matches, with the one the user selected highlighted in red; (d) output image after replacement and blending.

(a) (b) (c) (d) (e)

Figure 6.41 Automatic photo pop-up (Hoiem, Efros, and Hebert 2005a) © 2005 ACM: (a) input image; (b) superpixels are grouped into (c) multiple regions; (d) labels indicating ground (green), vertical (red), and sky (blue); (e) novel view of resulting piecewise-planar 3D model.

6.4.4 *Application*: Intelligent photo editing

Advances in object recognition and scene understanding have greatly increased the power of intelligent (semi-automated) photo editing applications. One example is the Photo Clip Art system of Lalonde, Hoiem *et al.* (2007), which recognizes and segments objects of interest, such as pedestrians, in internet photo collections and then allows users to paste them into their own photos. Another is the scene completion system of Hays and Efros (2007), which tackles the same *inpainting* problem we will study in Section 10.5. Given an image in which we wish to erase and fill in a large section (Figure 6.40a–b), where do you get the pixels to fill in the gaps in the edited image? Traditional approaches either use smooth continuation (Bertalmio, Sapiro *et al.* 2000) or borrow pixels from other parts of the image (Efros and Leung 1999; Criminisi, Pérez, and Toyama 2004; Efros and Freeman 2001). With the availability of huge numbers of images on the web, it often makes more sense to find a *different* image to serve as the source of the missing pixels.

In their system, Hays and Efros (2007) compute the *gist* of each image (Oliva and Torralba 2001; Torralba, Murphy *et al.* 2003) to find images with similar colors and composition. They then run a graph cut algorithm that minimizes image gradient differences and composite the new replacement piece into the original image using Poisson image blending (Section 8.4.4) (Pérez, Gangnet, and Blake 2003). Figure 6.40d shows the resulting image with the erased foreground rooftops region replaced with sailboats. Additional examples of photo editing and computational photography applications enabled by what has been dubbed "internet computer vision" can be found in the special journal issue edited by Avidan, Baker, and Shan (2010).

A different application of image recognition and segmentation is to infer 3D structure from a single photo by recognizing certain scene structures. For example, Criminisi, Reid, and Zisserman

Figure 6.42 OpenPose real-time multi-person 2D pose estimation (Cao, Simon *et al.* 2017) © 2017 IEEE.

(2000) detect vanishing points and have the user draw basic structures, such as walls, to infer the 3D geometry (Section 11.1.2). Hoiem, Efros, and Hebert (2005a), on the other hand, work with more "organic" scenes such as the one shown in Figure 6.41. Their system uses a variety of classifiers and statistics learned from labeled images to classify each pixel as either ground, vertical, or sky (Figure 6.41d). To do this, they begin by computing superpixels (Figure 6.41b) and then group them into plausible regions that are likely to share similar geometric labels (Figure 6.41c). After all the pixels have been labeled, the boundaries between the vertical and ground pixels can be used to infer 3D lines along which the image can be folded into a "pop-up" (after removing the sky pixels), as shown in Figure 6.41e. In related work, Saxena, Sun, and Ng (2009) develop a system that directly infers the depth and orientation of each pixel instead of using just three geometric class labels. We will examine techniques to infer depth from single images in more detail in Section 12.8.

6.4.5 Pose estimation

The inference of human pose (head, body, and limb locations and attitude) from a single images can be viewed as yet another kind of segmentation task. We have already discussed some pose estimation techniques in Section 6.3.2 on pedestrian detection section, as shown in Figure 6.25. Starting with the seminal work by Felzenszwalb and Huttenlocher (2005), 2D and 3D pose detection and estimation rapidly developed as an active research area, with important advances and datasets (Sigal and Black 2006a; Rogez, Rihan *et al.* 2008; Andriluka, Roth, and Schiele 2009; Bourdev and Malik 2009; Johnson and Everingham 2011; Yang and Ramanan 2011; Pishchulin, Andriluka *et al.* 2013; Sapp and Taskar 2013; Andriluka, Pishchulin *et al.* 2014).

More recently, deep networks have become the preferred technique to identify human body keypoints in order to convert these into pose estimates (Tompson, Jain *et al.* 2014; Toshev and

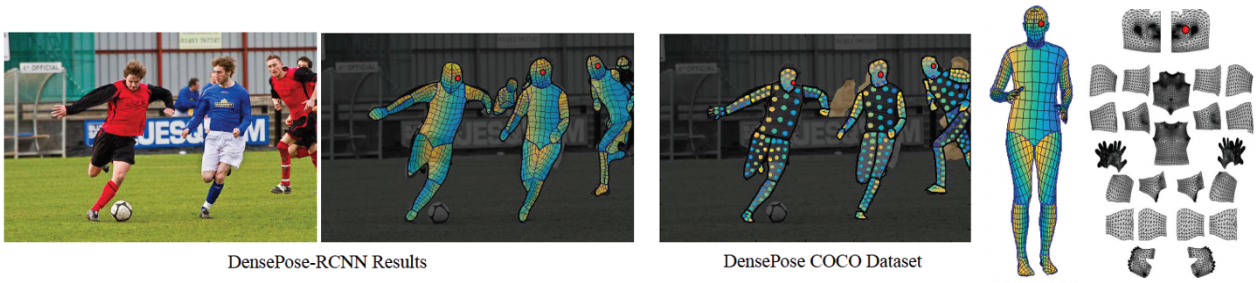


Figure 6.43 Dense pose estimation aims at mapping all human pixels of an RGB image to the 3D surface of the human body (Güler, Neverova, and Kokkinos 2018) © 2018 IEEE. The paper describes DensePose-COCO, a large-scale ground-truth dataset containing manually annotated image-to-surface correspondences for 50K persons and a DensePose-RCNN trained to densely regress UV coordinates at multiple frames per second.

Szegedy 2014; Pishchulin, Insafutdinov *et al.* 2016; Wei, Ramakrishna *et al.* 2016; Cao, Simon *et al.* 2017; He, Gkioxari *et al.* 2017; Hidalgo, Raaj *et al.* 2019; Huang, Zhu *et al.* 2020).[26] Figure 6.42 shows some of the impressive real-time multi-person 2D pose estimation results produced by the OpenPose system (Cao, Hidalgo *et al.* 2019).

The latest, most challenging, task in human pose estimation is the DensePose task introduced by Güler, Neverova, and Kokkinos (2018), where the task is to associate each pixel in RGB images of people with 3D points on a surface-based model, as shown in Figure 6.43. The authors provide dense annotations for 50,000 people appearing in COCO images and evaluate a number of correspondence networks, including their own DensePose-RCNN with several extensions. A more in-depth discussion on 3D human body modeling and tracking can be found in Section 13.6.4.

6.5 Video understanding

As we've seen in the previous sections of this chapter, image understanding mostly concerns itself with naming and delineating the objects and stuff in an image, although the relationships between objects and people are also sometimes inferred (Yao and Fei-Fei 2012; Gupta and Malik 2015; Yatskar, Zettlemoyer, and Farhadi 2016; Gkioxari, Girshick *et al.* 2018). (We will look at the topic of describing complete images in the next section on vision and language.)

What, then, is video understanding? For many researchers, it starts with the detection and description of human actions, which are taken as the basic atomic units of videos. Of course, just as with images, these basic primitives can be chained into more complete descriptions of longer video sequences.

Human activity recognition began being studied in the 1990s, along with related topics such as human motion tracking, which we discuss in Sections 9.4.4 and 13.6.4. Aggarwal and Cai (1999) provide a comprehensive review of these two areas, which they call *human motion analysis*. Some of the techniques they survey use point and mesh tracking, as well as spatio-temporal signatures.

In the 2000s, attention shifted to spatio-temporal features, such as the clever use of optical flow in small patches to recognize sports activities (Efros, Berg *et al.* 2003) or spatio-temporal feature detectors for classifying actions in movies (Laptev, Marszalek *et al.* 2008), later combined with image context (Marszalek, Laptev, and Schmid 2009) and tracked feature trajectories (Wang and Schmid 2013). Poppe (2010), Aggarwal and Ryoo (2011), and Weinland, Ronfard, and Boyer (2011)

[26] You can find the leaderboards for human keypoint detection at https://cocodataset.org.

Name/URL	Metadata	Contents/Reference
Charades	Actions, objects, descriptions	9.8k videos
https://prior.allenai.org/projects/charades		Sigurdsson, Varol *et al.* (2016)
YouTube8M	Entities	4.8k visual entities, 8M videos
https://research.google.com/youtube8m		Abu-El-Haija, Kothari *et al.* (2016)
Kinetics	Action classes	700 action classes, 650k videos
https://deepmind.com/research/open-source/kinetics		Carreira and Zisserman (2017)
"Something-something"	Actions with objects	174 actions, 220k videos
https://20bn.com/datasets/something-something		Goyal, Kahou *et al.* (2017)
AVA	Actions	80 actions in 430 15-minute videos
https://research.google.com/ava		Gu, Sun *et al.* (2018)
EPIC-KITCHENS	Actions and objects	100 hours of egocentric videos
https://epic-kitchens.github.io		Damen, Doughty *et al.* (2018)

Table 6.3 Datasets for video understanding and action recognition.

provide surveys of algorithms from this decade. Some of the datasets used in this research include the KTH human motion dataset (Schüldt, Laptev, and Caputo 2004), the UCF sports action dataset (Rodriguez, Ahmed, and Shah 2008), the Hollywood human action dataset (Marszalek, Laptev, and Schmid 2009), UCF-101 (Soomro, Zamir, and Shah 2012), and the HMDB human motion database (Kuehne, Jhuang *et al.* 2011).

In the last decade, video understanding techniques have shifted to using deep networks (Ji, Xu *et al.* 2013; Karpathy, Toderici *et al.* 2014; Simonyan and Zisserman 2014a; Tran, Bourdev *et al.* 2015; Feichtenhofer, Pinz, and Zisserman 2016; Carreira and Zisserman 2017; Varol, Laptev, and Schmid 2017; Wang, Xiong *et al.* 2019; Zhu, Li *et al.* 2020), sometimes combined with temporal models such as LSTMs (Baccouche, Mamalet *et al.* 2011; Donahue, Hendricks *et al.* 2015; Ng, Hausknecht *et al.* 2015; Srivastava, Mansimov, and Salakhudinov 2015).

While it is possible to apply these networks directly to the pixels in the video stream, e.g., using 3D convolutions (Section 5.5.1), researchers have also investigated using optical flow (Chapter 9.3) as an additional input. The resulting *two-stream architecture* was proposed by Simonyan and Zisserman (2014a) and is shown in Figure 6.44a. A later paper by Carreira and Zisserman (2017) compares this architecture to alternatives such as 3D convolutions on the pixel stream as well as hybrids of two streams and 3D convolutions (Figure 6.44b).

The latest architectures for video understanding have gone back to using 3D convolutions on the raw pixel stream (Tran, Wang *et al.* 2018, 2019; Kumawat, Verma *et al.* 2021). Wu, Feichtenhofer *et al.* (2019) store 3D CNN features into what they call a *long-term feature bank* to give a broader temporal context for action recognition. Feichtenhofer, Fan *et al.* (2019) propose a two-stream SlowFast architecture, where a slow pathway operates at a lower frame rate and is combined with features from a fast pathway with higher temporal sampling but fewer channels (Figure 6.44c). Some widely used datasets used for evaluating these algorithms are summarized in Table 6.3. They include Charades (Sigurdsson, Varol *et al.* 2016), YouTube8M (Abu-El-Haija, Kothari *et al.* 2016), Kinetics (Carreira and Zisserman 2017), "Something-something" (Goyal, Kahou *et al.* 2017), AVA (Gu, Sun *et al.* 2018), EPIC-KITCHENS (Damen, Doughty *et al.* 2018), and AVA-Kinetics (Li, Thotakuri *et al.* 2020). A nice exposition of these and other video understanding algorithms can be found in Johnson (2020, Lecture 18).

As with image recognition, researchers have also started using self-supervised algorithms to train video understanding systems. Unlike images, video clips are usually *multi-modal*, i.e., they contain

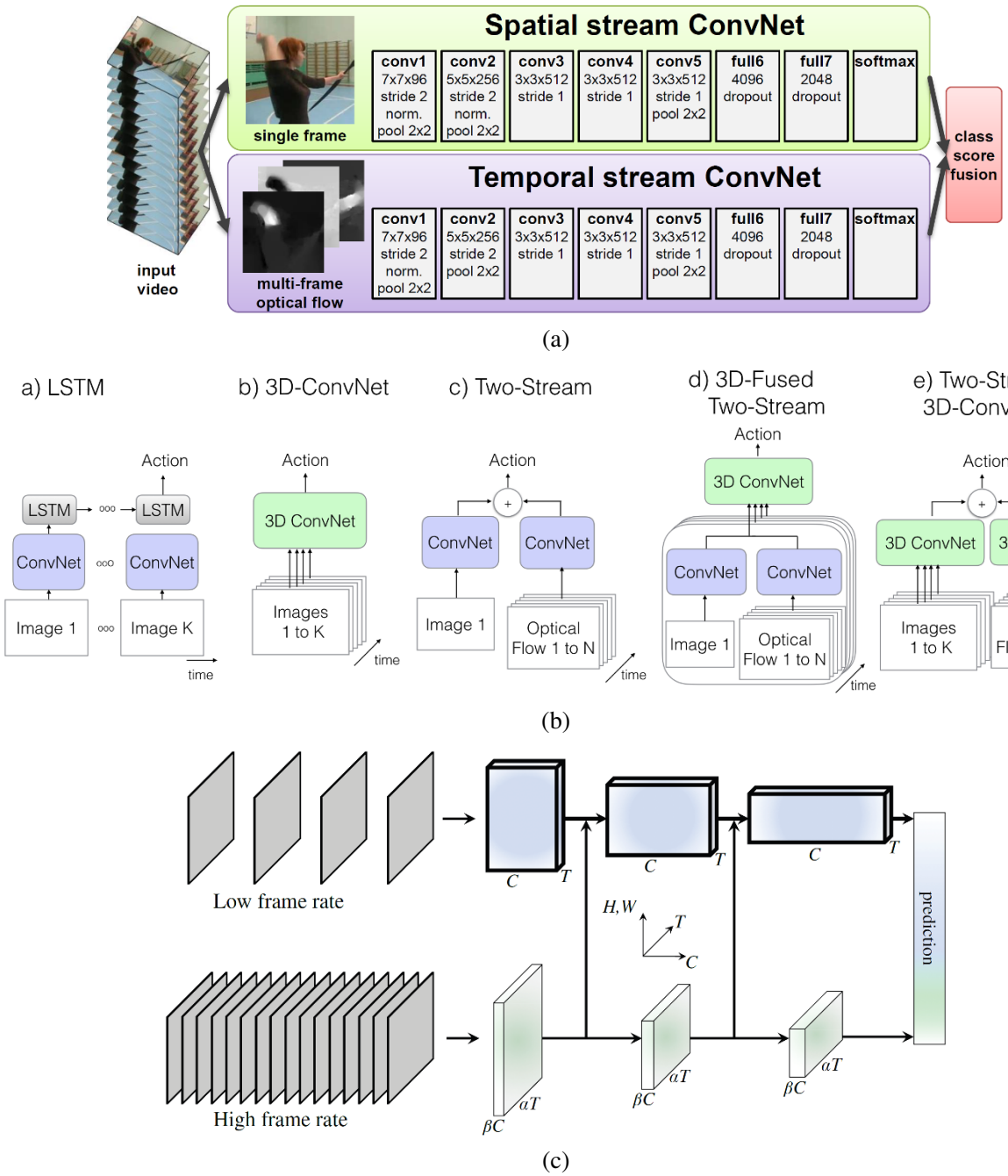


Figure 6.44 Video understanding using neural networks: (a) two-stream architecture for video classification © Simonyan and Zisserman (2014a); (b) some alternative video processing architectures (Carreira and Zisserman 2017) © 2017 IEEE; (c) a SlowFast network with a low frame rate, low temporal resolution Slow pathway and a high frame rate, higher temporal resolution Fast pathway (Feichtenhofer, Fan *et al.* 2019) © 2019 IEEE.

audio tracks in addition to the pixels, which can be an excellent source of unlabeled supervisory signals (Alwassel, Mahajan *et al.* 2020; Patrick, Asano *et al.* 2020). When available at inference time, audio signals can improve the accuracy of such systems (Xiao, Lee *et al.* 2020).

Finally, while action recognition is the main focus of most recent video understanding work, it is also possible to classify videos into different scene categories such as "beach", "fireworks", or "snowing." This problem is called *dynamic scene recognition* and can be addressed using spatio-temporal CNNs (Feichtenhofer, Pinz, and Wildes 2017).

6.6 Vision and language

The ultimate goal of much of computer vision research is not just to solve simpler tasks such as building 3D models of the world or finding relevant images, but to become an essential component of *artificial general intelligence* (AGI). This requires vision to integrate with other components of artificial intelligence such as speech and language understanding and synthesis, logical inference, and commonsense and specialized knowledge representation and reasoning.

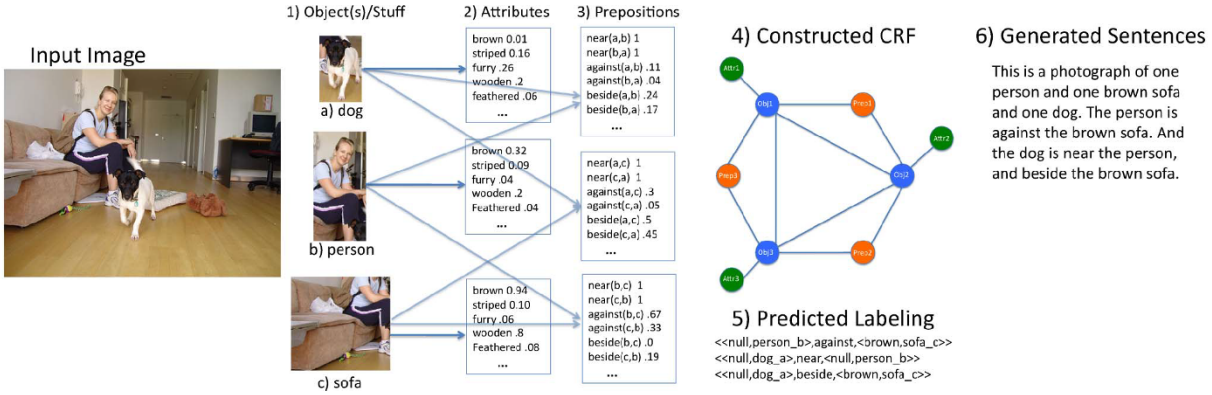
Advances in speech and language processing have enabled the widespread deployment of speech-based intelligent virtual assistants such as Siri, Google Assistant, and Alexa. Earlier in this chapter, we've seen how computer vision systems can name individual objects in images and find similar images by appearance or keywords. The next natural step of integration with other AI components is to merge vision and language, i.e., *natural language processing* (NLP).

While this area has been studied for a long time (Duygulu, Barnard *et al.* 2002; Farhadi, Hejrati *et al.* 2010), the last decade has seen a rapid increase in performance and capabilities (Mogadala, Kalimuthu, and Klakow 2021; Gan, Yu *et al.* 2020). An example of this is the BabyTalk system developed by Kulkarni, Premraj *et al.* (2013), which first detects objects, their attributes, and their positional relationships, then infers a likely compatible labeling of these objects, and finally generates an image caption, as shown in Figure 6.45a.

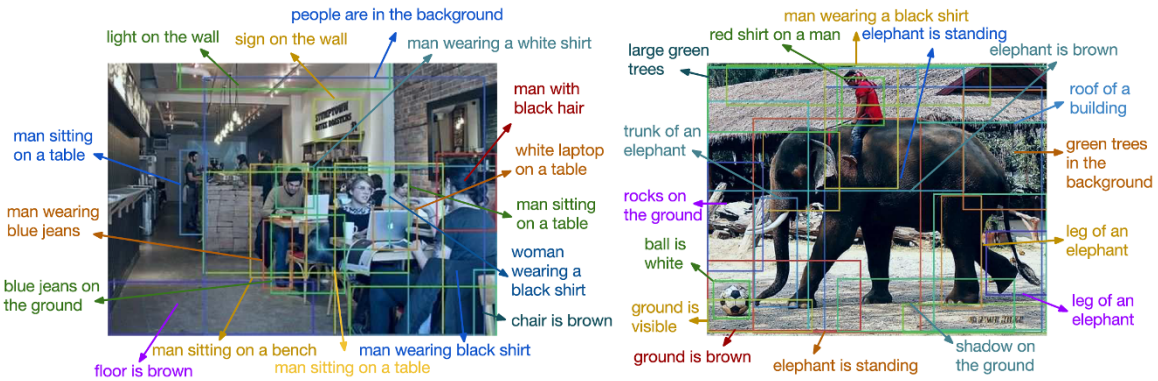
Visual captioning

The next few years brought a veritable explosion of papers on the topic of image captioning and description, including (Chen and Lawrence Zitnick 2015; Donahue, Hendricks *et al.* 2015; Fang, Gupta *et al.* 2015; Karpathy and Fei-Fei 2015; Vinyals, Toshev *et al.* 2015; Xu, Ba *et al.* 2015; Johnson, Karpathy, and Fei-Fei 2016; Yang, He *et al.* 2016; You, Jin *et al.* 2016). Many of these systems combine CNN-based image understanding components (mostly object and human action detectors) with RNNs or LSTMs to generate the description, often in conjunction with other techniques such as multiple instance learning, maximum entropy language models, and visual attention. One somewhat surprising early result was that nearest-neighbor techniques, i.e., finding sets of similar looking images with captions and then creating a consensus caption, work surprisingly well (Devlin, Gupta *et al.* 2015).

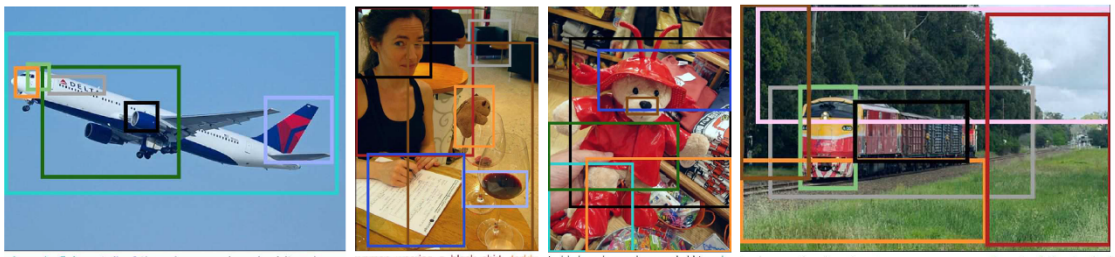
Over the last few years, attention-based systems have continued to be essential components of image captioning systems (Lu, Xiong *et al.* 2017; Anderson, He *et al.* 2018; Lu, Yang *et al.* 2018). Figure 6.46 shows examples from two such papers, where each word in the generated caption is *grounded* with a corresponding image region. The CVPR 2020 tutorial by (Zhou 2020) summarizes over two dozen related papers from the last five years, including papers that use transformers (Section 5.5.3) to do the captioning. It also covers video description and dense video captioning (Aafaq, Mian *et al.* 2019; Zhou, Kalantidis *et al.* 2019) and vision-language pre-training (Sun, Myers *et al.* 2019; Zhou, Palangi *et al.* 2020; Li, Yin *et al.* 2020). The tutorial also has lectures on



(a)



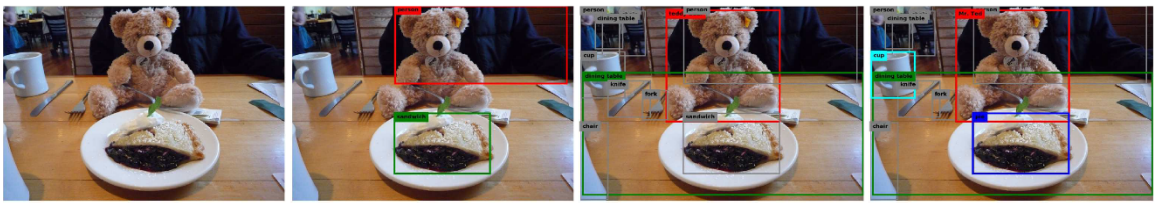
(b)



(c)

Figure 6.45 Image captioning systems: (a) BabyTalk detects objects, attributes, and positional relationships and composes these into image captions (Kulkarni, Premraj *et al.* 2013) © 2013 IEEE; (b–c) DenseCap associates word phrases with regions and then uses an RNN to construct plausible sentences (Johnson, Karpathy, and Fei-Fei 2016) © 2016 IEEE.

(a)



(b)

Figure 6.46 Image captioning with attention: (a) The "Show, Attend, and Tell" system, which uses hard attention to align generated words with image regions © Xu, Ba *et al.* (2015); (b) Neural Baby Talk captions generated using different detectors, showing the association between words and grounding regions (Lu, Yang *et al.* 2018) © 2018 IEEE.

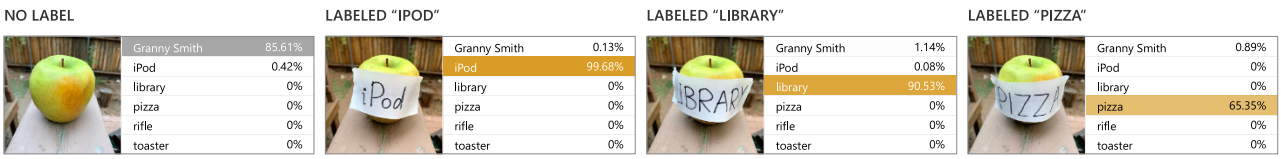


Figure 6.47 An adversarial typographic attack used against CLIP (Radford, Kim *et al.* 2021) discovered by ©Goh, Cammarata *et al.* (2021). Instead of predicting the object that exists in the scene, CLIP predicts the output based on the adversarial handwritten label.

visual question answering and reasoning (Gan 2020), text-to-image synthesis (Cheng 2020), and vision-language pre-training (Yu, Chen, and Li 2020).

For the task of image classification (Section 6.2), one of the major restrictions is that a model can only predict a label from the discrete pre-defined set of labels it trained on. CLIP (Radford, Kim *et al.* 2021) proposes an alternative approach that relies on image captions to enable zero-shot transfer to any possible set of labels. Given an image with a set of labels (e.g., $\{\text{dog}, \text{cat}, \ldots, \text{house}\}$), CLIP predicts the label that maximizes the probability that the image is captioned with a prompt similar to "A photo of a {label}". Section 5.4.7 discusses the training aspect of CLIP, which collects 400 million text-image pairs and uses contrastive learning to determine how likely it is for an image to be paired with a caption.

Remarkably, without having seen or fine-tuned to many popular image classification benchmarks (e.g., ImageNet, Caltech 101), CLIP can outperform independently fine-tuned ResNet-50 models supervised on each specific dataset. Moreover, compared to state-of-the-art classification models, CLIP's zero-shot generalization is significantly more robust to dataset distribution shifts, performing well on each of ImageNet Sketch (Wang, Ge *et al.* 2019), ImageNetV2 (Recht, Roelofs *et al.* 2019), and ImageNet-R (Hendrycks, Basart *et al.* 2020), without being specifically trained on any of them.

Name/URL	Metadata	Contents/Reference
Flickr30k (Entities)	Image captions (grounded)	30k images (+ bounding boxes)
https://shannon.cs.illinois.edu/DenotationGraph		Young, Lai *et al.* (2014)
http://bryanplummer.com/Flickr30kEntities		Plummer, Wang *et al.* (2017)
COCO Captions	Whole image captions	1.5M captions, 330k images
https://cocodataset.org/#captions-2015		Chen, Fang *et al.* (2015)
Conceptual Captions	Whole image captions	3.3M image caption pairs
https://ai.google.com/research/ConceptualCaptions		Sharma, Ding *et al.* (2018)
YFCC100M	Flickr metadata	100M images with metadata
http://projects.dfki.uni-kl.de/yfcc100m		Thomee, Shamma *et al.* (2016)
Visual Genome	Dense annotations	108k images with region graphs
https://visualgenome.org		Krishna, Zhu *et al.* (2017)
VQA v2.0	Question/answer pairs	265k images
https://visualqa.org		Goyal, Khot *et al.* (2017)
VCR	Multiple choice questions	110k movie clips, 290k QAs
https://visualcommonsense.com		Zellers, Bisk *et al.* (2019)
GQA	Compositional QA	22M questions on Visual Genome
https://visualreasoning.net		Hudson and Manning (2019)
VisDial	Dialogs for chatbot	120k COCO images + dialogs
https://visualdialog.org		Das, Kottur *et al.* (2017)

Table 6.4 Image datasets for vision and language research.

In fact, Goh, Cammarata *et al.* (2021) found that CLIP units responded similarly with concepts presented in different modalities (e.g., an image of Spiderman, text of the word spider, and a drawing of Spiderman). Figure 6.47 shows the adversarial typographic attack they discovered that could fool CLIP. By simply placing a handwritten class label (e.g., iPod) on a real-world object (e.g., Apple), CLIP often predicted the class written on the label.

As with other areas of visual recognition and learning-based systems, datasets have played an important role in the development of vision and language systems. Some widely used datasets of images with captions include Conceptual Captions (Sharma, Ding *et al.* 2018), the UIUC Pascal Sentence Dataset (Farhadi, Hejrati *et al.* 2010), the SBU Captioned Photo Dataset (Ordonez, Kulkarni, and Berg 2011), Flickr30k (Young, Lai *et al.* 2014), COCO Captions (Chen, Fang *et al.* 2015), and their extensions to 50 sentences per image (Vedantam, Lawrence Zitnick, and Parikh 2015) (see Table 6.4). More densely annotated datasets such as Visual Genome (Krishna, Zhu *et al.* 2017) describe different sub-regions of an image with their own phrases, i.e., provide *dense captioning*, as shown in Figure 6.48. YFCC100M (Thomee, Shamma *et al.* 2016) contains around 100M images from Flickr, but it only includes the raw user uploaded metadata for each image, such as the title, time of upload, description, tags, and (optionally) the location of the image.

Metrics for measuring sentence similarity also play an important role in the development of image captioning and other vision and language systems. Some widely used metrics include BLEU: BiLingual Evaluation Understudy (Papineni, Roukos *et al.* 2002), ROUGE: Recall Oriented Understudy of Gisting Evaluation (Lin 2004), METEOR: Metric for Evaluation of Translation with Explicit ORdering (Banerjee and Lavie 2005), CIDEr: Consensus-based Image Description Evaluation (Vedantam, Lawrence Zitnick, and Parikh 2015), and SPICE: Semantic Propositional Image Caption Evaluation (Anderson, Fernando *et al.* 2016).[27]

[27] See https://www.cs.toronto.edu/~fidler/slides/2017/CSC2539/Kaustav_slides.pdf.

Figure 6.48 Images and data from the Visual Genome dataset (Krishna, Zhu *et al.* 2017) © 2017 Springer. (a) An example image with its region descriptors. (b) Each region has a graph representation of objects, attributes, and pairwise relationships, which are combined into a scene graph where all the objects are grounded to the image, and also associated questions and answers. (c) Some sample question and answer pairs, which cover a spectrum of visual tasks from recognition to high-level reasoning.

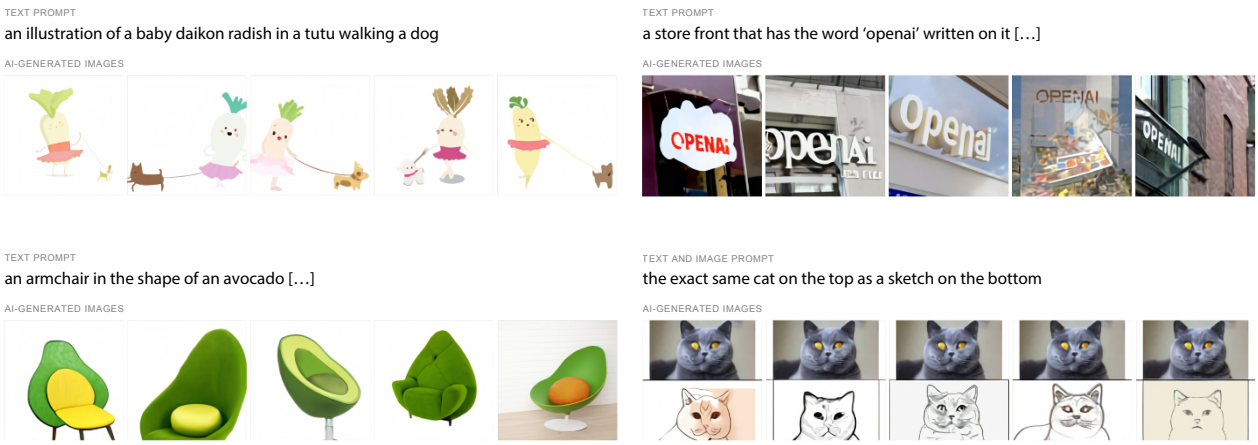


Figure 6.49 Qualitative text-to-image generation results from DALL·E, showing a wide range of generalization abilities ©Ramesh, Pavlov *et al.* (2021). The bottom right example provides a partially complete image prompt of a cat, along with text, and has the model fill in the rest of the image. The other three examples only start with the text prompt as input, with the model generating the entire image.

Text-to-image generation

The task of text-to-image generation is the inverse of visual captioning, i.e., given a text prompt, generate the image. Since images are represented in such high dimensionality, generating them to look coherent has historically been difficult. Generating images from a text prompt can be thought of as a generalization of generating images from a small set of class labels (Section 5.5.4). Since there is a near-infinite number of possible text prompts, successful models must be able to generalize from the relatively small fraction seen during training.

Early work on this task from Mansimov, Parisotto *et al.* (2016) used an RNN to iteratively draw an image from scratch. Their results showed some resemblance to the text prompts, although the generated images were quite blurred. The following year, Reed, Akata *et al.* (2016) applied a GAN to the problem, where unseen text prompts began to show promising results. Their generated images were relatively small (64×64), which was improved in later papers, which often first generated a small-scale image and then conditioned on that image and the text input to generate a higher-resolution image (Zhang, Xu *et al.* 2017, 2018; Xu, Zhang *et al.* 2018; Li, Qi *et al.* 2019).

DALL·E (Ramesh, Pavlov *et al.* 2021) uses orders of magnitude of more data (250 million text-image pairs on the internet) and compute to achieve astonishing qualitative results (Figure 6.49).[28] Their approach produces promising results for generalizing beyond training data, even compositionally piecing together objects that are not often related (e.g., an armchair and an avocado), producing many styles (e.g., painting, cartoon, charcoal drawings), and working reasonably well with difficult objects (e.g., mirrors or text).

The model for DALL·E consists of two components: a VQ-VAE-2 (Section 5.5.4) and a decoder transformer (Section 5.5.3). The text is tokenized into 256 tokens, each of which is one of 16,384 possible vectors using a BPE-encoding (Sennrich, Haddow, and Birch 2015). The VQ-VAE-2 uses a codebook of size 8,192 (significantly larger than the codebook of size 512 used in the original VQ-VAE-2 paper) to compress images as a 32×32 grid of vector tokens. At inference time, DALL·E uses a transformer decoder, which starts with the 256 text tokens to autoregressively predict the 32×32 grid of image tokens. Given such a grid, the VQ-VAE-2 is able to use its decoder to generate

[28] Play with the results at https://openai.com/blog/dall-e.

the final RGB image of size 256 × 256. To achieve better empirical results, DALL·E generates 512 image candidates and reranks them using CLIP (Radford, Kim *et al.* 2021), which determines how likely a given caption is associated with a given image.

An intriguing extension of DALL·E is to use the VQ-VAE-2 encoder to predict a subset of the compressed image tokens. For instance, suppose we are given a text input and an image. The text input can be tokenized into its 256 tokens, and one can obtain the 32 × 32 image tokens using the VQ-VAE-2 encoder. If we then discard the bottom half of the image tokens, the transformer decoder can be used to autoregressively predict which tokens might be there. These tokens, along with the non-discarded ones from the original image, can be passed into the VQ-VAE-2 decoder to produce a completed image. Figure 6.49 (bottom right) shows how such a text and partial image prompt can be used for applications such as image-to-image translation (Section 5.5.4).

Visual Question Answering and Reasoning

Image and video captioning are useful tasks that bring us closer to building artificially intelligent systems, as they demonstrate the ability to put together visual cues such as object identities, attributes, and actions. However, it remains unclear if the system has understood the scene at a deeper level and if it can reason about the constituent pieces and how they fit together.

To address these concerns, researchers have been building *visual question answering* (VQA) systems, which require the vision algorithm to answer open-ended questions about the image, such as the ones shown in Figure 6.48c. A lot of this work started with the creation of the Visual Question Answering (VQA) dataset (Antol, Agrawal *et al.* 2015), which spurred a large amount of subsequent research. The following year, VQA v2.0 improved this dataset by creating a *balanced* set of image pairs, where each question had different answers in the two images (Goyal, Khot *et al.* 2017).[29] This dataset was further extended to reduce the influence of prior assumptions and data distributions and to encourage answers to be grounded in the images (Agrawal, Batra *et al.* 2018).

Since then, many additional VQA datasets have been created. These include the VCR dataset for visual commonsense reasoning (Zellers, Bisk *et al.* 2019) and the GQA dataset and metrics for evaluating visual reasoning and compositional question answering (Hudson and Manning 2019), which is built on top of the information about objects, attributes, and relations provided through the Visual Genome scene graphs (Krishna, Zhu *et al.* 2017). A discussion of these and other datasets for VQA can be found in the CVPR 2020 tutorial by Gan (2020), including datasets that test visual grounding and referring expression comprehension, visual entailment, using external knowledge, reading text, answering sub-questions, and using logic. Some of these datasets are summarized in Table 6.4.

As with image and video captioning, VQA systems use various flavors of attention to associate pixel regions with semantic concepts (Yang, He *et al.* 2016). However, instead of using sequence models such as RNNs, LSTMs, or transformers to generate text, the natural language question is first parsed to produce an encoding that is then fused with the image embedding to generate the desired answer.

The image semantic features can either be computed on a coarse grid, or a "bottom-up" object detector can be combined with a "top-down" attention mechanism to provide feature weightings (Anderson, He *et al.* 2018). In recent years, the pendulum has swung back and forth between techniques that use bottom-up regions and gridded feature descriptors, with two of the recent best-performing algorithms going back to the simpler (and much faster) gridded approach (Jiang, Misra *et al.* 2020; Huang, Zeng *et al.* 2020). The CVPR 2020 tutorial by Gan (2020) discusses these and

[29] https://visualqa.org

dozens of other VQA systems as well as their subcomponents, such as multimodal fusion variants (bilinear pooling, alignment, relational reasoning), neural module networks, robust VQA, and multimodal pre-training, The survey by Mogadala, Kalimuthu, and Klakow (2021) and the annual VQA Challeng workshop (Shrivastava, Hudson *et al.* 2020) are also excellent sources of additional information. And if you would like to test out the current state of VQA systems, you can upload your own image to https://vqa.cloudcv.org and ask the system your own questions.

Visual Dialog. An even more challenging version of VQA is *visual dialog*, where a chatbot is given an image and asked to answer open-ended questions about the image while also referring to previous elements of the conversation. The VisDial dataset was the earliest to be widely used for this task (Das, Kottur *et al.* 2017).[30] You can find pointers to systems that have been developed for this task at the Visual Dialog workshop and challenge (Shrivastava, Hudson *et al.* 2020). There's also a chatbot at https://visualchatbot.cloudcv.org where you can upload your own image and start a conversation, which can sometimes lead to humorous (or weird) outcomes (Shane 2019).

Vision-language pre-training. As with many other recognition tasks, pre-training has had some dramatic success in the last few years, with systems such as ViLBERT (Lu, Batra *et al.* 2019), Oscar (Li, Yin *et al.* 2020), and many other systems described in the CVPR 2020 tutorial on self-supervised learning for vision-and-language (Yu, Chen, and Li 2020).

6.7 Additional reading

Unlike machine learning or deep learning, there are no recent textbooks or surveys devoted specifically to the general topics of image recognition and scene understanding. Some earlier surveys (Pinz 2005; Andreopoulos and Tsotsos 2013) and collections of papers (Ponce, Hebert *et al.* 2006; Dickinson, Leonardis *et al.* 2007) review the "classic" (pre-deep learning) approaches, but given the tremendous changes in the last decade, many of these techniques are no longer used. Currently, some of the best sources for the latest material, in addition to this chapter and university computer vision courses, are tutorials at the major vision conferences such as ICCV (Xie, Girshick *et al.* 2019), CVPR (Girshick, Kirillov *et al.* 2020), and ECCV (Xie, Girshick *et al.* 2020). Image recognition datasets such as those listed in Tables 6.1–6.4 that maintain active leaderboards can also be a good source for recent papers.

Algorithms for instance recognition, i.e., the detection of static manufactured objects that only vary slightly in appearance but may vary in 3D pose, are still often based on detecting 2D points of interest and describing them using viewpoint-invariant descriptors, as discussed in Chapter 7 and (Lowe 2004), Rothganger, Lazebnik *et al.* (2006), and Gordon and Lowe (2006). In more recent years, attention has shifted to the more challenging problem of *instance retrieval* (also known as *content-based image retrieval*), in which the number of images being searched can be very large (Sivic and Zisserman 2009). Section 7.1.4 in the next chapter reviews such techniques, as does the survey in (Zheng, Yang, and Tian 2018). This topic is also related to visual similarity search (Bell and Bala 2015; Arandjelovic, Gronat *et al.* 2016; Song, Xiang *et al.* 2016; Gordo, Almazán *et al.* 2017; Rawat and Wang 2017; Bell, Liu *et al.* 2020), which was covered in Section 6.2.3.

A number of surveys, collections of papers, and course notes have been written on the topic of feature-based whole image (single-object) category recognition (Pinz 2005; Ponce, Hebert *et al.* 2006; Dickinson, Leonardis *et al.* 2007; Fei-Fei, Fergus, and Torralba 2009). Some of these papers

[30] https://visualdialog.org

use a bag of words or keypoints (Csurka, Dance *et al.* 2004; Lazebnik, Schmid, and Ponce 2006; Csurka, Dance *et al.* 2006; Grauman and Darrell 2007b; Zhang, Marszalek *et al.* 2007; Boiman, Shechtman, and Irani 2008; Ferencz, Learned-Miller, and Malik 2008). Other papers recognize objects based on their contours, e.g., using shape contexts (Belongie, Malik, and Puzicha 2002) or other techniques (Shotton, Blake, and Cipolla 2005; Opelt, Pinz, and Zisserman 2006; Ferrari, Tuytelaars, and Van Gool 2006a).

Many object recognition algorithms use part-based decompositions to provide greater invariance to articulation and pose. Early algorithms focused on the relative positions of the parts (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991) while later algorithms used more sophisticated models of appearance (Felzenszwalb and Huttenlocher 2005; Fergus, Perona, and Zisserman 2007; Felzenszwalb, McAllester, and Ramanan 2008). Good overviews on part-based models for recognition can be found in the course notes by Fergus (2009). Carneiro and Lowe (2006) discuss a number of graphical models used for part-based recognition, which include trees and stars, k-fans, and constellations.

Classical recognition algorithms often used scene context as part of their recognition strategy. Representative papers in this area include Torralba (2003), Torralba, Murphy *et al.* (2003), Rabinovich, Vedaldi *et al.* (2007), Russell, Torralba *et al.* (2007), Sudderth, Torralba *et al.* (2008), and Divvala, Hoiem *et al.* (2009). Machine learning also became a key component of classical object detection and recognition algorithms (Felzenszwalb, McAllester, and Ramanan 2008; Sivic, Russell *et al.* 2008), as did exploiting large human-labeled databases (Russell, Torralba *et al.* 2007; Torralba, Freeman, and Fergus 2008).

The breakthrough success of the "AlexNet" SuperVision system of Krizhevsky, Sutskever, and Hinton (2012) shifted the focus in category recognition research from feature-based approaches to deep neural networks. The rapid improvement in recognition accuracy, captured in Figure 5.40 and described in more detail in Section 5.4.3 has been driven to a large degree by deeper networks and better training algorithms, and also in part by larger (unlabeled) training datasets (Section 5.4.7).

More specialized recognition systems such as those for recognizing faces underwent a similar evolution. While some of the earliest approaches to face recognition involved finding the distinctive image features and measuring the distances between them (Fischler and Elschlager 1973; Kanade 1977; Yuille 1991), later approaches relied on comparing gray-level images, often projected onto lower dimensional subspaces (Turk and Pentland 1991; Belhumeur, Hespanha, and Kriegman 1997; Heisele, Ho *et al.* 2003) or local binary patterns (Ahonen, Hadid, and Pietikäinen 2006). A variety of shape and pose deformation models were also developed (Beymer 1996; Vetter and Poggio 1997), including Active Shape Models (Cootes, Cooper *et al.* 1995), 3D Morphable Models (Blanz and Vetter 1999; Egger, Smith *et al.* 2020), and Active Appearance Models (Cootes, Edwards, and Taylor 2001; Matthews and Baker 2004; Ramnath, Koterba *et al.* 2008). Additional information about classic face recognition algorithms can be found in a number of surveys and books on this topic (Chellappa, Wilson, and Sirohey 1995; Zhao, Chellappa *et al.* 2003; Li and Jain 2005).

The concept of shape models for *frontalization* continued to be used as the community shifted to deep neural network approaches (Taigman, Yang *et al.* 2014). Some more recent deep face recognizers, however, omit the frontalization stage and instead use data augmentation to create synthetic inputs with a larger variety of poses (Schroff, Kalenichenko, and Philbin 2015; Parkhi, Vedaldi, and Zisserman 2015). Masi, Wu *et al.* (2018) provide an excellent tutorial and survey on deep face recognition, including a list of widely used training and testing datasets, a discussion of frontalization and dataset augmentation, and a section on training losses.

As the problem of whole-image (single object) category recognition became more "solved", attention shifted to multiple object delineation and labeling, i.e., object detection. Object detection

was originally studied in the context of specific categories such as faces, pedestrians, cars, etc. Seminal papers in face detection include those by Osuna, Freund, and Girosi (1997); Sung and Poggio (1998); Rowley, Baluja, and Kanade (1998); Viola and Jones (2004); Heisele, Ho *et al.* (2003), with Yang, Kriegman, and Ahuja (2002) providing a comprehensive survey of early work in this field. Early work in pedestrian and car detection was carried out by Gavrila and Philomin (1999); Gavrila (1999); Papageorgiou and Poggio (2000); Schneiderman and Kanade (2004). Subsequent papers include (Mikolajczyk, Schmid, and Zisserman 2004; Dalal and Triggs 2005; Leibe, Seemann, and Schiele 2005; Andriluka, Roth, and Schiele 2009; Dollár, Belongie, and Perona 2010; Felzenszwalb, Girshick *et al.* 2010).

Modern generic object detectors are typically constructed using a region proposal algorithm (Uijlings, Van De Sande *et al.* 2013; Zitnick and Dollár 2014) that then feeds selected regions of the image (either as pixels or precomputed neural features) into a multi-way classifier, resulting in architectures such as R-CNN (Girshick, Donahue *et al.* 2014), Fast R-CNN (Girshick 2015), Faster R-CCNN (Ren, He *et al.* 2015), and FPN (Lin, Dollár *et al.* 2017). An alternative to this two-stage approach is a *single-stage network*, which uses a single network to output detections at a variety of locations. Examples of such architectures include SSD (Liu, Anguelov *et al.* 2016), RetinaNet (Lin, Goyal *et al.* 2017), and YOLO (Redmon, Divvala *et al.* 2016; Redmon and Farhadi 2017, 2018; Bochkovskiy, Wang, and Liao 2020). These and more recent convolutional object detectors are described in the recent survey by Jiao, Zhang *et al.* (2019).

While object detection can be sufficient in many computer vision applications such as counting cars or pedestrians or even describing images, a detailed pixel-accurate labeling can be potentially even more useful, e.g., for photo editing. This kind of labeling comes in several flavors, including semantic segmentation (what stuff is this?), instance segmentation (which countable object is this?), panoptic segmentation (what stuff or object is it?). One early approach to this problem was to pre-segment the image into pieces and then match these pieces to portions of the model (Mori, Ren *et al.* 2004; Russell, Efros *et al.* 2006; Borenstein and Ullman 2008; Gu, Lim *et al.* 2009). Another popular approach was to use conditional random fields (Kumar and Hebert 2006; He, Zemel, and Carreira-Perpiñán 2004; Winn and Shotton 2006; Rabinovich, Vedaldi *et al.* 2007; Shotton, Winn *et al.* 2009). which at that time produced some of the best results on the PASCAL VOC segmentation challenge. Modern semantic segmentation algorithms use pyramidal fully-convolutional architectures to map input pixels to class labels (Long, Shelhamer, and Darrell 2015; Zhao, Shi *et al.* 2017; Xiao, Liu *et al.* 2018; Wang, Sun *et al.* 2020).

The more challenging task of instance segmentation, where each distinct object gets its own unique label, is usually tackled using a combination of object detectors and per-object segmentation, as exemplified in the seminal Mask R-CNN paper by He, Gkioxari *et al.* (2017). Follow-on work uses more sophisticated backbone architectures (Liu, Qi *et al.* 2018; Chen, Pang *et al.* 2019). Two recent workshops that highlight the latest results in this area are the COCO + LVIS Joint Recognition Challenge (Kirillov, Lin *et al.* 2020) and the Robust Vision Challenge (Zendel *et al.* 2020).

Putting semantic and instance segmentation together has long been a goal of semantic scene understanding (Yao, Fidler, and Urtasun 2012; Tighe and Lazebnik 2013; Tu, Chen *et al.* 2005). Doing this on a per-pixel level results in a *panoptic segmentation* of the scene, where all of the objects are correctly segmented and the remaining stuff is correctly labeled (Kirillov, He *et al.* 2019; Kirillov, Girshick *et al.* 2019). The COCO dataset has now been extended to include a panoptic segmentation task, on which some recent results can be found in the ECCV 2020 workshop on this topic (Kirillov, Lin *et al.* 2020).

Research in video understanding, or more specifically human activity recognition, dates back to the 1990s; some good surveys include (Aggarwal and Cai 1999; Poppe 2010; Aggarwal and Ryoo

2011; Weinland, Ronfard, and Boyer 2011). In the last decade, video understanding techniques shifted to using deep networks (Ji, Xu *et al.* 2013; Karpathy, Toderici *et al.* 2014; Simonyan and Zisserman 2014a; Donahue, Hendricks *et al.* 2015; Tran, Bourdev *et al.* 2015; Feichtenhofer, Pinz, and Zisserman 2016; Carreira and Zisserman 2017; Tran, Wang *et al.* 2019; Wu, Feichtenhofer *et al.* 2019; Feichtenhofer, Fan *et al.* 2019). Some widely used datasets used for evaluating these algorithms are summarized in Table 6.3.

While associating words with images has been studied for a while (Duygulu, Barnard *et al.* 2002), sustained research into describing images with captions and complete sentences started in the early 2010s (Farhadi, Hejrati *et al.* 2010; Kulkarni, Premraj *et al.* 2013). The last decade has seen a rapid increase in performance and capabilities of such systems (Mogadala, Kalimuthu, and Klakow 2021; Gan, Yu *et al.* 2020). The first sub-problem to be widely studied was image captioning (Donahue, Hendricks *et al.* 2015; Fang, Gupta *et al.* 2015; Karpathy and Fei-Fei 2015; Vinyals, Toshev *et al.* 2015; Xu, Ba *et al.* 2015; Devlin, Gupta *et al.* 2015), with later systems using attention mechanisms (Anderson, He *et al.* 2018; Lu, Yang *et al.* 2018). More recently, researchers have developed systems for visual question answering (Antol, Agrawal *et al.* 2015) and visual commonsense reasoning (Zellers, Bisk *et al.* 2019).

The CVPR 2020 tutorial on recent advances in visual captioning (Zhou 2020) summarizes over two dozen related papers from the last five years, including papers that use Transformers to do the captioning. It also covers video description and dense video captioning (Aafaq, Mian *et al.* 2019; Zhou, Kalantidis *et al.* 2019) and vision-language pre-training (Sun, Myers *et al.* 2019; Zhou, Palangi *et al.* 2020; Li, Yin *et al.* 2020). The tutorial also has lectures on visual question answering and reasoning (Gan 2020), text-to-image synthesis (Cheng 2020), and vision-language pre-training (Yu, Chen, and Li 2020).

6.8 Exercises

Ex 6.1: Pre-trained recognition networks. Find a pre-trained network for image classification, segmentation, or some other task such as face recognition or pedestrian detection.

After running the network, can you characterize the most common kinds of errors the network is making? Create a "confusion matrix" indicating which categories get classified as other categories. Now try the network on your own data, either from a web search or from your personal photo collection. Are there surprising results?

My own favorite code to try is Detectron2,[31] which I used to generate the panoptic segmentation results shown in Figure 6.39.

Ex 6.2: Re-training recognition networks. After analyzing the performance of your pre-trained network, try re-training it on the original dataset on which it was trained, but with modified parameters (numbers of layers, channels, training parameters) or with additional examples. Can you get the network to perform more to you liking?

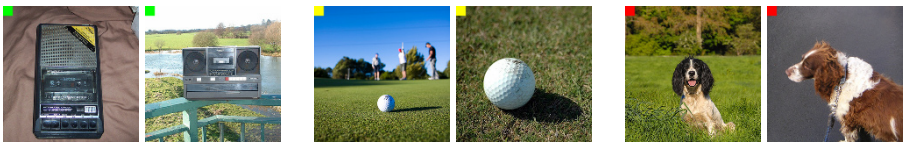
Many of the online tutorials, such as the Detectron2 Collab notebook mentioned above, come with instructions on how to re-train the network from scratch on a different dataset. Can you create your own dataset, e.g., using a web search and figure out how to label the examples? A low effort (but not very accurate) way is to trust the results of the web search. Russakovsky, Deng *et al.* (2015), Kovashka, Russakovsky *et al.* (2016), and other papers on image datasets discuss the challenges in obtaining accurate labels.

[31] Click on the "Colab Notebook" link at https://github.com/facebookresearch/detectron2 and then edit the input image URL to try your own.

Train your network, try to optimize its architecture, and report on the challenges you faced and discoveries you made.

Note: the following exercises were suggested by Matt Deitke.

Ex 6.3: Image perturbations. Download either ImageNet or Imagenette.[32] Now, perturb each image by adding a small square to the top left of the image, where the color of the square is unique for each label, as shown in the following figure:



(a) cassette player (b) golf ball (c) English Springer

Using any image classification model,[33] e.g., ResNet, EfficientNet, or ViT, train the model from scratch on the perturbed images. Does the model overfit to the color of the square and ignore the rest of the image? When evaluating the model on the training and validation data, try adversarially swapping colors between different labels.

Ex 6.4: Image normalization. Using the same dataset downloaded for the previous exercise, take a ViT model and remove all the intermediate layer normalization operations. Are you able to train the network? Using techniques in Li, Xu *et al.* (2018), how do the plots of the loss landscape appear with and without the intermediate layer normalization operations?

Ex 6.5: Semantic segmentation. Explain the differences between instance segmentation, semantic segmentation, and panoptic segmentation. For each type of segmentation, can it be post-processed to obtain the other kinds of segmentation?

Ex 6.6: Class encoding. Categorical inputs to a neural network, such as a word or object, can be encoded with one-hot encoded vector.[34] However, it is common to pass the one-hot encoded vector through an embedding matrix, where the output is then passed into the neural network loss function. What are the advantages of vector embedding over using one-hot encoding?

Ex 6.7: Object detection. For object detection, how do the number of parameters for DETR, Faster-RCNN, and YOLOv4 compare? Try training each of them on MS COCO. Which one tends to train the slowest? How long does it take each model to evaluate a single image at inference time?

Ex 6.8: Image classification vs. description. For image classification, list at least two significant differences between using categorical labels and natural language descriptions.

Ex 6.9: ImageNet Sketch. Try taking several pre-trained models on ImageNet and evaluating them, without any fine-tuning, on ImageNet Sketch (Wang, Ge *et al.* 2019). For each of these models, to what extent does the performance drop due to the shift in distribution?

Ex 6.10: Self-supervised learning. Provide examples of self-supervised learning pretext tasks for each of the following data types: static images, videos, and vision-and-language.

[32] Imagenette, https://github.com/fastai/imagenette, is a smaller 10-class subset of ImageNet that is easier to use with limited computing resources. .

[33] You may find the PyTorch Image Models at https://github.com/rwightman/pytorch-image-models useful.

[34] With a categorical variable, one-hot encoding is used to represent which label is chosen, i.e., when a label is chosen, its entry in the vector is 1 with all other entries being 0.

Ex 6.11: Video understanding. For many video understanding tasks, we may be interested in tracking an object through time. Why might this be preferred to making predictions independently for each frame? Assume that inference speed is not a problem.

Ex 6.12: Fine-tuning a new head. Take the backbone of a network trained for object classification and fine-tune it for object detection with a variant of YOLO. Why might it be desirable to freeze the early layers of the network?

Ex 6.13: Movie understanding. Currently, most video understanding networks, such as those discussed in this chapter, tend to only deal with short video clips as input. What modifications might be necessary in order to operate over longer sequences such as an entire movie?