Computer Vision

Chapter 1: Introduction

Computer Vision vs Image Processing

- Computer Vision: a field of computer science that works on enabling a computers to `see`, `identify` and `process` images as in the same way that human vision does and provide appropriate output. Input is image/video and output is intrepretation.
- Image Processing: the act of `manipulating`, `interpreting`, and `analysis` of an image object. Image is both the input and the output.

Steps in Computer Vision

- Camera -> Image Processing -> Pattern / AI model decision

Goals of Computer Vision

- What is the image about?
- What objects are in the image?
- How are they oriented?
- What is the layout of the scene in 3D?
- What is the shape of each object?

How Does Computer Vision Work?

- Input (`image`, `video`) -> Processing (`Feature extraction`, `Pattern recognition`, `Deep learning`) -> Output (`Object identification`, `Action prediction`, `Decision making`).

Computer Vision Tasks

- Image Classification: labeling entire images based on content.
- Object Detection: identifying objects in an image and their locations.
- Image Segmentation: partitioning an image into regions for analysis.
- Object Recognition: task of identifying and classifying objects in an image or video.
- Image Generation: creating new images from scratch or modifying existing images using algorithms.

Applications of Computer Vision

- Medical Imaging
- Autonomous Vehicles
- Facial Recognition

- Agriculture

Image Processing Techniques

- Edge Detection: identifying object boundaries. Example: Canny Edge Detection.
- Color Space Conversion: changing image representation (RGB to grayscale).
- Filtering: removing noise from images.

Deep Learning in Computer Vision

- Neural Networks
- Pre-trained Models

Challenges in Computer Vision

- `Lighting Variations` : difficulty recognizing objects in different lighting conditions.
- `Occlusion` : when objects are partially blocked or hidden.
- `Scale and Viewpoint Variation` : changes in size and angle of the objects.
- No data value
- Shadows on image
- Mixed pixel value

Future of Computer Vision

- 3D Image Understanding.
- AI-Powered Real-Time Object Detection.
- Advanced Healthcare Diagnostics.
- Augmented and Virtual Reality (AR/VR).

Image processing

- Method used to perform operations on images to enhance them, extract meaningful information, or prepare them for analysis.
- It involves manipulating an image to improve its quality, detect features, or transform it into a different format or representation.

Image

- A two-dimensional visual representation of an object, scene, or information, created by capturing or generating light or electromagnetic waves.
- Analog Image: continuous representation, such as a photograph or a painting, where intensity values change smoothly.
- Digital Image: discrete representation consisting of pixels, where each pixel has an intensity or color value.

How Images are Formed?

- `Natural images` : formed by light interacting with objects and being captured by cameras and sensors.

- `Synthetic images` : created using computer graphics or simulations.

Components of Image Formation

- **Light source**:
  - Provides illumination (e.g., sunlight, artificial light).
  - Determines scene visibility and appearance.
- **Object**:
  - Reflects or emits light that the camera captures.
  - Focuses light rays onto a flat image plane (sensor)
  - Key properties:
    - `Focal length` : distance between the lens and the focus point.
    - `Aperture` : controls the amount of light entering the lens.
- **Cameras**:
  - `Pinhole camera` : a simple aperture-based system.
  - `Lens-based camera` : includes focus-adjustable lenses.
  - `Stereo camera` : two lenses capturing depth information.
  - `Multispectral camera` : capturing light at different wavelengths.
- **Summary**:
  - `Light Source` : provides illumination to the scene.
  - `Camera Optics` : lenses focus the light to form an image.
  - `Sensors` : capture the light and convert it into a digital format (pixels)

Radiometry of Image Formation

- Radiance (L): light energy emitted or reflected by an object.
- Irradiance (E): light energy received by the sensor.
- Sensors capture irradiance and map it to pixel intensity.

$$I(x, y) = g(E(x, y))$$

- Where g is Nonlinear response of the sensor

Color and light

- Each pixel records amount of energy in `red` light, `blue` light, and `green` light.

Eye as measurement device

- Light is measured by the `photoreceptors` in the retina.
- Photoreceptor cells absorb photons and convert to electrical signals.
- Different photoreceptor types respond to different wavelengths.
- Retina is composed of two major classes of photoreceptors known as the `rods` and `cones` .

Image sources

- `RGB` : 3 separated bands.
- `Multispectral` : N separated bands.

- `Hyperspectral` : continous spectrum.

Digital Image Representation

- Pixels: smallest unit of an image, representing intensity or color.
- Grayscale Images: single intensity value (0 to 255).
- Color Images: combination of `Red` , `Green` , and `Blue` channels (RGB).
- Resolution: number of pixels (e.g., 1920x1080).
- Bit Depth: number of bits per pixel (e.g., 8-bit, 16-bit).

Subdomain of image processing

- Applied math, signal processing, computer photography, computer vision, statistics, machine learning, and graphics.

Image Formation

- The process of capturing a 3D scene and representing it as a 2D digital image.

Image Representation

- Refers how an image is stored and interpreted digitally or in computational format.
- Pixel Representation:
  - An image is divided into a grid of tiny squares called pixels.
  - Each pixel represents a specific intensity or color:
- Color Models:
  - `Grayscale Images` : each pixel value ranges from 0 (black) to 255 (white) in an 8-bit system.
  - `RGB (Color Images)` : pixels have three components—Red, Green, and Blue represented as a combination to create a wide range of colors.
- Bit Depth: Determines the number of colors or intensity levels an image can have. Examples: 8-bit, 16-bit, or 32-bit images.

Objective of Image Processing

- The primary objective of image processing is to enhance and analyze images to extract meaningful information or make them more suitable for specific applications
- **Basic tasks**:
  - Image Enhancement: improve the visual appearance of images for better interpretation by humans. Example: Brightening, contrast adjustment, noise removal.
  - Image Restoration: reconstruct or recover degraded or corrupted images to their original form. Example: De-blurring, removing noise, and correcting distortions.
  - Image Compression: reduce the size of an image file for efficient storage and transmission.
  - Feature Extraction: identify and extract significant features like edges, corners, or regions for analysis. Example: Object detection, facial recognition.
  - Image Segmentation: divide an image into meaningful parts for further analysis.

Image processing techniques

- **Image Filtering**:
  - A critical process in image processing, designed to remove noise, enhance details, or extract specific features.
  - Work by modifying pixel values based on certain criteria, often considering neighboring pixels.
  - Types:
    - `Spatial Domain Filtering` :
      - Used to reduce noise or blur an image by averaging pixel values.
      - `Mean Filter (Averaging Filter)` : each pixel is replaced with the average of its neighbors.
      - `Gaussian Filter` : uses a gaussian kernel to smooth an image to reduce high frequency noise more effectively than a mean filter.
    - `Sharpening Filter` :
      - Enhance edges or details in an image by emphasizing high frequency components.
      - `Laplacian Filter` : computes the second derivative of the image, highlighting regions of rapid intensity change.
      - Sharpens an image is computed by subtracting a blurred version from the original.
    - `Edge Detection Filters` :
      - `Sobel Filter` : detects edges by calculating gradients in horizontal (X) and vertical (Y) directions.
      - Gradients are used to identify boundaries between regions in an image, such as edges where there is a significant change in pixel values.
      - Sobel operator, for instance calculates the first derivative of an image in the horizontal and vertical directions (usually represented as G_x and G_y):
        - G_x: Gradient in the x-direction (horizontal edges).
        - G_y: Gradient in the y-direction (vertical edges).
      - `Canny Edge Detection` :
        - One of the most popular and effective edge detection techniques used in image processing.
        - A multi stage algorithm to detect edges with precise thresholds:
          - *Noise Reduction (Smoothing)*: To reduce noise in the image before detecting edges, as noise can lead to false edges.
          - *Gradient Calculation (Edge Detection)*: o detect areas where there are significant changes in intensity (edges).
          - *Non-Maximum Suppression (NMS)*: To thin out the edges and remove unnecessary pixels. This step ensures that the detected edges are thin and well-defined.
          - *Double Thresholding*: To classify edge pixels as strong, weak, or non-edges.

            - *Edge Tracking by Hysteresis*: To finalize edge detection by connecting weak edges that are connected to strong edges, ensuring that the detected edges form continuous contours.
        - `Frequency Domain Filtering` :
            - Transforms the image into its frequency components (Fourier Transform), modifies specific frequencies, and transforms it back to the spatial domain.
            - *Low-Pass Filtering*: removes high-frequency components (noise) to smooth the image.
            - *High-Pass Filtering*: removes low-frequency components, enhancing edges and details.
        - `Non-linear filters` :
            - *Median Filter*: reduces noise by replacing a pixel's value with the median of its neighborhood.
            - *Bilateral Filter*: smooths the image while preserving edges.
            - *Custom Kernel Filtering*: users can define custom filters to achieve specific effects.
- **Image Segmentation**:
    - A crucial process in computer vision and image processing, where an image is divided into multiple segments or regions.
    - Used for detecting objects, boundaries, and features in an image.
    - Types:
        - `Thresholding based` :
            - One of the simplest techniques for image segmentation that divides an image into foreground and background based on pixel intensity.
            - `Global threshold` : pixels above the threshold are classified as foreground, and those below are classified as background.
            - `Otsu's Thresholding` : an adaptive technique that automatically determines the best threshold based on the image histogram.
        - `Edge based` :
            - Focuses on detecting the boundaries of objects within an image by identifying edges in the image, which are the places where the intensity of pixels changes sharply.
            - `Canny Edge Detection` : used for identifying the edges in an image.
            - `Sobel Edge Detection` : computes gradients of the image and finds edges based on those gradients.
        - `Region based` :
            - Involves dividing the image into regions based on predefined criteria such as intensity or texture.
            - It usually starts with an initial seed region and then expands to neighboring pixels that are similar.
        - `Clustering based` :
            - Groups pixels into clusters based on their similarity, often using unsupervised methods.

- `K-means Clustering` : a popular clustering algorithm that partitions pixels into K clusters based on their color or intensity values.
- `Mean-Shift Clustering` : a non-parametric clustering technique that is used to detect regions in an image.

- `Deep learning based` :
  - Models like Convolutional Neural Networks (CNNs) and fully convolutional networks (FCNs) learn complex patterns from large datasets and can segment images with high accuracy.

Chapter 2: Introduction to Machine Learning

Why Machine Learning?

- The complication of some problems and lack of efficient implementation for those problems.
- Programs produced by the learning algorithm can change when data changes by taking a lot of examples.

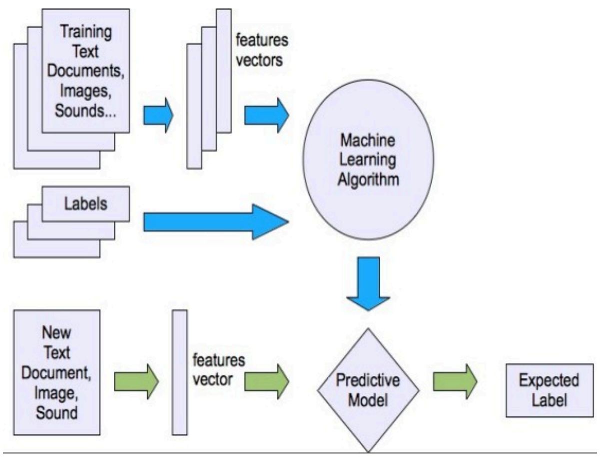
What's Machine Learning?

- A field of study that gives computers the ability to learn without being explicitly programmed.
- A computer program is said to learn from experience E with respect to some class of task T and performance measure P, if its performance at task in T, as measured by P, improves with experience E.
- Building computational artifacts/ objects that learn over time based on experience.
- Includes maths, scicen, engineering and computing.

Major Classes of ML Algorithms

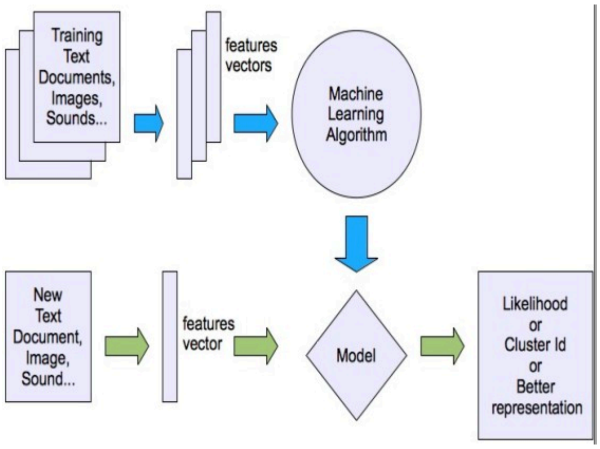
Supervised Learning

- Algorithm is on labeled dataset.
- It learns to map input features to targets based on labeled training data.
- It's provided with input and corresponding output labels to generalize from this data to make predictions on new unseen data.
- Regression and classification problems.



Unsupervised Learning

- Find clusters of similar inputs in the data without being explicitly told that some data points belong to one class and the other in other classes.
- The algorithm has to discover this similarity by itself.
- Discover a good internal representation of the input.



Reinforcement Learning

- The algorithm searches over the state space of possible inputs and outputs in order to maximize a reward.
- Learn to select an action to maximize payoff.

Machine Learning vs AI

- **Machine Learning**:
  - Subset of AI focused on learning from data.
  - More of induction (from specific to general).
  - Data is centeral.
- **AI**:
  - A broader concept that seeks to simulate human intelligence.
  - More of deduction (from general to specific).
  - Algorithm is centeral.

Machine Learning Application

- What are the serious of steps I need to do in order to solve some problem?
- If I tried to describe this problem in a particular way, is it solvable?

Machine Learning

- Machine learning is the semi-automated extraction of knowledge from data?
  - `Knowledge from data` : Starts with a question that might be answerable using data.
  - `Automated extraction` : A computer provides the insight.
  - `Semi-automated` : Requires many smart decisions by a human.

How does ML “work”?

- High-level steps of supervised learning:
  - First, train a machine learning model using labeled data.
  - “Labeled data” has been labeled with the outcome.
  - “ML model” learns the relationship b/n the attributes of the data and its outcome.
  - Then, make predictions on new data for which the label is unknown.

Questions about ML

- How do I choose which attributes of my data to include in the model?
- How do I choose which model to use?
- How do I optimize the model for best performance?
- How do I ensure that I’m building a model that will generalize to unseen data?
- Can I estimate how well my model is likely to perform on unseen data?

Chapter 3: Classification

Key Terminologies

- **Features**: Features (attributes) describe an instance.
- **Target Variable**: The target variable is what the model aims to predict. In classification problems, target variables are finite classes.
- **Training Set**: A dataset with known target variables used to train the algorithm.
- **Test Set**: A separate dataset with unknown target variables used to test the model.

Key Tasks of Machine Learning

- **Classification**: Predicts the class/category of an instance (supervised learning).
- **Regression**: Predicts numeric values (supervised learning).
- **Supervised Learning**: Tasks where the algorithm is trained with labeled data, specifying what to predict.
- **Unsupervised Learning**: Tasks without labeled data, including:
  - Clustering: Grouping similar data points.
  - Density Estimation: Finding statistical patterns in the data.
  - Dimensionality Reduction: Reducing features to visualize data in 2D or 3D.

Supervised learning tasks	
k-Nearest Neighbors	Linear
Naive Bayes	Locally weighted linear
Support vector machines	Ridge
Decision trees	Lasso
Unsupervised learning tasks	
k-Means	Expectation maximization
DBSCAN	Parzen window

Choosing the right Algorithm

- **Identify Your Goal**:
  - For predicting or forecasting target values, use `supervised learning`.
  - For uncovering patterns without target values, use `unsupervised learning`.
- **Supervised Learning**:
  - Classification: Target values are `discrete` (e.g., Yes/No, Red/Yellow/Black).
  - Regression: Target values are `continuous` (e.g., 0.00 to 100.00).
- **Understand Your Data**:
  - Check if features are `nominal` or `continuous`.
  - Identify missing values and their causes.
  - Look for `outliers`.
  - A deeper understanding of your data helps narrow down algorithm choices.
- **Iterative Process**: Finding the best algorithm requires trial and error.

Steps to Develop a Machine Learning Application

- Collect Data: Gather data via scraping, APIs, or other sources.
- Prepare Data: Ensure data formats are consistent and clean.
- Analyze Data: Explore and understand the data.
- Train Algorithm: Apply the model (not applicable for unsupervised learning).
- Test Algorithm: Evaluate performance and iterate as needed.
- Deploy: Implement the machine learning solution.

Problem Solving Framework

- Business issue understanding
- Data understanding
- Data preparation
- Analysis Modeling
- Validation
- Presentation / Visualization

Classifying with K-Nearest Neighbors

- A simple and effective algorithm for `classification` and `regression`.
- Works with numeric and nominal values.
- **How It Works**:
  - Use a training set with labeled data.
  - For a new, unlabeled data point:
    - Compare it to all existing data points in the training set.
    - Identify the k most similar points (nearest neighbors).
    - Use a majority vote among the k neighbors to classify the new data point.
- **Generalized Approach to kNN**:
  - Collect Data: Gather data using any method (e.g., scraping, APIs, or datasets).

   - Prepare Data: Ensure numeric values for distance calculation and clean the data as needed.
   - Analyze Data: Use methods like visualization or statistical analysis to understand the data.
   - Train: Not applicable, as kNN is a lazy learning algorithm (it doesn't require training).
   - Test: Evaluate performance by calculating error rates (e.g., accuracy or mean error).
   - Use: Input structured data and output classifications or predictions.
- **Advantages of kNN**:
   - High accuracy.
   - Remembers all training data (lazy learning).
   - No training time required, making it fast for small datasets.
   - Simple and easy to implement.
- **Disadvantages of kNN**:
   - No generalization beyond the training set.
   - Sensitive to noise and outliers, leading to potential overfitting.
   - Computationally expensive for large datasets (distance calculation for all points).

Eculidean and Manhatan distance

- **Eculidean distance**:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

- **Manhatan distance**:

$$d = |x_2 - x_1| + |y_2 - y_1|$$

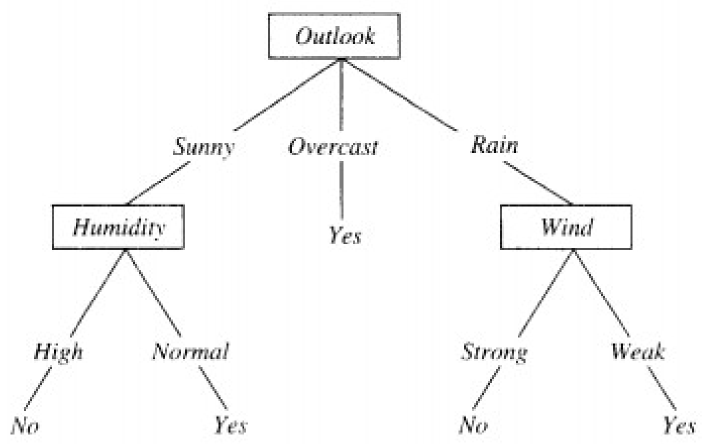
Decision Tree

- A popular classification technique that splits datasets based on features, one at a time.
- **Structure**:
   - Decision Blocks: Represented as rectangles.
   - Termination Blocks: Represented as ovals.
   - Branches: Arrows that connect decision and termination blocks.
- Unlike kNN, decision trees provide interpretable insights into the data by visualizing decision paths.
- Easy for humans to understand and visualize.
- **How It Works**:
   - Takes a dataset (training examples).
   - Builds a decision tree (model) and visualizes it.
   - Can be converted into if-then rules for better readability.
- **Key Benefits**:
   - Extracts knowledge by distilling data into clear rules.
   - Handles unfamiliar data effectively by generating interpretable rules.

Expressing Decision Tree in Logical Expression

- **Steps**:

- Identify the paths leading to a “Yes” decision:
  - Start from the root node and follow the branches to the terminal blocks labeled “Yes.”
  - For each path, list the conditions (feature values) that must be true.
- Combine the conditions in each path: Use conjunctions (⋀ meaning “AND”) to combine conditions in a single path.
- Combine all “Yes” paths: Use disjunctions (⋁ meaning “OR”) to connect all the paths that lead to “Yes.”
- Example:
  - Consider the following decision tree



  - **Path 1**:
    - Start at “Outlook = Sunny.”
    - Go to “Humidity = Normal.”
    - `Result` : “Yes.”
    - `Expression` : (Outlook = Sunny ⋀ Humidity = Normal) → Yes
  - **Path 2**:
    - Start at “Outlook = Overcast.”
    - `Result` : “Yes.”
    - `Expression` : (Outlook = Overcast) → Yes
  - **Path 3**:
    - Start at “Outlook = Rain.”
    - Go to “Wind = Weak.”
    - `Result` : “Yes.”
    - `Expression` : (Outlook = Rain ⋀ Wind = Weak) → Yes
  - **Combine all the paths using**:
    - (Outlook = Sunny ⋀ Humidity = Normal) → Yes ⋁
    - (Outlook = Overcast) → Yes ⋁
    - (Outlook = Rain ⋀ Wind = Weak) → Yes.

Pros and Cons of Decision Trees

- Pros:
  - Efficiency: Computationally inexpensive to use.
  - Interpretability: Easy for humans to understand and interpret results.
  - Robustness: Handles missing values and irrelevant features effectively.

- **Cons**:
  - Overfitting: Can overfit the training data.

Appropriate Use Cases for Decision Trees

- **Attribute-Value Representation**: Instances are represented as fixed attributes with specific values.
- **Discrete Outputs**: Target functions yield discrete outcomes.
- **Complex Descriptions**: Disjunctive descriptions (multiple conditions) may be required.
- **Error Tolerance**: Can handle errors in training data.
- **Missing Data**: Can work with datasets with missing attribute values.

Decision Tree Splitting Process

- Uses information theory to decide the best feature to split the data.
- **Steps**:
  - Evaluate all features to determine the optimal split.
  - Divide the dataset into subsets based on the chosen feature.
  - Traverse subsets down the branches of the decision node.
  - Stop splitting when data in a branch is uniform; otherwise, continue the process.
- This approach enables decision trees to segment data effectively while maintaining interpretability.

```
Check if every item in the dataset is in the same class:
    If so return the class label
    Else
        find the best feature to split the data
        split the dataset
        create a branch node
            for each split
                call createBranch and add the result to the branch node
        return branch node
```

General Approach to Decision Trees

- Collect: Gather data using any suitable method.
- Prepare: If using the ID3 algorithm, convert continuous values into nominal (discrete) values as ID3 only works with nominal data.
- Analyze: Inspect the tree visually after it is built to ensure proper structure and splits.
- Train: Construct a tree data structure by splitting the data based on the best features.
- Test: Evaluate the learned tree by calculating its error rate on test data.
- Use: Apply the decision tree in any supervised learning task, typically for classification, and to gain insights from the data.

Information Theory and Decision Trees

- **Information Theory**: A branch of science focused on quantifying information.
- **Information Gain**:
  - The change in information before and after a split in a decision tree.

- The split with the highest information gain is considered the best.
- **Shannon Entropy**:
  - A measure of information in a dataset.
  - Higher entropy indicates more disorder or randomness in the data.
- **Gini Impurity**: Another measure of disorder, representing the probability of misclassification when picking an item at random from the dataset.

Building Decision Trees

- **Entropy Calculation**: To calculate entropy, use the expected value of all possible class values.

$$H = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

- **Dataset Splitting**: Evaluate the information gain when splitting a dataset on a given feature.
- **Recursion**: Recursively split the dataset based on the best attribute, until `no attributes remain`, or instances in a branch belong to the same class.

Types of Decision Trees

- Classification Tree: Target variable has finite, discrete values.
- Regression Tree: Target variable has continuous values.

Popular DT Algorithms

- `ID3`: Iterative Dichotomiser 3.
- `C4.5`: Successor to ID3.
- `CART`: Classification and Regression Tree.
- `CHAID`: Chi-squared Automatic Interaction Detector.
- `MARS`: Extends DT for better handling of numerical data.

Attribute Selection Measures

- Information Gain: Used by ID3, C4.5, and C5.0.
- Gini Impurity: Used by CART.
- Gain Ratio: Addresses bias toward attributes with many values (used in C4.5).

ID3 Characteristics

- Explores a complete hypothesis space of finite discrete-valued functions.
- Maintains a single hypothesis during the search, with no backtracking (post-pruning addresses overfitting).
- Uses all training examples at each step to make statistical decisions, making it robust to noise.

Challenges in Decision Tree Learning

- Tree Depth: Determining how deeply to grow the tree.
- Continuous Attributes: Handled by discretizing or using multiple intervals.
- Attribute Selection: Choosing the right measure (e.g., Information Gain or Gain Ratio).

- Missing Values: Addressed by assigning the most common value or probability-based imputation (used in C4.5).
- Attribute Costs: Introduce a cost term into the selection process.
- Computational Efficiency: Optimizing training for large datasets.

Avoiding Overfitting

- Overfitting Risks:
  - Common with noisy data or small training sets.
  - Can reduce accuracy by 10–25%.
- Solutions:
  - `Pre-Pruning` : Stop tree growth early (less practical).
  - `Post-Pruning` : Grow the tree fully and prune later (preferred in practice).

Extending DT Learning

- Continuous-Value Attributes: Convert to Boolean or interval-based splits.
- Improving Attribute Selection: Use Gain Ratio instead of Information Gain to reduce bias.
- Handling Missing Values: Use the most common value or assign probabilities
- Addressing Attribute Costs: Divide the gain by the attribute cost for selection.

Chapter 4: Recognition

What's Recognition?

- The process of `identifying` , `categorizing` , or `assigning` a label to an object, pattern, or scene based on its distinguishing features or characteristics.
- Involves analyzing data and matching it to predefined classes or categories.
- Example: `Visual` , `Speech` , `Facial` , and `Scene` recognition.

Key Aspects of Recognition

- Understanding and interpreting the input data.
- Matching features of the data against stored templates or learned models.
- Assigning the best-matching category or class to the input data.

Image features and categorization

- General concepts of categorization: `Why?` `What?` `How?`
- Image features:
  - Color, texture, gradient, shape, interest points
  - Histograms, feature encoding, and pooling
  - CNN as feature

Image Recognition vs Image Detection

- **Image Recognition**:

- Definition:
    - The process of identifying and classifying objects, patterns, or scenes in an image.
    - Recognition assigns a label to the entire image or specific objects within it.
  - Goal: To categorize objects or scenes into predefined classes.
  - Applications:
    - Facial recognition (e.g., unlocking smartphones)
    - Product recognition in retail (e.g., scanning groceries for pricing).
    - Medical diagnosis (e.g., identifying tumors in MRI scans).
- **Image Detection**:
  - Definition:
    - The process of locating objects within an image and identifying their presence by assigning bounding boxes or specific coordinates to each detected object.
  - Goal: To find and identify multiple objects and their positions within an image.
  - Applications:
    - Autonomous vehicles (e.g., detecting traffic signs and pedestrians).
    - Surveillance systems (e.g., detecting unauthorized access).
    - Retail inventory management (e.g., detecting product counts in a warehouse).

Scene Recognition

- Classification of the overall environment or context of an image into categories such as “beach”, “forest”, or “urban area”.
- Analyzes the global spatial arrangement and contextual relationships of features.
- Example: Categorizing an image as a “cityscape” by identifying buildings, roads, and vehicles collectively.
- Applications:
  - Autonomous navigation (e.g., differentiating between highways and urban streets).
  - Surveillance systems (e.g., identifying public spaces like parks or markets).
  - Content-based image retrieval (e.g., retrieving images of “mountains” from a travel album).
- Challenges:
  - `Diverse Categories` : The wide variety of scene types with overlapping visual elements.
  - `Data Scale` : Handling massive datasets with thousands of categories.
  - `Variability` : Changes in lighting, viewpoint, and weather conditions.
  - `Complexity` : The need for understanding high-level semantic content and spatial relationships.
- Key Components:
  - Feature Extraction:
    - `Low-level features` (e.g., edges, corners):
      - Extracted using traditional techniques like `SIFT` and `HOG` .
      - Example: Detecting edges of a building in a cityscape using edge detection
    - `High-level features` :
      - Extracted using deep learning models (e.g., CNNs).
      - Example: A CNN model identifying key objects like trees and pathways in a park scene.
    - `Scale Invariant Feature Transform (SIFT)` :

- Detects and describes local features in image that are invariant to `scale`, `rotation`, and `minor illusion` changes.
- Steps:
  - `Scale-Space Extrema Detection`: Identifies keypoints by searching for local maxima and minima in the Difference of Gaussian (DoG) across multiple scales.
  - `Keypoint Localization`: Refines keypoint locations by discarding unstable points with low contrast or along edges.
  - `Orientation Assignment`: Assigns one or more orientations to each keypoint based on the local gradient directions.
  - `Feature Descriptor`: Creates a descriptor using a histogram of gradient orientations in the neighborhood of each keypoint.
- Advantage: `scale` and `rotation` invariant.
- Example: detecting and matching landmarks between two aerial images of the same city taken from different angles.
- `Object recognition`, `image stitching`, and `3D reconstruction`.

- `Histogram of Oriented Gradients (HOG)`:
  - Captures the structure or shape of objects in an image by analyzing the distribution of gradient orientations.
  - Steps:
    - `Gradient Computation`: Calculates the gradient magnitude and direction for each pixel in an image.
    - `Spatial Cells`: Divides the image into small connected regions called `cells`.
    - `Orientation Histograms`: Creates a histogram of gradient orientations for each cell, weighted by the gradient magnitude.
    - `Block Normalization`: Normalizes the histograms over larger overlapping blocks to ensure illumination invariance.
    - `Feature Vector Formation`: Concatenates the normalized histograms into a feature vector representing the image.
  - Advantages:
    - Effective for detecting objects like pedestrians and vehicles.
    - Robust against small deformations and illumination changes.
    - Works well for classification tasks with a fixed object structure.
  - Example: Detecting pedestrians in street images using the Dalal-Triggs approach for human detection.
- `SIFT vs HOG`:

Aspect	SIFT	HOG
Type of Features	Local Keypoints	Global Shape Descriptors
Invariance	Scale, Rotation	Partial Illumination
Purpose	Matching & Recognition	Object Detection
Complexity	Higher due to multi-step processing	Lower, simpler gradient histograms

- `Gradient`:

        - The measure of how the intensity (brightness) of an image changes at a particular point.
        - It represents the direction and rate of the most significant intensity change in the neighborhood of a pixel.
        - Widely used to detect edge, textures, and other features in an image.
        - Applications: `Edge Detection`, `Feature Extraction`, `Image Segementation`, and `Optical Flow`.
  - Representation Learning:
    - `Bag of Visual Words (BoVW)`:
      - Converts local features into histograms for image representation.
      - Example: Representing a forest scene with a histogram of features like leaf textures and tree shapes.
    - `Fisher Vectors and VLAD (Vector of Locally Aggregated Descriptors)`:
      - Compact representations capturing richer information.
      - Example: Encoding detailed architectural features of a Gothic cathedral in an urban scene.
    - `Deep Feature Encoding`:
      - Learned representations through layers of neural networks.
      - Example: A ResNet model encoding the spatial and texture details of a snowy mountain scene
    - `Image categorization with bag of words`:
      - Training:
        - Extract keypoints and descriptors for all training images
        - Cluster descriptors
        - Quantize descriptors using cluster centers to get “visual words”
        - Represent each image by normalized counts of “visual words”
        - Train classifier on labeled examples using histogram values as features
      - Testing:
        - Extract keypoints/descriptors and quantize into visual words.
        - Compute visual word histogram.
        - Compute label or confidence using classifier
  - Classification:
    - `Traditional Classifiers`:
      - SVM, Random Forest.
      - Example: Using SVM to classify between “desert” and “savanna” based on extracted features.
    - `Modern Classifiers`:
      - Fully connected layers in deep learning networks.
      - Example: A fully connected layer in a CNN outputting “suburban” as the predicted class.

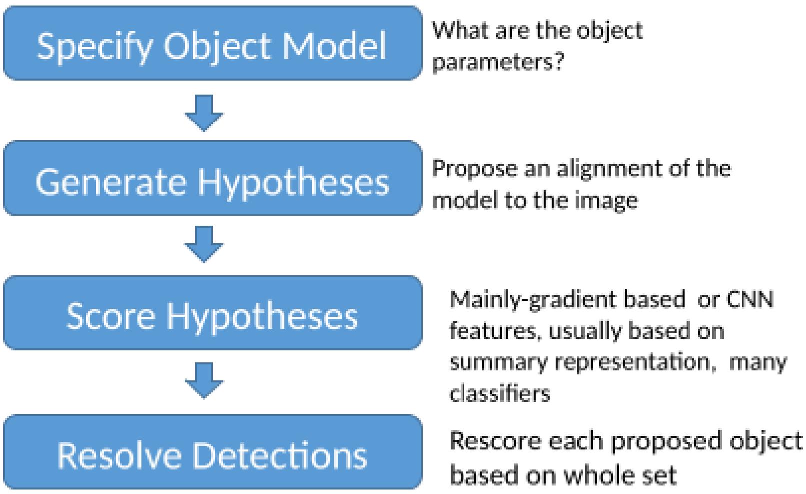
Advanced Feature Encoding Techniques

- **Deep Features and Transfer Learning**:
  - `CNNs`:

        - Extract hierarchical features capturing both local and global patterns.
        - Example: VGG16 recognizing both the texture of grass and the layout of pathways in a park.
    - `Transfer Learning` :
        - Pre-trained models like ResNet, Inception, and ViT for feature extraction.
        - Example: Fine-tuning a pre-trained ResNet model for classifying interior scenes like “living room” or “kitchen.”
- **Attention Mechanisms**:
    - `Self-Attention` :
        - Focuses on important regions of an image for better encoding.
        - Example: Highlighting the skyline and skyscrapers in a “city” scene.
    - `Vision Transformers (ViT)` :
        - Breaks images into patches and applies attention for global context capture.
        - Example: A ViT model analyzing both the sand and water regions in a “beach” image.
- Hybrid Representations:
    - Combining handcrafted and deep features to leverage the strengths of both approaches.
    - Example: Using HOG for edge detection and a CNN for texture analysis in a “forest” scene.
- Multi-Scale Feature Encoding:
    - Captures information at different scales to recognize both fine details and global structure.
    - Example: Recognizing individual leaves in a “garden” scene as well as the overall layout of flowerbeds

Detection with Sliding Windows, Dalal-Triggs, and Viola Jones

- **Traditional methods**:
    - `Dalal-Triggs` detector (basic concept)
    - `Viola-Jones` detector (cascades, integral images)
- **Deep learning methods**:
    - Review of CNN,
    - Two-stage: `R-CNN` ,
    - One-stage: `YOLO` , `SSD` , and `Retina Net` .
- **Sliding Windows**:
    - A technique to apply a fixed-size window across an image at different scales and positions to detect objects or features.
    - Example: Using sliding windows to identify cars in an aerial cityscape.
    - Challenges: Computationally intensive due to exhaustive search over positions and scales.
- **Dalal-Triggs Method**:
    - Based on Histogram of Oriented Gradients (HOG) for human detection.
    - Key Steps:
        - Divide an image into small connected regions (cells).
        - Compute histogram of gradient orientations within each cell.
        - Normalize histograms for illumination invariance.
        - Use SVM for classification.
    - Example: Detecting pedestrians in urban scenes using gradient features.
- **General Process of Object Recognition**:

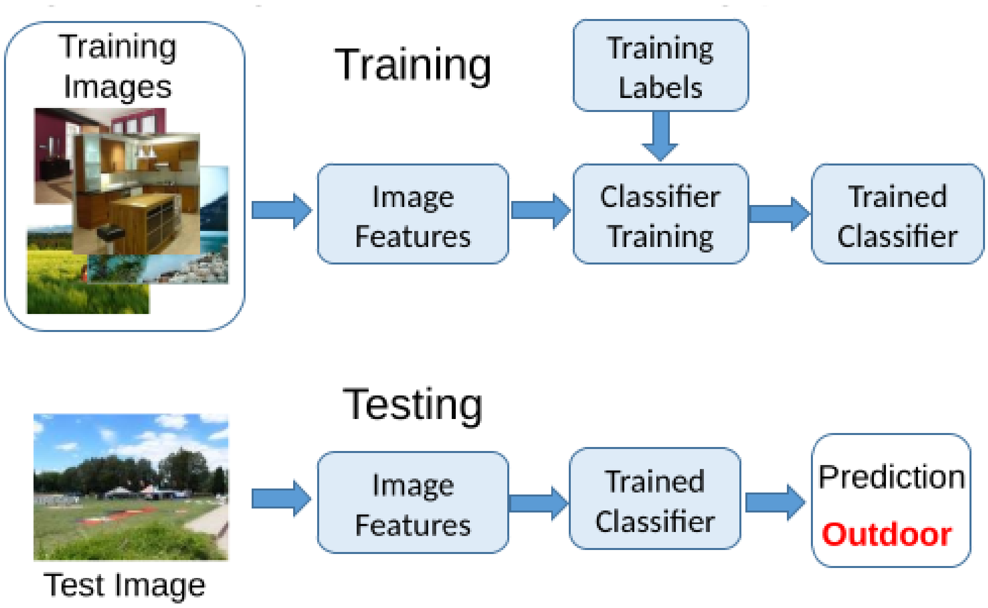


- **Basic Steps of Category Detection**:
  - Align:
    - Example: choose position, scale orientation
    - How to make this tractable?
  - Compare:
    - Compute similarity to an example object or to a summary representation.
    - Which differences in appearance are important?
- **Viola-Jones Algorithm**:
  - A real-time object detection framework primarily used for face detection.
  - Key Features:
    - `Integral Images` : Enables fast computation of feature sums.
    - `Haar-like Features` : Captures patterns like edges and lines.
    - `AdaBoost` : Combines weak classifiers to create a strong one.
    - `Cascade Classifiers` : Speeds up detection by focusing on promising regions.
    - Example: Detecting windows and doors in architectural images.

Architectures for Scene Recognition

- **Convolutional Neural Networks (CNNs)**:
  - `AlexNet` , `VGG` , `ResNet` , `DenseNet` .
  - Example: Using ResNet to classify between “industrial area” and “residential area” based on building patterns.
  - Pros: Excellent for spatial feature extraction.
  - Cons: Limited ability to capture long-range dependencies.
- **Vision Transformers (ViTs)**:
  - Exploits self-attention for global understanding.
  - Example: Identifying complex scenes like “airports” by analyzing both terminals and runways.
- **Hybrid Models**:
  - `CNN + Transformer` : Combines local feature extraction with global context modeling.
  - Example: Using CNN for local object detection and Transformer for overall scene interpretation in a “shopping mall.”.

Image Categorization



Convolutional Neural Networks

- `Input Image -> Convolution -> Non-linearity, Spatial pooling, Normalization -> Feature maps`.
- CNN can be used as feature extractor because of the extensive computational power needed by another feature extractors like `sliding window`.

Context in Recognition

- Objects usually are surrounded by a scene that can provide context in the form of nearby objects, surfaces, scene category, geometry, etc.
- Types:
    - `Local pixels` : window, surround, image neighborhood, object boundary/shape, global image statistics.
    - `2D Scene Gist` : global image statistics.
    - `3D Geometric` : 3D scene layout, support surface, surface orientations, occlusions, contact points, etc.
    - `Semantic` : event/activity depicted, scene category, objects present in the scene and their spatial extents, keywords
    - `Photogrammetric` : camera height orientation, focal length, lens distorition, radiometric, response function.
    - `Illumination` : sun direction, sky color, cloud cover, shadow contrast, etc.
    - `Geographic` : GPS location, terrain type, land use category, elevation, population density, etc.
    - `Temporal` : nearby frames of video, photos taken at similar times, videos of similar scenes, time of capture.
    - `Cultural` : photographer bias, dataset selection bias, visual cliches, etc.

Action Recognition

- `Action` is a transition from one state to another.
- Tries to answer the following questions:
  - Who is the actor?
  - How is the state of the actor changing?
  - What (if anything) is being acted on?
  - How is that thing changing?
  - What is the purpose of the action (if any)?
- We can search actions in video by using trained `HOG` detector to detect each keyframe and classify them as positive and negative.
- The purpose of the action detection is to understand the intention and motivation of the action.

Descriptor Failures and Big Data Challenges

- **Descriptor Failures**:
  - Limitations of traditional descriptors like SIFT, SURF, and HOG in complex scenarios:
    - `Lighting Variations` : Inconsistent performance under changing illumination.
    - `Occlusion` : Difficulty in handling partially visible objects.
    - `Scale Sensitivity` : Struggles with extremely large or small objects.
    - `Context Loss` : Traditional descriptors often ignore global context.
  - Example: Failing to recognize a “stadium” scene due to varying lighting and crowd occlusion.
- **Big Data Challenges in Scene Recognition**:
  - `Massive Data Volumes` : Managing and processing billions of images.
  - `Scalability` : Training deep learning models on distributed systems.
  - `Annotation Bottleneck` : Labeling large datasets is time-consuming and costly.
  - `Data Imbalance` : Unequal representation of categories leading to biased models.
  - Solutions:
    - `Distributed Computing` : Leveraging frameworks like Hadoop and Spark for data processing.
    - `Synthetic Data` : Using GANs to generate additional data for underrepresented categories.
    - `Active Learning` : Reducing annotation effort by prioritizing the most informative samples.
    - Example: Training a model on a dataset with millions of “beach” images but few “desert” examples.

Chapter 5: Neural Networks

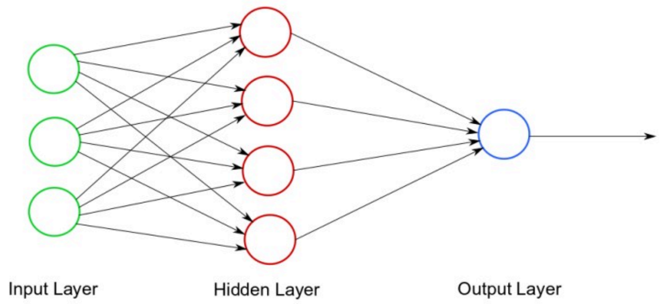
What is a Neural Network?

- A collection of neurons , or nodes linked together in a fashion that mimics the human brain.

How Does a Neural Network work?

- **Layers**:

- Input Layer: Receives raw data, like pixels from an image.
  - Hidden Layers: Perform the bulk of the processing, often multiple layers are stacked.
  - Output Layer: Produces the final result, like a classification
- **Connections**:
  - Each neuron connects to others in the next layer with associated weights.
  - `Weights` determine the influence of one neuron on another.



Current Neural Network Limitations

- Treats inputs as `independent`, `lacking awareness of relationships` (e.g., between pixels).
- Fully connected layers for high-resolution images would require an impractically large number of parameters.

Key Characteristics of Image Data

- Structural properties such as `pixel topology`, `translation invariance`, and `scale invariance`.
- Visual features like `edges`, `shapes`, `textures`, and `hierarchical patterns` (e.g., shapes forming objects).

Motivation for Specialized Architectures

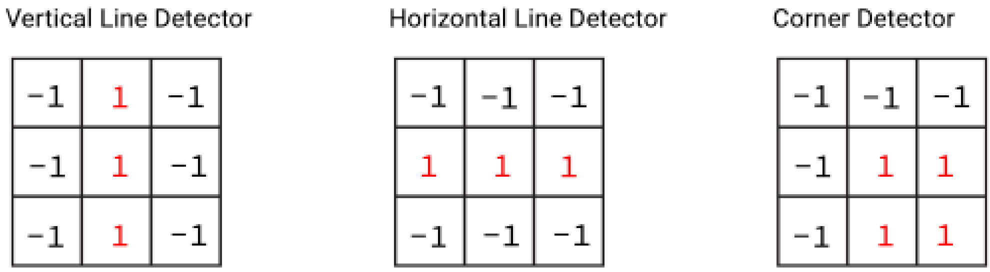
- Incorporate knowledge of human vision and image structures into neural network design.
- Reduce variance by introducing biases in the network to detect specific patterns.
- Build features hierarchically (e.g., edges → shapes → object relations).

Practical Challenges

- Large-scale images (e.g., 200x200 RGB) result in massive parameter requirements for fully connected networks.
- Inefficiency and high variance necessitate structured approaches to pattern recognition in images.

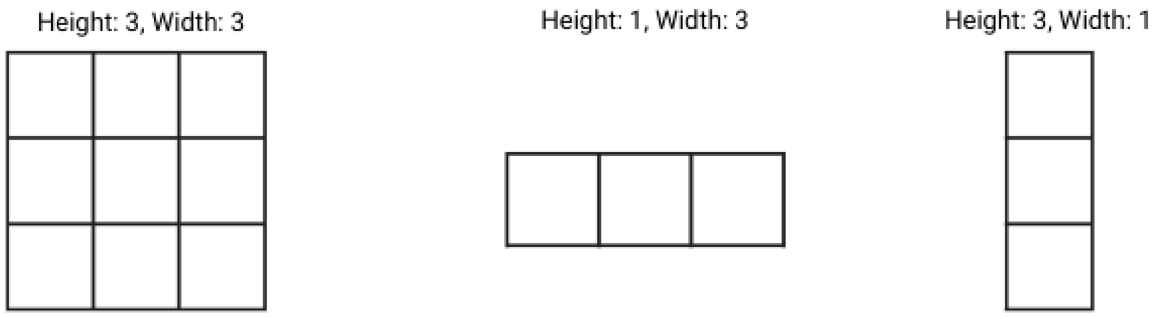
Kernels

- A `kernel` is a grid of weights applied to an image, centered on a pixel.
- Each weight is multiplied by the corresponding pixel value, and the results are summed to produce an output for the centered pixel.
- Kernels are used in traditional image processing techniques like `Blurring`, `Sharpening`, `Edge Detection` and `Embossing`.
- Kernels as Feature Detectors:

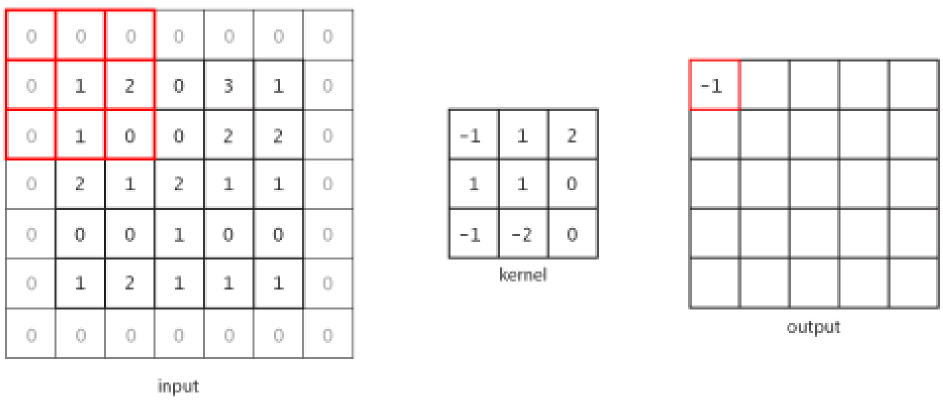


Convolutional Neural Nets

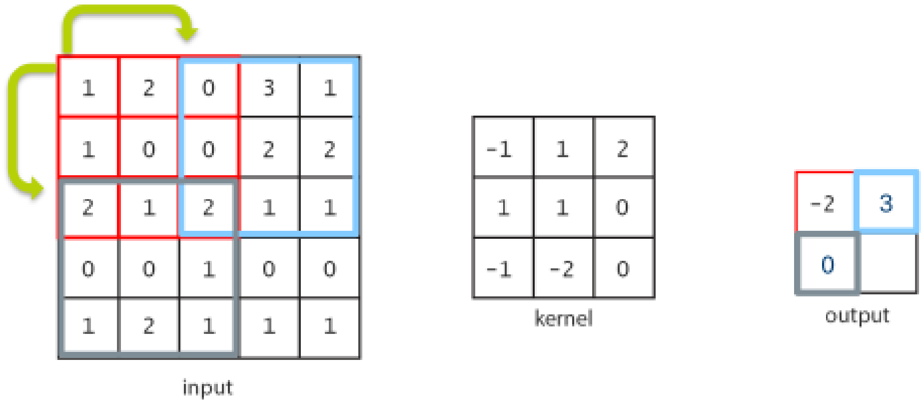
- Primary Ideas behind Convolutional Neural Networks:
  - Let the Neural Network learn which kernels are most useful,
  - Use same set of kernels across entire image (translation invariance) and
  - Reduces number of parameters and “variance” (from bias variance point of view).
- **Convolution Settings**:
  - Grid Size:
    - The number of pixels a kernel “sees” at once.
    - Typically use odd numbers so that there is a “center” pixel.
    - Kernel does not need to be square.



  - Padding:
    - Using Kernels directly, there will be an “edge effect”.
    - Pixels near the edge will not be used as “center pixels” since there are not enough surrounding pixels.
    - Padding adds extra pixels around the frame.
    - So every pixel of the original image will be a center pixel as the kernel moves across the image.
    - Added pixels are typically of value zero (zero-padding).



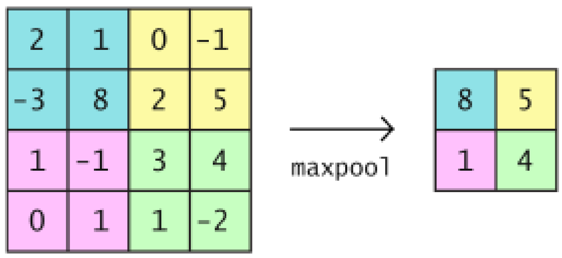
- Stride:
  - The ”step size” as the kernel moves across the image.
  - Can be different for vertical and horizontal steps (but usually is the same value).
  - When stride is greater than 1, it scales down the output dimension.



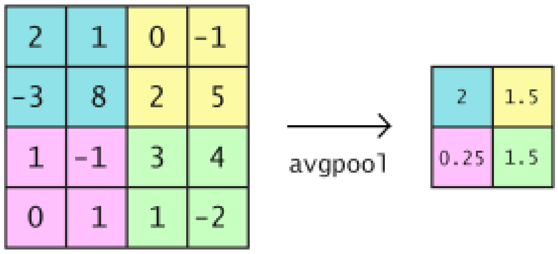
- Depth:
  - `Channels` are the multiple number associated with each pixel location.
  - The number of channels is referred to as the `depth` .
  - $weight = kernelsize \times depth$
  - The kernel itself will have a “depth” the same size as the number of input channels and the output from the layer will also have a depth.
  - The output of the layer will have number of depth equal to the number of kernels in the layer.

Pooling

- Reduce the image size by mapping a patch of pixels to a single value.
- Shrinks the dimensions of the image.
- Does not have parameters, though there are different types of pooling operations.
- **Types**:
  - Max-pool:
    - For each distinct patch, represent it by the maximum.



  - Average-pool:
    - For each distinct patch, represent it by the average.


2 1 0 -1
-3 8 2 5
1 -1 3 4
0 1 1 -2
avgpool
2 1.5
0.25 1.5