**Introduction to HOG and its features:**

# HOG (Histogram of Oriented Gradients):

The Histogram of Oriented Gradient (HOG) feature descriptor is popular for object detection.

## Algorithm overview:

Compute a Histogram of Oriented Gradients (HOG) by

* 1. (optional) global image normalization.
  2. computing the gradient image in x and y
  3. computing gradient histograms
  4. normalizing across blocks
  5. flattening into a feature vector

The first stage applies an optional global image normalization equalization that is designed to reduce the influence of illumination effects. In practice we use gamma (power law) compression, either computing the square root or the log of each color channel. Image texture strength is typically proportional to the local surface illumination so this compression helps to reduce the effects of local shadowing and illumination variations.

The second stage computes first order image gradients. These capture contour, silhouette and some texture information, while providing further resistance to illumination variations. The locally dominant color channel is used, which provides color invariance to a large extent. Variant methods may also include second order image derivatives, which act as primitive bar detectors - a useful feature for capturing, e.g. bar like structures in bicycles and limbs in humans.

The third stage aims to produce an encoding that is sensitive to local image content while remaining resistant to small changes in pose or appearance. The adopted method pools gradient orientation information locally in the same way as the SIFT [[2]](https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html#id4) feature. The image window is divided into small spatial regions, called “cells”. For each cell we accumulate a local 1-D histogram of gradient or edge orientations over all the pixels in the cell. This combined cell-level 1-D histogram forms the basic “orientation histogram” representation. Each orientation histogram divides the gradient angle range into a fixed number of predetermined bins. The gradient magnitudes of the pixels in the cell are used to vote into the orientation histogram.

The fourth stage computes normalization, which takes local groups of cells and contrast normalizes their overall responses before passing to next stage. Normalization introduces better invariance to illumination, shadowing, and edge contrast. It is performed by accumulating a measure of local histogram “energy” over local groups of cells that we call “blocks”. The result is used to normalize each cell in the block. Typically, each individual cell is shared between several blocks, but its normalizations are block dependent and thus different. The cell thus appears several times in the final output vector with different normalizations. This may seem redundant but it improves the performance. We refer to the normalized block descriptors as Histogram of Oriented Gradient (HOG) descriptors.

The final step collects the HOG descriptors from all blocks of a dense overlapping grid of blocks covering the detection window into a combined feature vector for use in the window classifiedd.

1. **Applied Image and output of HOG:**

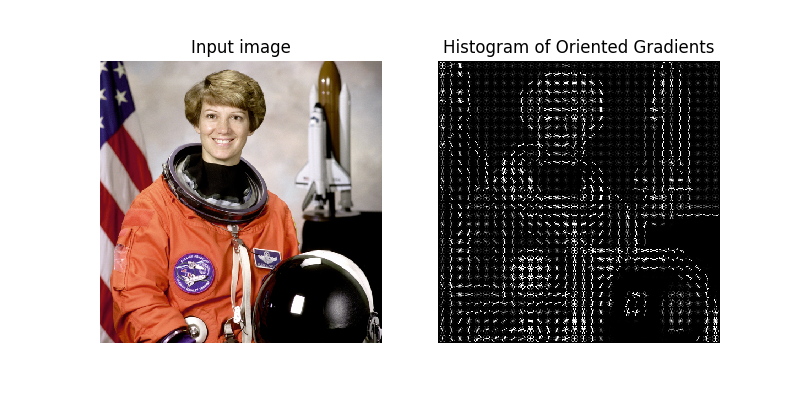


Figure 5.1 Input image and output HOG image

**MNIST dataset**

In order to understand the better MNIST dataset let us first understand what is dataset and it’s uses:

* A data set (or dataset) is a collection of [data](https://en.wikipedia.org/wiki/Data). Most commonly a data set corresponds to the contents of a single [database table](https://en.wikipedia.org/wiki/Table_(database)), or a single statistical [data matrix](https://en.wikipedia.org/wiki/Data_matrix_(multivariate_statistics)), where every [column](https://en.wikipedia.org/wiki/Column_(database)) of the table represents a particular variable, and each [row](https://en.wikipedia.org/wiki/Row_(database)) corresponds to a given member of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. The data set may comprise data for one or more members, corresponding to the number of rows.
* The term data set may also be used more loosely, to refer to the data in a collection of closely related tables, corresponding to a particular experiment or event. Less used names for this kind of data sets are **data corpus** and **data stock**. An example of this type is the data sets collected by space agencies performing experiments with instruments aboard [space probes](https://en.wikipedia.org/wiki/Space_probe). Data sets that are so large that traditional [data processing](https://en.wikipedia.org/wiki/Data_processing) applications are inadequate to deal with them are known as [big data](https://en.wikipedia.org/wiki/Big_data)
* In the [open data](https://en.wikipedia.org/wiki/Open_data) discipline, data set is the unit to measure the information released in a public open data repository. The European Open Data portal aggregates more than half a million data sets. In this field other definitions have been proposed but currently there is not an official one. Some other issues (real-time data sources, non-relational data sets, etc.) increases the difficulty to reach a consensus about it.

The **MNIST database** (Modified [National Institute of Standards and Technology](https://en.wikipedia.org/wiki/National_Institute_of_Standards_and_Technology) database) is a large [database](https://en.wikipedia.org/wiki/Database) of handwritten digits that is commonly used for [training](https://en.wikipedia.org/wiki/Training_set) various [image processing](https://en.wikipedia.org/wiki/Image_processing) systems. The database is also widely used for training and testing in the field of [machine learning](https://en.wikipedia.org/wiki/Machine_learning). It was created by "re-mixing" the samples from [NIST's original datasets](https://www.nist.gov/srd/upload/nistsd19.pdf). The creators felt that since NIST's training dataset was taken from American [Census Bureau](https://en.wikipedia.org/wiki/United_States_Census_Bureau) employees, while the testing dataset was taken from [American](https://en.wikipedia.org/wiki/Americans) [high school](https://en.wikipedia.org/wiki/High_school) students, it was not well-suited for machine learning experiments Furthermore, the black and white images from NIST were [normalized](https://en.wikipedia.org/wiki/Normalization_(image_processing)) to fit into a 28x28 pixel bounding box and [anti-aliased](https://en.wikipedia.org/wiki/Spatial_anti-aliasing), which introduced grayscale levels.

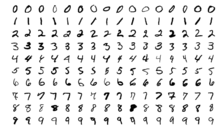
[](https://en.wikipedia.org/wiki/File:MnistExamples.png)

Figure 5.2.1 Sample images from MNIST test dataset.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. There have been a number of [scientific papers](https://en.wikipedia.org/wiki/Academic_publishing) on attempts to achieve the lowest error rate; one paper, using a hierarchical system of [convolutional neural networks](https://en.wikipedia.org/wiki/Convolutional_neural_network), manages to get an [error rate](https://en.wikipedia.org/wiki/Computer_performance) on the MNIST database of 0.23%.The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a [support vector machine](https://en.wikipedia.org/wiki/Support_vector_machine) to get an error rate of 0.8% . An extended dataset similar to MNIST called EMNIST has been published in 2017, which contains 240,000 training images, and 40,000 testing images of handwritten digits and characters

**Performance:**

Some researchers have achieved "near-human performance" on the MNIST database, using a committee of neural networks; in the same paper, the authors achieve performance double that of humans on other recognition tasks. The highest error rate listed on the original website of the database is 12 percent, which is achieved using a simple linear classifier with no pre-processing.

In 2004, a best-case error rate of 0.42 percent was achieved on the database by researchers using a new classifier called the LIRA, which is a neural classifier with three neuron layers based on Rosenblatt's perceptron principles.

Some researchers have tested artificial intelligence systems using the database put under random distortions. The systems in these cases are usually neural networks and the distortions used tend to be either [affine distortions](https://en.wikipedia.org/wiki/Affine_transformation) or [elastic distortions](https://en.wikipedia.org/wiki/Elastic_deformation). Sometimes, these systems can be very successful; one such system achieved an error rate on the database of 0.39 percent.

In 2011, an error rate of 0.27 percent, improving on the previous best result, was reported by researchers using a similar system of neural networks in 2013, an approach based on regularization of neural networks using Drop Connect has been claimed to achieve a 0.21 percent error rate. Recently, the single convolutional neural network best performance was 0.31 percent error rate. As of August 2018, the best performance of a single convolutional neural network trained on MNIST training data using Realtime data augmentation is 0.26 percent error rate. Also, the Parallel Computing Center (Khmelnitskiy, Ukraine) obtained an ensemble of only 5 convolutional neural networks which performs on MNIST at 0.21 percent error rate. Incorrect labelling of the testing dataset may prevent reaching test error rates of 0%.

**Support Vector Machine Algorithm**

**Introduction to SVM**

In machine learning, **support-vector machines** (**SVMs**, also **support-vector networks**) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier) (although methods such as [Platt scaling](https://en.wikipedia.org/wiki/Platt_scaling) exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

In addition to performing [linear classification](https://en.wikipedia.org/wiki/Linear_classifier), SVMs can efficiently perform a non-linear classification using what is called the [kernel trick](https://en.wikipedia.org/wiki/Kernel_trick), implicitly mapping their inputs into high-dimensional feature spaces.

When data is un-labelled, supervised learning is not possible, and an [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) approach is required, which attempts to find natural [clustering of the data](https://en.wikipedia.org/wiki/Cluster_analysis) to groups, and then map new data to these formed groups. The **support-vector clustering** algorithm, created by [Hava Siegelmann](https://en.wikipedia.org/wiki/Hava_Siegelmann" \o "Hava Siegelmann) and [Vladimir Vapnik](https://en.wikipedia.org/wiki/Vladimir_Vapnik), applies the statistics of support vectors, developed in the support vector machines algorithm, to categorize unlabeled data, and is one of the most widely used clustering algorithms in industrial applications

**Types of SVM algorithm:**

Support Vector Machines are based on the concept of decision planes that define decision boundaries. A decision plane is one that separates between a set of objects having different class memberships. A schematic example is shown in the illustration below. In this example, the objects belong either to class GREEN or RED. The separating line defines a boundary on the right side of which all objects are GREEN and to the left of which all objects are RED. Any new object (white circle) falling to the right is labeled, i.e., classified, as GREEN (or classified as RED should it fall to the left of the separating line)**.**

****

Figure 5.1 Decision boundaries

The above is a classic example of a linear classifier, i.e., a classifier that separates a set of objects into their respective groups (GREEN and RED in this case) with a line. Most classification tasks, however, are not that simple, and often more complex structures are needed in order to make an optimal separation, i.e., correctly classify new objects (test cases) on the basis of the examples that are available (train cases). This situation is depicted in the illustration below. Compared to the previous schematic, it is clear that a full separation of the GREEN and RED objects would require a curve (which is more complex than a line). Classification tasks based on drawing separating lines to distinguish between objects of different class memberships are known as hyperplane classifiers. Support Vector Machines are particularly suited to handle such tasks.

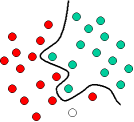


Figure 5.2 Hyperplane classifiers

The illustration below shows the basic idea behind Support Vector Machines. Here we see the original objects (left side of the schematic) mapped, i.e., rearranged, using a set of mathematical functions, known as kernels. The process of rearranging the objects is known as mapping (transformation). Note that in this new setting, the mapped objects (right side of the schematic) is linearly separable and, thus, instead of constructing the complex curve (left schematic), all we have to do is to find an optimal line that can separate the GREEN and the RED objects.

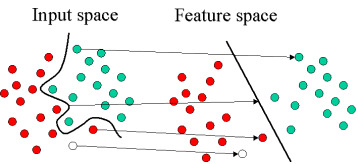


Figure 5.3 Mapping

Support Vector Machine (SVM) is primarily a classier method that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels. SVM supports both regression and classification tasks and can handle multiple continuous and categorical variables. For categorical variables a dummy variable is created with case values as either 0 or 1. Thus, a categorical dependent variable consisting of three levels, say (A, B, C), is represented by a set of three dummy variables:

A: {1 0 0}, B: {0 1 0}, C: {0 0 1}

To construct an optimal hyperplane, SVM employs an iterative training algorithm, which is used to minimize an error function. According to the form of the error function, SVM models can be classified into four distinct groups:

* Classification SVM Type 1 (also known as C-SVM classification)
* Classification SVM Type 2 (also known as nu-SVM classification)
* Regression SVM Type 1 (also known as epsilon-SVM regression)
* Regression SVM Type 2 (also known as nu-SVM regression)

Following is a brief summary of each model.

## Classification of SVM:

### TYPE 1:

For this type of SVM, training involves the minimization of the error function:

IMG_259

subject to the constraints:

IMG_260

where C is the capacity constant, w is the vector of coefficients, b is a constant, and IMG_261 represents parameters for handling non-separable data (inputs). The index i labels the N training cases. Note that IMG_262 represents the class labels and xi represents the independent variables. The kernel IMG_263 is used to transform data from the input (independent) to the feature space. It should be noted that the larger the C, the more the error is penalized. Thus, C should be chosen with care to avoid over fitting.

### TYPE 2

In contrast to Classification SVM Type 1, the Classification SVM Type 2 model minimizes the error function:

IMG_264

subject to the constraints:

IMG_265

In a regression SVM, you have to estimate the functional dependence of the dependent variable y on a set of independent variables x. It assumes, like other regression problems, that the relationship between the independent and dependent variables is given by a deterministic function f plus the addition of some additive noise:

## Regression SVM:

y = f(x) + noise

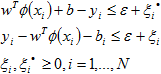
The task is then to find a functional form for f that can correctly predict new cases that the SVM has not been presented with before. This can be achieved by training the SVM model on a sample set, i.e., training set, a process that involves, like classification (see above), the sequential optimization of an error function. Depending on the definition of this error function, two types of SVM models can be recognized:

### **REGRESSION SVM TYPE 1:**

For this type of SVM the error function is:

IMG_266

which we minimize subject to:

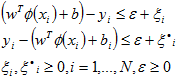


### **REGRESSION SVM TYPE 2:**

For this SVM model, the error function is given by:

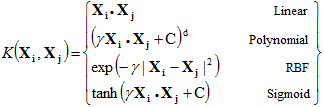
IMG_268

which we minimize subject to:



There are number of kernels that can be used in Support Vector Machines models. These include linear, polynomial, radial basis function (RBF) and sigmoid:

**Kernal Functions**



where  IMG_257

that is, the kernel function, represents a dot product of input data points mapped into the higher dimensional feature space by transformation IMG_258. **Gamma is an adjustable parameter of certain kernel functions.** The RBF is by far the most popular choice of kernel types used in Support Vector Machines. This is mainly because of their localized and finite responses across the entire range of the real x-axis.