COMP3204/COMP6223 Computer Vision

10

Image classification and auto-annotation

Summary

In the previous lectures we've looked at how features can be extracted from images, and briefly discussed how supervised machine learning techniques like linear classifiers and k-nearest-neighbours can be used to train a computer vision system to predict a *class* for a particular feature input. Current research is looking at how we might make computers able to *see* in the human sense, fully understanding the content of a visual scene. The choice of feature for image classification is very important. Last lecture we saw how a Bag of Visual Words (BoVW) representation was a powerful technique for image search. It turns out that BoVWs are very useful for use in high performance image classification.

Key points

Classification using image features

- Recap:
    - Typical computer vision system takes an image, passes it through a *feature extractor* and eventually feeds these features to a machine learning system to make decisions.
    - A supervised machine-learning algorithm uses a set of **pre-labelled training data** to learn how to assign class labels to vectors (and the corresponding objects).
    - A **binary classifier** only classifies into two classes.
    - A **multiclass classifier** can classify into one of many classes.
- A **multilabel** classifier can predict multiple labels or classes for a given input.
    - Often with probabilities/confidences.
    - In the context of images multilabel classification is often called *automatic image annotation* or *auto-annotation*.
    - Automatic image annotation does not try to determine where in an image a particular *thing* is – it just looks for presence of the *thing*.
    - Object recognition goes a step further and attempts to both localise an *object* and determine (recognise) its class.

Current research challenges

- Current research is looking at how to solve a number of challenges:
    - Unconstrained object recognition in natural scenes.
    - Global classification of images into scene categories/events/topics.
    - Automatic annotation of large sets of imagery.
- The fundamental problem of computer vision that researchers are trying to solve is to make computers able perceive images in the same way humans can.
    - We refer to human-level understanding as "semantic understanding".
        - The semantics encode the not only the objects within and image, but also the interplay of those objects and the overall *meaning* of the image.
    - The gap between what computer vision can currently achieve and what humans can perceive is known as the **Semantic Gap**.

A history of different approaches and the use of the BoVW

- Historically, bags of visual word (histograms of visual word occurrences) have been very important.

    - It turns out that the occurrence histograms created from the BoVW representation used in image search are also incredibly powerful descriptors for whole-image classification and object detection tasks.
    - Many approaches to auto-annotation treat the problem as one of **language translation** or as of mapping different languages to a common sub-space known as a **semantic space**.
    - The raw features are not just from local features like SIFT, but also from quantised descriptors of **segments** and even the **colour** of individual pixels.
    - Unlike the large-scale search scenario where the vocabularies/codebooks of visual words were very large, for classification/auto-annotation, the codebook vocabulary needs to be much **smaller**:
        - In general, machine-learning techniques need much smaller vectors (for both performance and effectiveness)
        - The visual words can be allowed to be **less distinctive**, allowing a little more variation between matching features.
        - Typically, the number of visual words might be as small as a **few hundred, and up to a few thousand**.
    - Although you can use interest points to select local regions of the image to describe, classification performance is often improved by **sampling the image at a greater rate**.
        - For example, **Dense SIFT**, extracts SIFT descriptors on a densely sampled grid across the image, rather than at interest points.
        - **Pyramid Dense SIFT** goes a step further and considers different sized sampling windows from a Gaussian Scale Space.
    - BoVW representations can be augmented with a ***spatial pyramid***, which builds sub-histograms of visual word occurrences for overlapping windows in the image.
        - This improves performance as it allows the machine learning system to learn *where* in an image objects/features are likely to appear.
        - Visual words from quantised Pyramid Dense SIFT, arranged in a spatial pyramid are know as a "**Pyramid Histogram of Words**" or **PHOW**.

Developing and benchmarking a BoVW scene classifier

- Common for academic research to use **standardised datasets** for developing scene classifiers and comparing results
    - Datasets are usually split into labelled "**training**" and "**test**" sets.
    - **Only the training set can be used to train the classifier**
    - Sometimes the test set labels are ***withheld*** completely to ensure there is no cheating!
- Firstly the raw features need to be extracted from the training images
    - Then (if necessary) learn a codebook from these features
        - I.e. using k-means on the raw features
            - You might use a *uniform random sample* of all the features rather than all of them for speed.
- Apply (vector) quantisation to the raw features and count the number of occurrences to build histograms of visual words for each image.
- Classifiers can be trained using the histograms.
    - E.g. OvR linear classifiers with a kernel map.
- You might train on a subset of the training data and use the remaining data to "**validate**" and **optimise** parameters.
    - Once you've chosen the optimal parameters you can then re-train using the optimal values with **all the training data**.
- You're now in a position to **apply the classifiers to the test data**:
    - Extract the features.
    - Quantise the features (using the codebook developed from the training set!).
    - Compute the occurrence histograms.
    - Use the classifiers to find the most likely class.
- Lots of ways to **evaluate performance** of classification on the test (and validation) set.

        - Conceptually the simplest summary measure is probably ***average precision***
            - This is literally the proportion of **number of correct classifications** to the **total number of predictions**.

Further reading

- Wikipedia has good articles on:
    - Vector quantisation: http://en.wikipedia.org/wiki/Vector_quantization
    - Bag of Visual Words (and applications): http://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision
- First work on spatial pyramids: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1641019&tag=1
- Info on the homogeneous kernel map (including software implementations and papers): http://www.robots.ox.ac.uk/~vgg/software/homkermap/

Practical exercises

- Chapter 12 of the OpenIMAJ tutorial covers dense local feature extraction, spatial pyramids and fast linear classification for learning a set of 101 object categories.