Computer Vision Notes

Ana Maria Sousa

Hugging Face

(Part I)

Content

1. Introduction

Computer vision is revolutionizing our world in many ways, with its both amazing applications and potential (from unlocking phones with facial recognition to analyzing medical images for disease detection, monitoring wildlife and creating new images).

Those notes compile basics to the latest advancements in computer vision using Hugging face course as a guide and additional sources such as articles and other courses from deeplearning.ai.

1.1. Human Vision vs Computer Vision

Our ability to see and interpret the world around us is crucial for our survival and everyday tasks. We take in light through our eyes, which then sends signals to our brain for processing. This process, called vision, is vital for our evolution, with scientists suggesting that it played a big role in the development of our complex nervous systems.

Think about kicking a ball: your brain does a lot of work without you even realizing it. It identifies the ball, tracks its movement, predicts where it will go, calculates how to kick it, and sends the signal to your foot. Computer vision, which is like teaching computers to see, tries to mimic this process.

Despite we don't need formal education for many tasks like kicking a ball, computer programs are different, they rely on rules. Replicating even a simple task like detecting a ball in an image isn't easy. We might think we can define a ball by its size or shape, but it's trickier than it seems. Not all balls are the same size or perfectly round, so it's hard to come up with a simple rule to identify them.

1.2. Why Machine Learning approach?

Our ability to recognize objects isn't just about strict rules. We often use context and generalizations. This natural skill is tough to replicate in systems with fixed rules. That's why we need smarter systems, like those in artificial intelligence, to adapt to different situations like we do.

Computer vision has come a long way, now letting us create images from text and describe images with text, all from our phones. It's everywhere, and those notes will dive into exploring its possibilities.

2. Fundamentals Image/(ing)

2.1. Image

An image is a visual representation of something, like object, a scene, a person, or even a concept. It's also an n-dimensional function. We will first consider it to be two-dimensional n=1. We will call it F(X,Y), where X and Y are spatial coordinates.

Images are made in different ways. In simple terms, images are made of pixels with different brightness or color, and they can be created in various ways.

2.1.1. 3D Images

A different type of image is **volumetric or 3D images.** The number of dimensions in 3D images is equal to three. As a result, we have **a F(X,Y,Z)** function. Most of our reasoning still applies, with the only difference being that the triplet **xi,yi,zi** is called a voxel (volume element).

These images can be acquired in 3D; that is, the images are acquired in a way that is reconstructed in a 3D space. Examples of such images include medical scans, magnetic resonance, and certain types of microscopes. It is also possible to reconstruct a 3D image from a 2D one.

Some fields, like biomedicine and microscopy, talk about 4D or 5D images. This means they're capturing more than just a regular picture. They might be taking pictures over time or using different types of imaging, like photos and x-rays. Each new piece of information adds a dimension, so a 5D image, for example, might be a 3D picture over time with different types of imaging. Thus, a 5D image is a volumetric image (3D) imaged in time (4D) and using different channels (5D).

2.1.2. Image properties and representation

Image channels are like different colors in an image. Each channel represents a color component, like red, green, or blue. The intensity of each color can vary from **0 (no color) to 255 (maximum color).**

In some cases, instead of showing color intensity, images label pixels—for example, marking foreground as **1** and background as **0.** These are called labeled images or **binary images.**

Images are commonly represented in computers as matrices, which are 2D numerical arrays. This makes it easy for computers to handle them. Alternatively, images can also be represented as graphs, where each node is a coordinate and edges connect neighboring coordinates. This means that algorithms and models used for graphs can also be applied to images, and vice versa.

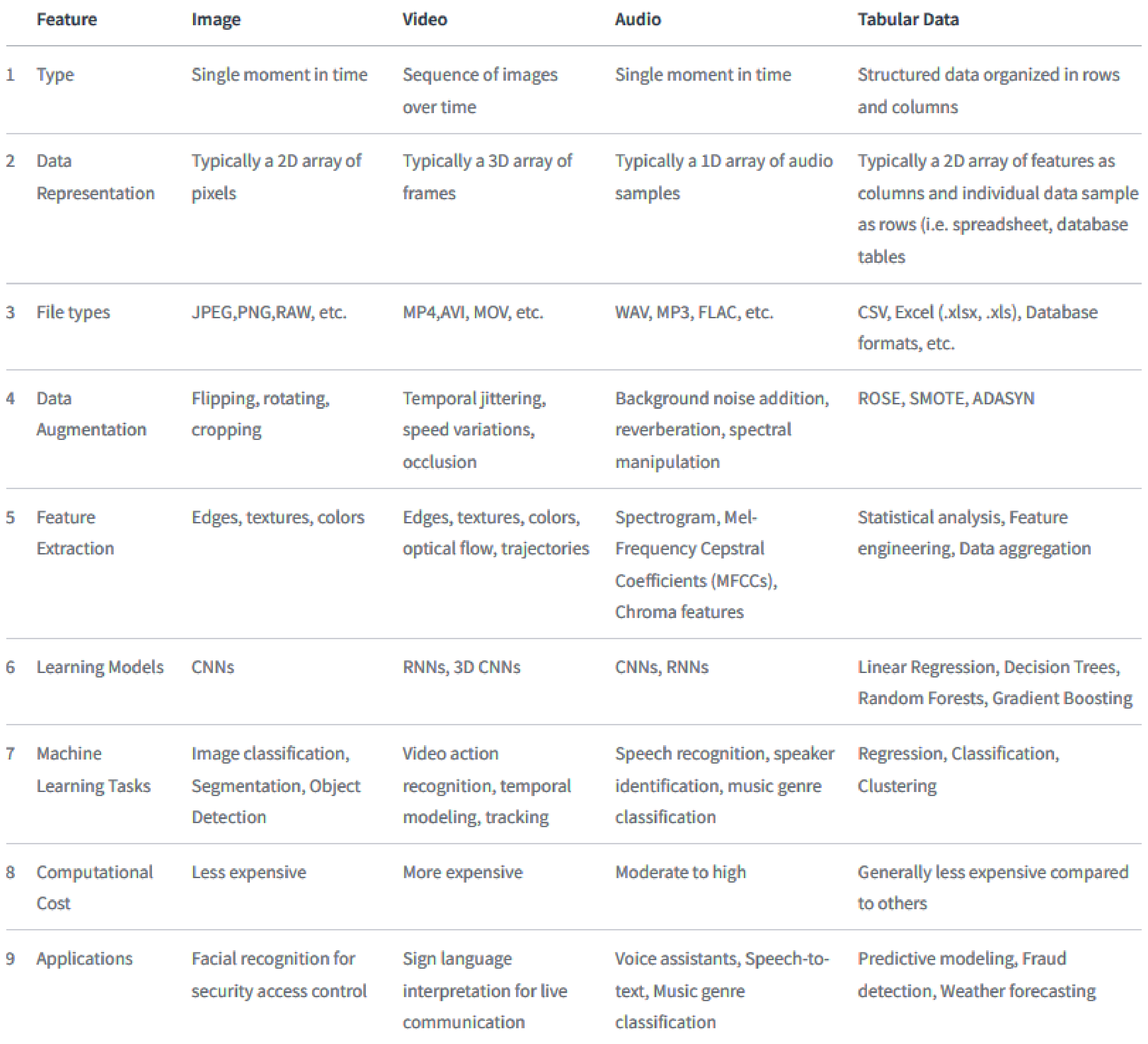
Overall, images are data points that contain a lot of spatial information, with differences in spatial resolution, color systems (like RGB), and whether they have a time component attached to them.

2.1.3. Image and Video

An image is a static representation captured at a single moment in time, while a video is a sequence of images played in succession to create the illusion of motion.

This sequence is played at a certain rate known as frames per second (fps). While images naturally have a temporal aspect, videos explicitly sample this temporal information, making them dynamic.

2.1.4. Key aspects of each data type



	Feature	Image	Video	Audio	Tabular Data
1	Type	Single moment in time	Sequence of images over time	Single moment in time	Structured data organized in rows and columns
2	Data Representation	Typically a 2D array of pixels	Typically a 3D array of frames	Typically a 1D array of audio samples	Typically a 2D array of features as columns and individual data sample as rows (i.e. spreadsheet, database tables
3	File types	JPEG,PNG,RAW, etc.	MP4,AVI, MOV, etc.	WAV, MP3, FLAC, etc.	CSV, Excel (.xlsx, .xls), Database formats, etc.
4	Data Augmentation	Flipping, rotating, cropping	Temporal jittering, speed variations, occlusion	Background noise addition, reverberation, spectral manipulation	ROSE, SMOTE, ADASYN
5	Feature Extraction	Edges, textures, colors	Edges, textures, colors, optical flow, trajectories	Spectrogram, Mel-Frequency Cepstral Coefficients (MFCCs), Chroma features	Statistical analysis, Feature engineering, Data aggregation
6	Learning Models	CNNs	RNNs, 3D CNNs	CNNs, RNNs	Linear Regression, Decision Trees, Random Forests, Gradient Boosting
7	Machine Learning Tasks	Image classification, Segmentation, Object Detection	Video action recognition, temporal modeling, tracking	Speech recognition, speaker identification, music genre classification	Regression, Classification, Clustering
8	Computational Cost	Less expensive	More expensive	Moderate to high	Generally less expensive compared to others
9	Applications	Facial recognition for security access control	Sign language interpretation for live communication	Voice assistants, Speech-to-text, Music genre classification	Predictive modeling, Fraud detection, Weather forecasting

2.2. Imaging

2.2.1. Image Acquisition in Digital Processing

It's the process of turning real-world scenes into digital images. It starts with capturing energy from an illumination source interacting with the subject. Sensors convert this energy into electrical signals, which are then digitized to form digital images. This requires advanced technology and precise calibration for accuracy.

2.2.2. Sensor Technologies

Different methods are used to create two-dimensional images. Single sensing elements like photodiodes can be moved along x and y axes, while sensor strips capture images linearly, requiring perpendicular movement for a complete image.

Sensor arrays, like CCDs in digital cameras, capture complete images without motion. They convert captured energy into an analog signal, which is then digitized to form a digital image. Specialized applications, like medical imaging, may use ring-configured sensor strips, requiring complex algorithms for image reconstruction.

2.2.3. Digital Image Formation/Representation

The core of digital image formation is the function f(x,y), which is determined by the illumination source i(x,y), and the reflectance r(x,y) from the scene.

In transmission-based imaging like X-rays, transmissivity replaces reflectivity. A digital image is essentially a matrix of numerical values, each representing a pixel. The process involves two steps:

- Sampling: Digitizing coordinate values.
- Quantization: Converting amplitude values into discrete quantities.

The resolution and quality of a digital image depend on:

- The number of samples and discrete intensity levels.
- The dynamic range of the imaging system, which affects appearance and contrast by defining the ratio of maximum measurable intensity to minimum detectable intensity.

Spatial resolution refers to the smallest detail that can be distinguished in an image, typically measured in line pairs or pixels per unit distance.

Intensity resolution, on the other hand, is about the smallest detectable change in intensity level. It's limited by hardware capabilities and quantized in binary increments, like 8 bits or 256 levels. However, perception of these changes is influenced by factors such as noise, saturation, and human vision capabilities.

2.2.4. Image techniques (Restorations, reconstructions, color processing)

Image restoration focuses on recovering a degraded image using knowledge about the degradation phenomenon. It often involves modelling the degradation process and applying inverse processes to regain the original image.

Image enhancement is a subjective process aimed at improving the visual quality of an image. It involves techniques to address issues like noise, which can arise during image acquisition or transmission. Advanced filters, including adaptive and non-adaptive ones, are commonly used to reduce noise.

Color is a powerful descriptor in image processing. It plays a role in object identification and recognition. Color image processing includes both pseudo-color and full-color processing. Pseudo-color processing assigns colors to grayscale intensities, while full-color processing uses actual color data from sensors.

2.2.5. Image Compression

In digital image compression, there are three main types of redundancies:

- **Coding redundancy:** This occurs when the distribution of intensity values in an image is uneven, leading to inefficient use of bits in encoding. Common values are encoded with the same number of bits as rare values, resulting in wasted bits. Ideally, more frequent values should be assigned shorter codes to minimize the number of bits used.
- **Spatial and temporal redundancy:** This arises from correlated pixel values within an image or across video frames. Similar patterns or information are repeated, leading to redundancy that can be exploited for compression.
- **Irrelevant information:** This includes data that is ignored by the human visual system or unnecessary for the image's purpose. Removing this irrelevant information can reduce the size of the image file without sacrificing perceived image quality.

Efficient coding in image compression involves considering event probabilities like intensity values. Techniques like run-length encoding reduce spatial redundancy in images with constant intensity lines, significantly compressing data, or addressing temporal redundancy in video sequences.

However, removing irrelevant information leads to irreversible loss through quantization. Information theory guides determining minimum data for accurate representation.

Image compression systems use **encoders and decoders**. Encoders eliminate redundancies through mapping (to reduce spatial/temporal redundancy), quantization (to discard irrelevant information), and symbol coding (assigning codes to quantizer output). Decoders reverse these processes, except for quantization. Standards file formats like JPEG and MPEG for storage. **Huffman coding efficiently removes** coding redundancy by prioritizing least probable symbols.

2.2.6. Pushing the limits of Imaging

We've developed various methods to visualize beyond the visible spectrum, using sensors to capture different wavelengths. Infrared, magnetic resonance, electron microscopy, and ultrasound are just a few examples.

These techniques have enabled us to explore the vastness of space and the intricacies of the microscopic world, from DNA structures to individual atoms. Scientists have even genetically modified animals to tag proteins with fluorescent markers for easier imaging.

This diversity in imaging methods has revolutionized our understanding of the universe and life itself, impacting everyday applications such as medical diagnostics and personal communication.

Ultimately, imaging tools serve as our eyes to perceive the universe, providing invaluable insights. This diversity in imaging is quite phenomenal. These optical tools have become the eyes through which we perceive the universe.

2.2.7. Imaging characteristics & Acquisition

Different image types may require different approaches, but not necessarily new neural network architectures. Pre-existing models can often be adapted or fine-tuned for specific tasks, sometimes with preprocessing to match the input the network was trained on.

Images acquired in different wavelengths can be treated as different color channels, simplifying analysis. However, technologies **like radar and ultrasound use polar grids**, where pixel size varies with distance from the center. In such cases, coordinate system adjustments or additional inputs may be needed for the model.

Additionally, **image acquisition can introduce biases**, such as measurement bias, which must be recognized and addressed to ensure model accuracy. Preprocessing techniques can help mitigate these biases and enhance model performance.

3. Computer Vision

3.1. Definition

Computer vision is the science and technology of enabling machines to perceive and understand visual data, leading to meaningful interpretations of the world. It encompasses acquiring, processing, analyzing, and interpreting visual information.

The field of computer vision has evolved through incremental advancements and interdisciplinary collaboration. A significant milestone was the widespread adoption of **deep learning** methods.

Unlike traditional approaches that relied on handcrafted features and domain knowledge, deep learning allows machines to automatically learn complex features from raw data, leading to more adaptive and sophisticated models.

3.2. A step forward in Image understanding

Image understanding is the process of making sense of the content of an image. It can be defined in three different levels:

- **Low-level processes** are primitive operations on images (i.e. image sharpening, changing the contrast). The input and the output are images.
- **Mid-level processes** include segmentation, description of objects, and object classification. The information is an image, but the result is attributes associated with the image. This could be done with a combination of image preprocessing and ML algorithms.
- **High-level processes** include making sense of the entirety of an image, i.e., recognition of a given object, scene reconstruction, and image-to-text. These are tasks typically associated with human cognition.

Image analysis primarily focuses on low and mid-level processes, while computer vision extends to mid- and high-level processes. This results in an overlap in the mid-level processes between image analysis and computer vision.

Combining a “preprocessing” part before moving on to a more robust model is usual. On the opposite side, sometimes, the layers of a neural network automatically perform tasks like these, eliminating the need for explicit preprocessing. Image analysis might act as a first exploratory data analysis step for those familiar with data science. Lastly, classical image analysis methods can also be used for data augmentation to improve the quality and diversity of training data for computer vision models.

3.3. Tasks overview

Some of the most popular tasks in computer vision are:

- Scene Recognition
- Object Recognition
- Object Detection
- Segmentation (instance, semantic)
- Tracking
- Dynamic Environment Adaptation
- Path Planning
- Image Captioning
- Image Classification
- Image Description
- Anomaly Detection
- Image Generation
- Image Restoration
- Autonomous Exploration
- Localization

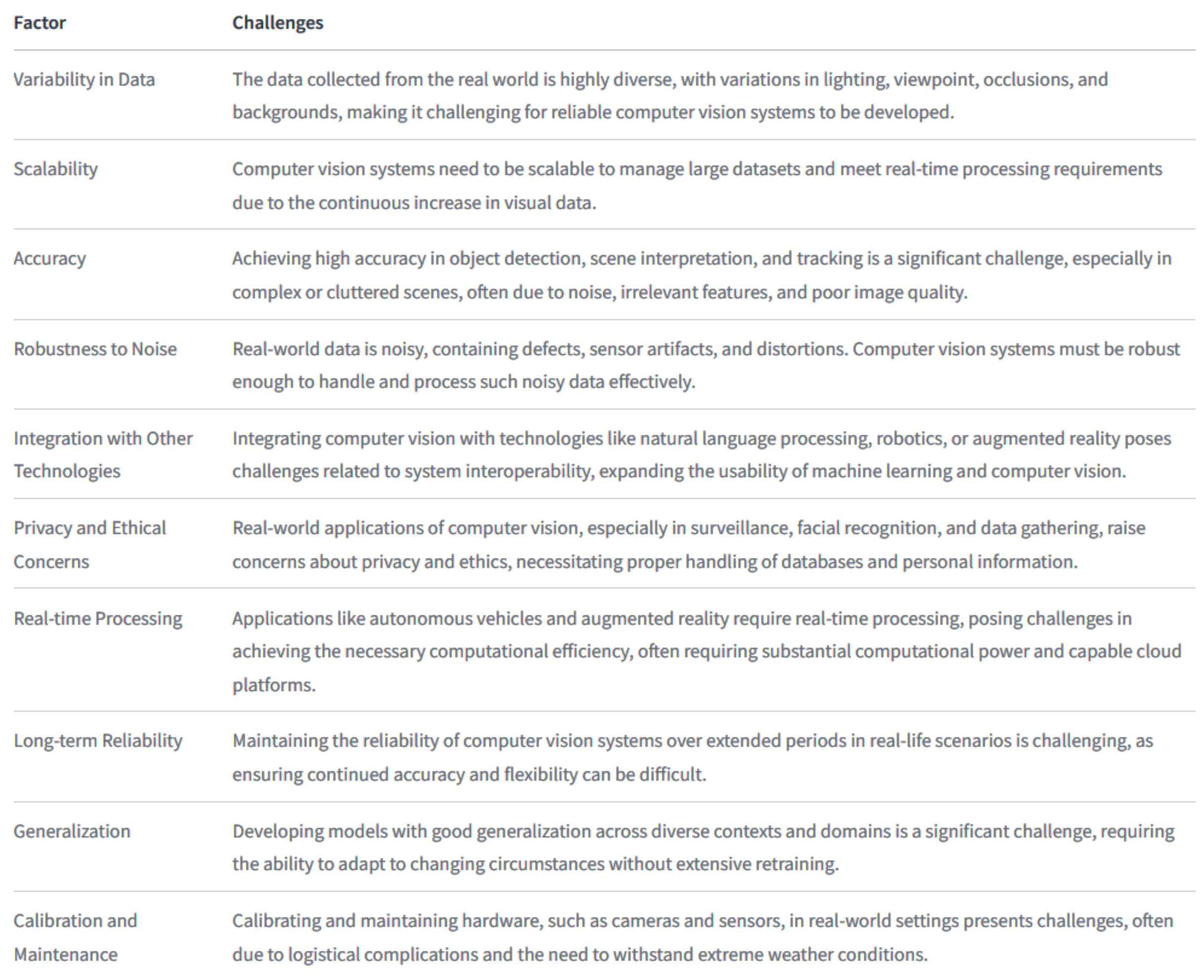
Task complexity in image analysis and computer vision depends more on the characteristics of the data being analyzed rather than the perceived difficulty of the task. Even seemingly basic tasks can become challenging in low-light conditions or with low-resolution images, requiring advanced techniques for image enhancement and machine learning. Lighting, occlusions, image resolution, and camera quality can greatly affect the task's complexity for a computer algorithm.

Computer vision applications, include:

- **Autonomous Vehicles:** Used for real-time decision-making by identifying objects, pedestrians, and traffic signs.
- **Retail and E-commerce:** Employs object recognition for product identification and recommendation systems.
- **Quality Control in Assembly Lines:** Detects defects, automates inspections, and provides real-time feedback for manufacturing processes.
- **Medical Image Analysis:** Assists in diagnosing diseases, segmenting and detecting anomalies in medical images, aiding in treatment planning and drug development…

3.4. Challenges for CV Systems

Computer Vision systems face a multitude of challenges that arise from the complexity of processing visual information in real-world scenarios, ranging from poor data quality, privacy, and ethical concerns along with other concerns, which are mentioned in the table below:

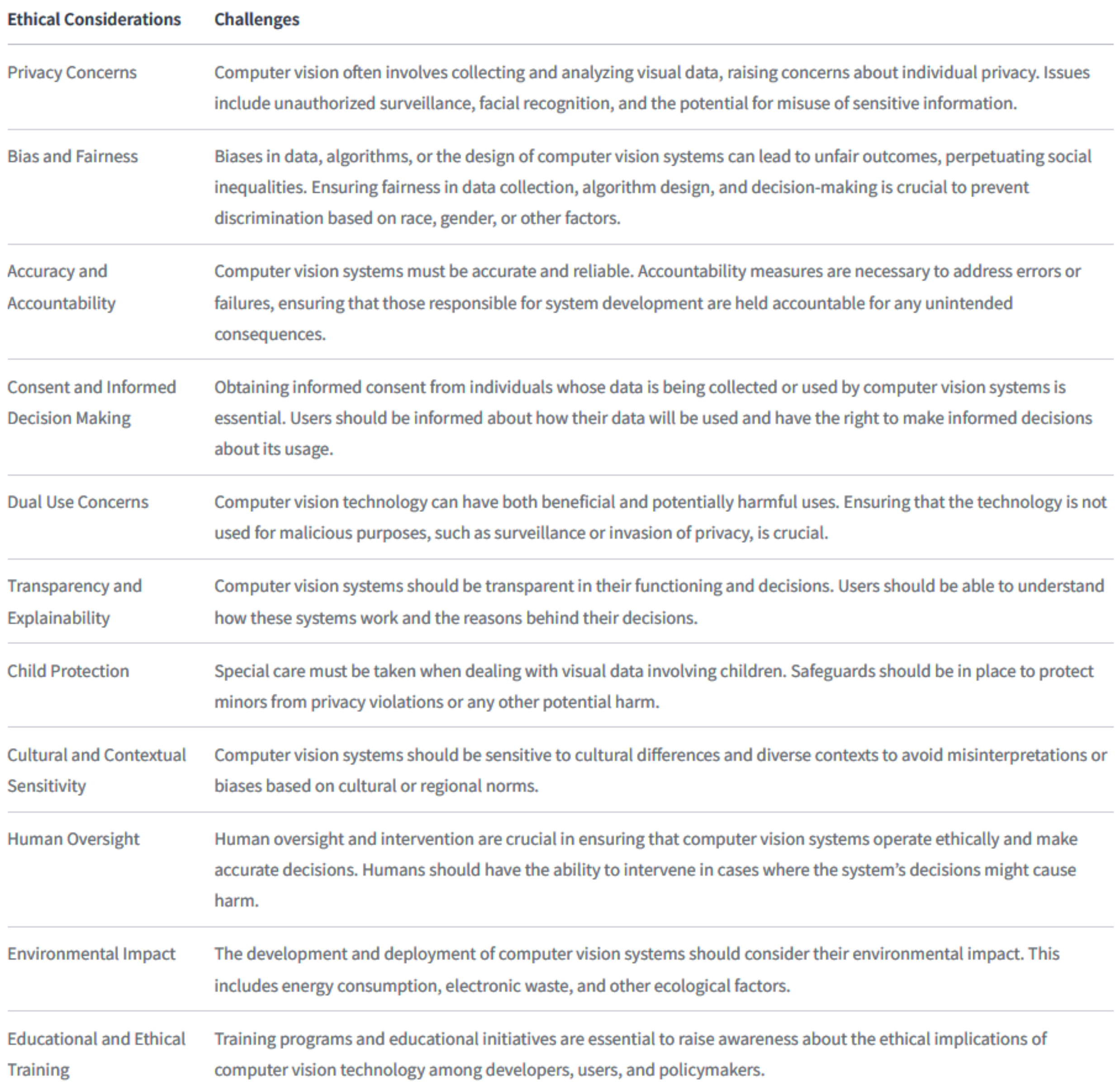


Factor	Challenges
Variability in Data	The data collected from the real world is highly diverse, with variations in lighting, viewpoint, occlusions, and backgrounds, making it challenging for reliable computer vision systems to be developed.
Scalability	Computer vision systems need to be scalable to manage large datasets and meet real-time processing requirements due to the continuous increase in visual data.
Accuracy	Achieving high accuracy in object detection, scene interpretation, and tracking is a significant challenge, especially in complex or cluttered scenes, often due to noise, irrelevant features, and poor image quality.
Robustness to Noise	Real-world data is noisy, containing defects, sensor artifacts, and distortions. Computer vision systems must be robust enough to handle and process such noisy data effectively.
Integration with Other Technologies	Integrating computer vision with technologies like natural language processing, robotics, or augmented reality poses challenges related to system interoperability, expanding the usability of machine learning and computer vision.
Privacy and Ethical Concerns	Real-world applications of computer vision, especially in surveillance, facial recognition, and data gathering, raise concerns about privacy and ethics, necessitating proper handling of databases and personal information.
Real-time Processing	Applications like autonomous vehicles and augmented reality require real-time processing, posing challenges in achieving the necessary computational efficiency, often requiring substantial computational power and capable cloud platforms.
Long-term Reliability	Maintaining the reliability of computer vision systems over extended periods in real-life scenarios is challenging, as ensuring continued accuracy and flexibility can be difficult.
Generalization	Developing models with good generalization across diverse contexts and domains is a significant challenge, requiring the ability to adapt to changing circumstances without extensive retraining.
Calibration and Maintenance	Calibrating and maintaining hardware, such as cameras and sensors, in real-world settings presents challenges, often due to logistical complications and the need to withstand extreme weather conditions.

3.5. Ethical Considerations

Ethical considerations in computer vision have been present since its inception. In today's digital society, fairness in model behavior is crucial, aiming to avoid discrimination and bias.

However, evaluating fairness is complex, lacking a mathematical metric, and bias can emerge at various stages of model development. Efforts include systematic reporting of risks and biases in model, but ethical considerations remain a significant focus.



Ethical Considerations	Challenges
Privacy Concerns	Computer vision often involves collecting and analyzing visual data, raising concerns about individual privacy. Issues include unauthorized surveillance, facial recognition, and the potential for misuse of sensitive information.
Bias and Fairness	Biases in data, algorithms, or the design of computer vision systems can lead to unfair outcomes, perpetuating social inequalities. Ensuring fairness in data collection, algorithm design, and decision-making is crucial to prevent discrimination based on race, gender, or other factors.
Accuracy and Accountability	Computer vision systems must be accurate and reliable. Accountability measures are necessary to address errors or failures, ensuring that those responsible for system development are held accountable for any unintended consequences.
Consent and Informed Decision Making	Obtaining informed consent from individuals whose data is being collected or used by computer vision systems is essential. Users should be informed about how their data will be used and have the right to make informed decisions about its usage.
Dual Use Concerns	Computer vision technology can have both beneficial and potentially harmful uses. Ensuring that the technology is not used for malicious purposes, such as surveillance or invasion of privacy, is crucial.
Transparency and Explainability	Computer vision systems should be transparent in their functioning and decisions. Users should be able to understand how these systems work and the reasons behind their decisions.
Child Protection	Special care must be taken when dealing with visual data involving children. Safeguards should be in place to protect minors from privacy violations or any other potential harm.
Cultural and Contextual Sensitivity	Computer vision systems should be sensitive to cultural differences and diverse contexts to avoid misinterpretations or biases based on cultural or regional norms.
Human Oversight	Human oversight and intervention are crucial in ensuring that computer vision systems operate ethically and make accurate decisions. Humans should have the ability to intervene in cases where the system's decisions might cause harm.
Environmental Impact	The development and deployment of computer vision systems should consider their environmental impact. This includes energy consumption, electronic waste, and other ecological factors.
Educational and Ethical Training	Training programs and educational initiatives are essential to raise awareness about the ethical implications of computer vision technology among developers, users, and policymakers.

4. Pre-processing for Computer Vision

4.1.1. Operations in Digital Image Processing:

Various techniques can be use in image manipulation such as logical, statistical, geometrical, mathematical, and transform operations.

Each category encompasses different techniques, such as morphological operations under logical operations or **fourier transforms** and **principal component analysis (PCA)** under transforms. Morphological operations involve using structuring elements to analyze pixel neighborhoods and generate images of the same size.

4.1.2. Mathematical Tools in Image Processing:

Set theory is crucial for operations on binary images, where pixels are categorized as foreground or background.

Intensity transformations and spatial filtering manipulate pixel values to achieve noise reduction and other enhancements.

4.1.3. Spatial Filtering Techniques and Image Enhancement:

Spatial filtering modifies images by altering each pixel's value based on neighboring pixels, using techniques like linear filters for blurring or sharpening.

4.1.4. Data Augmentation:

Data augmentation enriches training datasets for Convolutional Neural Networks (CNNs) by creating modified versions of data, enhancing model accuracy, generalization and preventing overfitting.

It involves applying transformations like flipping, cropping, and filtering to existing data, or generating entirely new data using techniques like Deep Neural Networks (DNNs) and Generative Adversarial Networks (GANs). Data augmentation is crucial for various data types, including images, audio, and text, and is instrumental in fields like healthcare, self-driving cars, and natural language processing.

4.2. Feature Description

Features are attributes that models learn and use to perform a given task. Recognizing these features is essential to ensure model accuracy.

Features, or attributes or variables, can be diverse, ranging from numerical values and categories to more complex structures like images or text. Some ways to represent features for computer vision tasks are:

- Numerical features: arrays/list, tensors
- Categorical features: dict/list, one hot encoding
- Image features: pixel values, extracted by CNN model.

4.2.1. What makes a good descriptor/feature extractor?

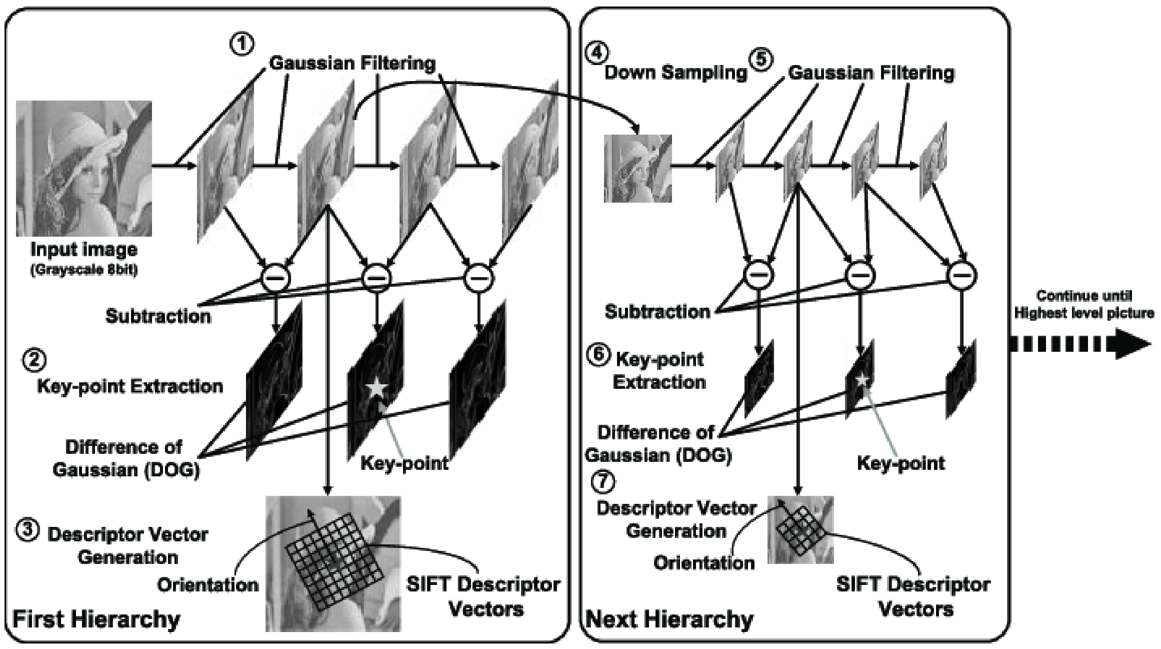
A good descriptor in computer vision is a set of features that effectively represent key information about an image. Here are some aspects that contribute to making a good descriptor:

- **Invariant to Transformation**: Descriptors should remain consistent despite changes like rotation, scaling, or lighting, ensuring reliable recognition across varied conditions.
- **Distinctiveness**: Effective descriptors must capture unique details, enabling discrimination between different objects or regions within an image.

- **Dimensionality**: Good descriptors often have a manageable size, conveying enough information without being excessively large. Balancing dimensionality is crucial for efficiency in processing and storage.
- **Locality**: Descriptors often identify local features within an image. Local descriptors focus on specific regions or keypoints and describe the characteristics of these areas, enabling matching and recognition of similar regions across different images.
- **Repeatability**: Descriptors should consistently represent the same object or scene, even amidst noise or minor variations, ensuring reliability in recognition.
- **Compatibility with Matching Algorithms**: Descriptors should align with the requirements of matching algorithms, whether based on distance metrics or machine learning, ensuring effective correspondence between images.
- **Computational Efficiency**: Descriptors must be computationally feasible for quick processing, particularly in real-time applications like robotics or autonomous vehicles.
- **Adaptability**: Descriptors that can adapt or learn from data variations enhance effectiveness, particularly in dynamic environments where object or scene characteristics evolve over time.
- **Noise Robustness**: Descriptors should handle image noise without compromising feature representation, maintaining accuracy in diverse conditions.

4.2.2. Scale invariant feature transform (SIFT)

Widely used algorithm in computer vision and image processing for detecting and describing local features in images.



This how SIFT works:

1. **Detecting potential interest points (Scale Space extrema detection):** It searches for locations where the difference of Gaussian function reaches a peak or trough across space and scale. These keypoint locations are considered stable under various scale changes.

2. **Keypoint Localization:** Once potential keypoints are identified, SIFT refines their positions to sub-pixel accuracy and discards low-contrast keypoints and keypoints on edges to ensure accurate localization.
3. **Orientation Assignment:** After identifying keypoints, SIFT computes a dominant orientation for each one by analyzing local image gradient directions. This ensures that the descriptor remains invariant to image rotation, as it aligns the keypoint with its most prominent orientation, allowing for consistent feature extraction regardless of the image's orientation.
4. **Description Generation:** computes a descriptor for each keypoint region, which encapsulates information about the local image gradients near the keypoint. This descriptor serves as a compact representation, capturing the essential characteristics of the image patch surrounding the keypoint.
5. **Descriptor Matching:** Finally, these descriptors are used for matching keypoints between different images. The descriptors from one image are compared to those in another image to find correspondences.

Note: SIFT is very valuable for applications like object recognition, image stitching, and 3D reconstruction.

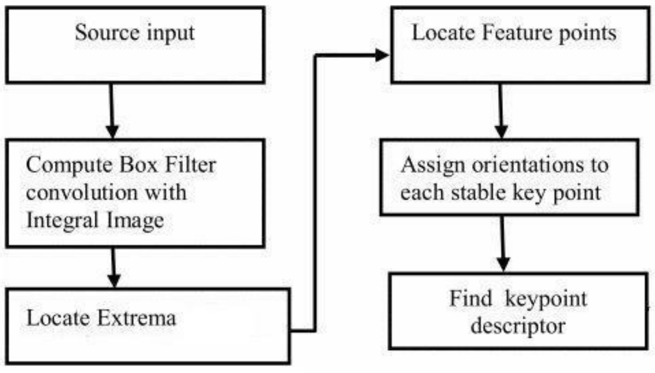
Additional ref:
- *Introduction to SIFT (Scale-Invariant Feature Transform)*
- *What is SIFT*
- *SIFT*
- *SIFT summary in 5min*

4.2.3. Speeded Up Robust features (SURF)

SURF is another very valuable algorithm for describing and detecting local features in images.

The key strengths of SURF lie in its computational efficiency, which is achieved through the use of integral images and **Haar wavelet approximations** while maintaining robustness to scale, rotation, and illumination changes.

This makes SURF suitable for **real-time applications** where speed plays a crucial part in object detection, tracking, and image stitching.


Source input
Compute Box Filter convolution with Integral Image
Locate Extrema
Locate Feature points
Assign orientations to each stable key point
Find keypoint descriptor

Workflow of SURF is given below:

1. **Integral images:** SURF utilizes integral images, precomputed representations of the original image. They allow fast calculations of rectangular area sums within an image, enabling a quicker feature computation.
2. **Blob detection:** Like other feature detection algorithms, SURF starts by identifying potential interest points or keypoints in the image. It uses a **Hessian matrix** to detect blobs or regions that exhibit significant variations in intensity in multiple directions and scales. These regions are potential keypoints.
3. **Scale Selection:** It determines the scale of the keypoints by identifying regions with significant changes in scale-space. It analyzes the determinant of the Hessian matrix across different scales to find robust keypoints at multiple scales.
4. **Orientation Assignment:** For each detected keypoint, SURF assigns a dominant orientation. This is done by calculating **Haar wavelet responses** in different directions around the keypoint's neighborhood. Unlike SIFT, SURF uses a set of rectangular filters (Haar wavelets) applied to subregions of the keypoint's neighborhood. The responses of these filters are used to create a feature vector representing the keypoint.
5. **Descriptor Matching:** The generated descriptors are then used to match key points between different images. Matching involves comparing the feature vectors of keypoints in one image to those in another image to find correspondences.

Additional ref:

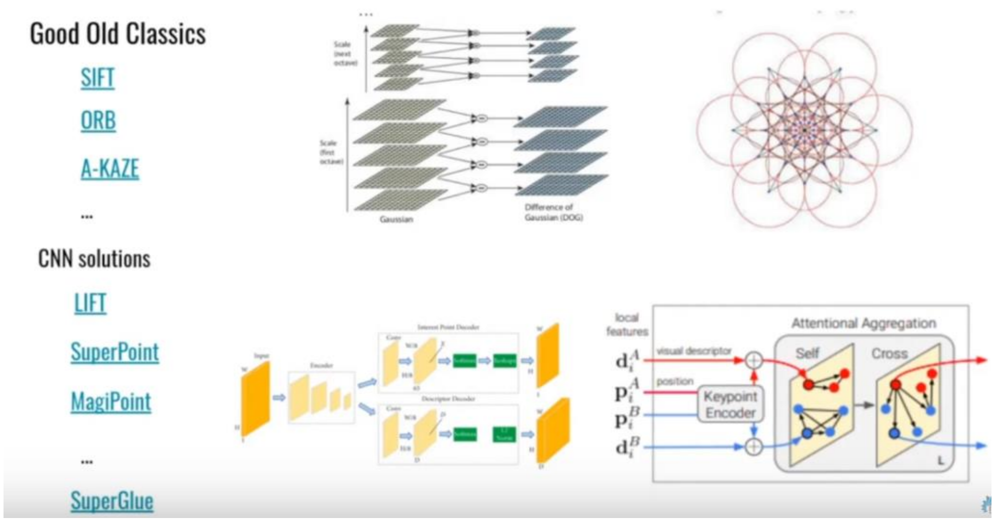
- *OpenCV Tutorial - Introduction to SURF (Speeded-Up Robust Features)*
- *Journal Paper - Feature Extraction Using SURF Algorithm for Object Recognition*
- *SURF explained*

4.3. Feature Matching

Feature matching involves comparing key attributes in different images to find similarities. Feature matching is useful in many computer vision applications, including scene understanding, image stitching, object tracking, and pattern recognition.

Bruce-Force search and LoFRT are to approaches for features matching. In traditional feature matching, like **brute-force search**, you systematically compare every possible combination until you find the best match, which can be time-consuming.

LoFTR, however, takes a different approach. Instead of comparing pixels directly, it identifies key points or features in each image. LoFTR then evaluates how well these features align, assigning a similarity score to indicate the quality of the match. What sets LoFTR apart is its ability to handle changes in rotation, scale, and other transformations, ensuring robust matching even under varying conditions like different lighting or perspectives. This makes LoFTR valuable for tasks such as image stitching, where seamlessly combining multiple images relies on accurately identifying and connecting common features.



Good Old Classics

SIFT

ORB

A-KAZE

...

CNN solutions

LIFT

SuperPoint

MagiPoint

...

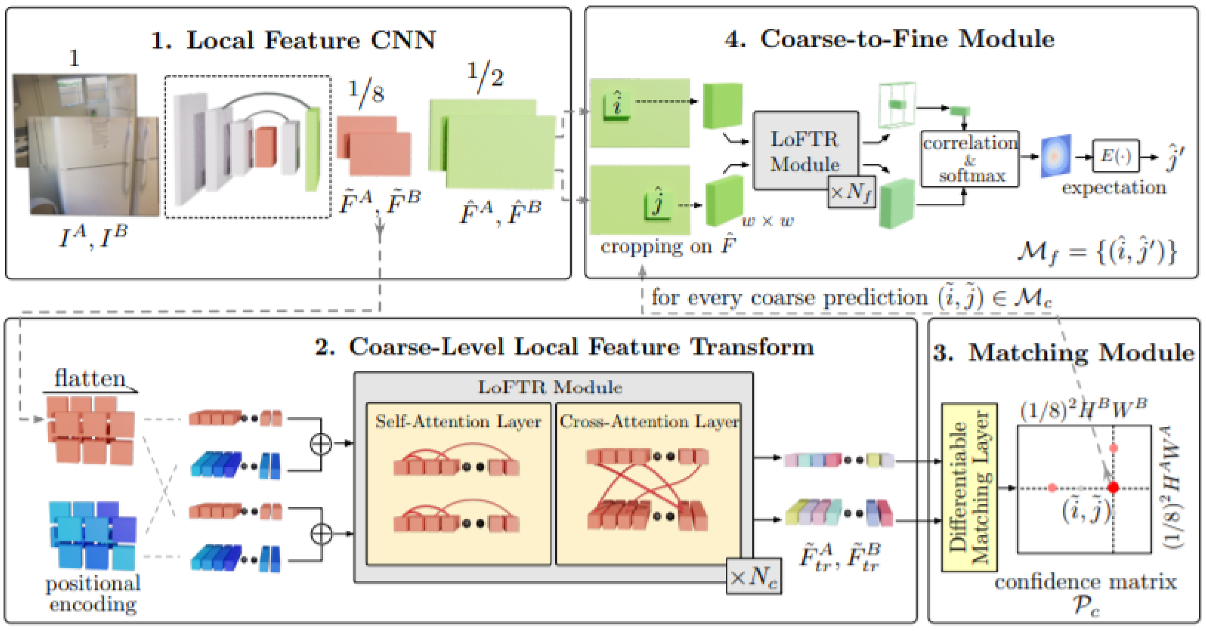
SuperGlue

4.3.1. Local Feature Matching with Transformers (LoFTR)

LoFTR (Local Feature TRansformer) revolutionizes image feature matching by introducing a framework that prioritizes dense matching over sequential processing.

Unlike traditional methods, LoFTR leverages self and cross attention layers from Transformer models to obtain feature descriptors from both images. This global receptive field allows LoFTR to excel in low-texture areas where traditional feature detectors struggle.

Additionally, the pre-trained model is adept at distinguishing between indoor and outdoor scenes, further enhancing its performance. As a result, LoFTR significantly outperforms other state-of-the-art methods in image feature matching.



https://analyticsindiamag.com/introduction-to-local-feature-matching-using-loftr/

LoFTR works as following steps:

1. **Feature Extraction**: A CNN extracts feature maps (Feature A and Feature B) and fine-level feature maps from image pairs A and B.
2. **Feature Preparation**: The feature maps are flattened into 1-D vectors and augmented with positional encoding to describe the spatial orientation of objects in the input images.
3. **Local Feature Transformer (LoFTR)**: The transformed features are processed by the LoFTR module, utilizing self and cross attention layers from Transformer models to obtain enhanced feature representations.
4. **Matching Layer**: A differentiable matching layer is employed to compare the transformed features and generate a confidence matrix indicating potential matches between features.
5. **Match Selection**: Matches are selected based on a confidence threshold and mutual nearest-neighbor criteria, yielding coarse-level match predictions.
6. **Refinement**: For each selected coarse prediction, a local window is cropped from the fine-level feature map. Coarse matches are then refined from this window to a sub-pixel level, producing final match predictions.

The Convolution Layers and the Transformers assume that the objective is to establish a connection between the Left and Right elements between two images to extract their joint feature representation. Due to the local connectivity of convolutions, many convolution layers are stacked together to achieve such a connection.

The global receptive field of the Transformers enables this connection to be established through only one attention layer, which stores the previously learned attributes as well.

(**For visualizing** the attention weights and transformed dense features, **Principal Component Analysis** is used to reduce the dimension of the transformed features A and B. In turn, it visualizes the results with RGB colors).

The visualization from attention weights demonstrates that the features in indistinctive or regions of low texture from the image can aggregate local and global context information through self-attention and cross-attention. Using such, the feature visualization with PCA further enables LoFTR to learn positional features for better representation.

Additional ref:

- *FLANN Github*
- *Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images*
- *ORB (Oriented FAST and Rotated BRIEF) tutorial*
- *Kornia tutorial on Image Matching*
- *LoFTR Github*
- *OpenCV Github*
- *OpenCV Feature Matching Tutorial*
- *OpenGlue: Open Source Graph Neural Net Based Pipeline for Image Matching*

5. Convolutional Neural Networks

5.1. Introduction

Convolutional Neural Networks (CNNs) are a key component of deep learning, particularly in image processing tasks. CNNs learn to automatically discover and optimize filters to extract relevant information from images. By convolving images with these learned **filters,** CNNs generate **feature maps** that capture important patterns and structures.

Pooling is a technique used in CNNs to reduce the dimensionality of feature maps while retaining the most important information. It involves summarizing groups of pixels into a single value, typically by taking the **maximum (max pooling) or average (average pooling**) pixel value within each group. This helps in reducing the number of parameters, making the model more efficient, and improving its ability to generalize to variations in input images.

In a CNN architecture, convolutional layers are followed by pooling layers to progressively extract and condense features. These layers are typically interspersed with activation functions such as ReLU to introduce non-linearity and increase the model's expressive power.

Finally, fully connected layers and output layers are used to classify or regress on the extracted features.

5.1.1. Backpropagation in CNN

In Convolutional Neural Networks (CNNs), **backpropagation** is used to optimize the kernel values, which are the weights of the network. These kernels are convolved with the input images to extract features, and the goal is to learn the optimal kernels that can effectively distinguish between different classes (e.g., cats and dogs).

During backpropagation, the **network adjusts the kernel weights** based on the error between the predicted output and the actual labels. By propagating this error backward through the network, the gradients of the loss function with respect to the kernel weights are computed. These gradients are then used to update the weights via optimization algorithms such as stochastic gradient descent (SGD) or its variants.

They automatically learn and optimize kernel values, allowing them to adapt to the unique characteristics of different images and datasets. This adaptability enables CNNs to outperform generic filters used in traditional image processing, which may not be effective for all images due to their variability.

5.1.2. Parameter sharing vs Sparse interactions.

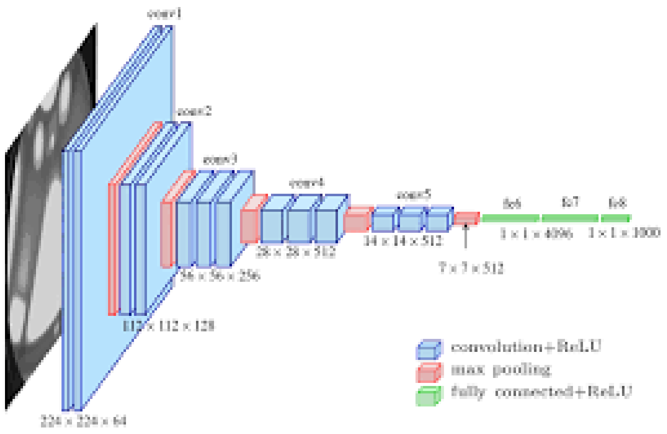
In CNN, we convolve with the same filter across all pixels/channels/images, this is much more efficient than going through an image with a dense neural network. This is called “weight tying” and those weights are called “tied weights”.

In densely connected neural networks (DNN), the input is the whole piece of data at once, meanwhile in convnets, we have smaller kernels that we use to extract features. This is called sparse interaction, and it helps us use less memory.

5.2. Deep Convolutional Networks Architectures

5.2.1. VGG

Developed in 2014, VGG demonstrated significant improvements over the past models at that time- to be specific 2014 Imagenet challenge also known as ILSVRC.



Advantages:

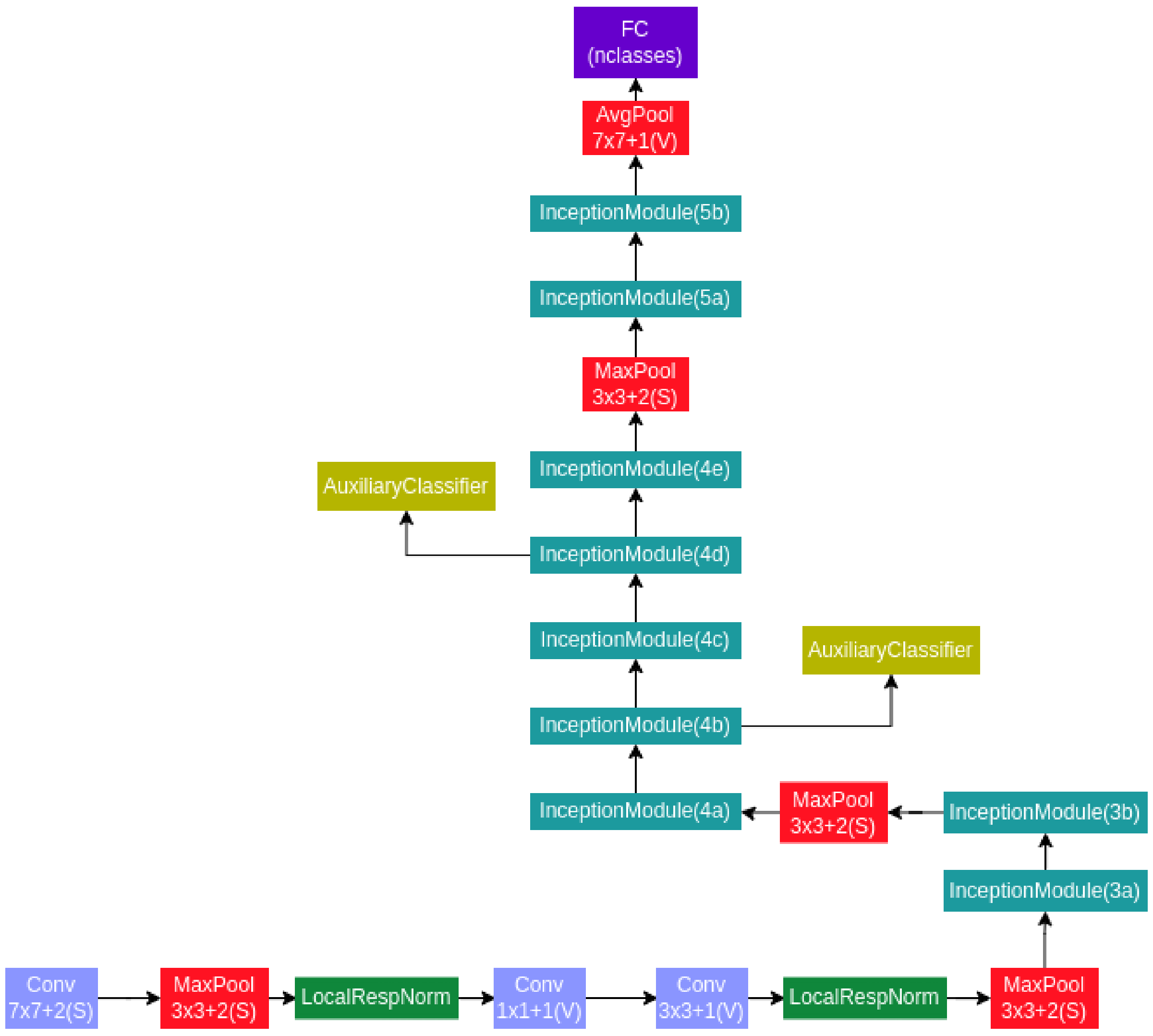
- **Multiple small (3X3)** receptive field filters with ReLU activation instead of one large (7X7 or 11X11) filter lead to better learning of complex features. Smaller filters also mean fewer parameters per layer, with additional nonlinearity introduced in between.
- **Consistency and simplicity** of the VGG network make **it easier to scale or modify** for future improvements.
- **Multiscale training and inference.** Each image was trained in multiple rounds with varying scales to ensure similar characteristics were captured at different sizes.

5.2.2. GoogleNet

The **Inception architecture**, a convolutional neural network (CNN) tailored for computer vision tasks like classification and detection, stands out for its efficiency.

It is notably more compact than its predecessors, being 9 times smaller than AlexNet and 22 times smaller than VGG16. **Prior models like AlexNet and VGG emphasized the benefits of deeper network structures but faced challenges such as increased computational complexity, overfitting,** and the vanishing gradient problem.

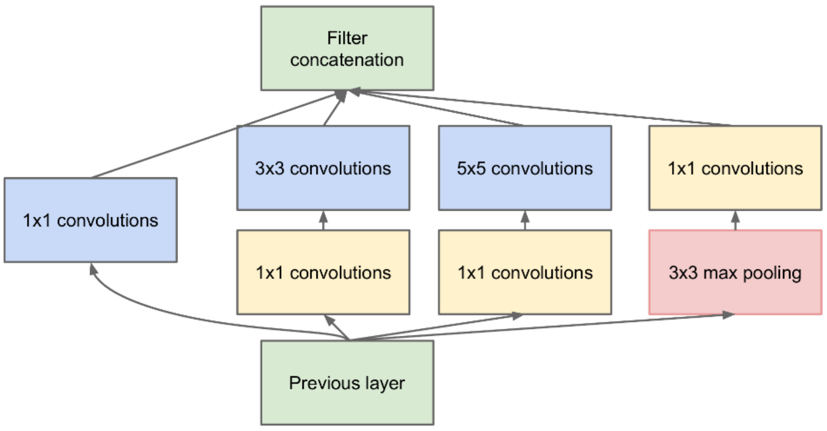
The Inception architecture addresses these issues by enabling the training of complex CNNs with fewer floating-point parameters, thereby offering a more efficient and effective solution for image-related tasks.



The **Inception Module**, inspired by the concept of *Network In Network*, introduces a more complex building block compared to traditional convolution layers.

It applies convolutions of different kernel sizes (1x1, 3x3, and 5x5) in parallel, along with a max pooling operation, to extract features at multiple scales. However, this approach can increase the number of parameters significantly, especially as input feature size grows. To mitigate this, a 1x1 convolution is applied to reduce feature dimension before the 3x3 and 5x5 convolutions.

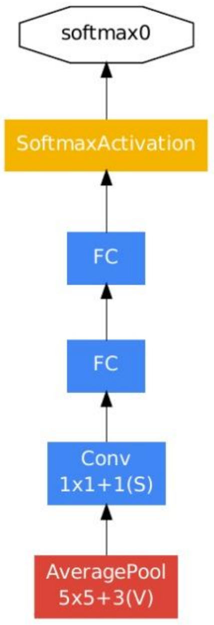
Additionally, another 1x1 convolution is added after max pooling to further reduce output features. Additionally, ReLU activation is applied after each 1x1 convolution to increase non-linearity and complexity.



INCEPTION MODULE

Average Pooling: In traditional networks (AlexNet, VGG), the final layers typically consist of fully connected layers, which contribute to a significant portion of the network's parameters. However, with the introduction of the Inception block, fully connected layers are not necessary.

Instead, simple average pooling along the spatial dimensions is sufficient, as proposed in the Network in Network paper. While GoogLeNet included one fully connected layer, it still reduced the reliance on such layers, resulting in only 15% of parameters in the fully connected layers. This approach not only reduces computational complexity but also maintains high accuracy.



AUXILIAR CLASSIFIER

Auxiliary Classifiers: With the reduction of parameters in the network through techniques like 1x1 convolutions and average pooling, it becomes feasible to add more layers and deepen the network. However, deeper networks can suffer from the vanishing gradient problem, where gradients diminish as they propagate backward through the layers.

To address this, auxiliary classifiers are introduced. These classifiers branch out from intermediate layers of the network and contribute their loss, albeit with less weight, to the total loss during training. This ensures that gradients of sufficient magnitude reach the initial layers of the network, mitigating the vanishing gradient problem and aiding in more effective training.

5.2.3. MobileNet

MobileNet employs depthwise separable convolutions, a more efficient alternative to standard convolutions. MobileNet utilizes depthwise separable convolutions, which break down the convolution process into two steps: depthwise convolution and pointwise convolution. This approach significantly reduces parameters and computational complexity, making it suitable for mobile devices.

This method splits the computation into two steps:

1. **Depthwise Convolution**: This step applies a separate convolutional operation to each channel of the input, effectively capturing spatial information within each channel independently.
2. **Pointwise Convolution**: Following the depthwise convolution, a 1x1 convolution, known as pointwise convolution, is applied to combine the information across channels, effectively capturing cross-channel correlations.

Depthwise convolutions analyze each input channel separately using small filters, reducing the number of channels while maintaining spatial dimensions. Pointwise convolutions then combine these channels to produce the final output. In contrast to regular convolutions, which use larger kernels to analyze groups of pixels simultaneously, MobileNet's depthwise separable convolutions focus on individual pixels.

By separating spatial and cross-channel computations, depthwise separable convolutions significantly reduce the computational cost while maintaining performance, making MobileNet suitable for resource-constrained devices like mobile phones.

Furthermore, MobileNet incorporates channel-wise linear bottleneck layers, which further reduce parameters and computational cost while maintaining accuracy. These layers consist of depthwise convolution, batch normalization, and activation functions like ReLU, addressing issues like the vanishing gradient problem and ensuring non-linearity for effective learning.

Key Differences

- **Area of Focus:** Normal convolutions analyze a set of pixels together to understand patterns, whereas 1x1 convolutions focus on individual pixels, combining the information from different channels.
- **Purpose:** Normal convolutions are used for detecting patterns and features in an image. In contrast, 1x1 convolutions are mainly used to alter the channel depth, helping in adjusting the complexity of the information for subsequent layers in a neural network for efficiency in weak devices.

5.2.4. ResNet (Residual Network)

As neural networks grew deeper, they were expected to perform better due to the enriched feature extraction seen in models like VGG16 and VGG19. However, a new problem emerged: **the degradation problem**, where deeper networks suffered from higher training and test errors despite their added layers. This wasn't due to overfitting but rather because the added layers didn't approximate the identity function well.

ResNet introduced **residual connections**, which addressed this issue and significantly improved accuracy compared to previous architectures. ResNet's building blocks designed as identity functions, preserve input information while enabling learning. This approach ensures efficient weight optimization and prevents degradation as the network becomes deeper.

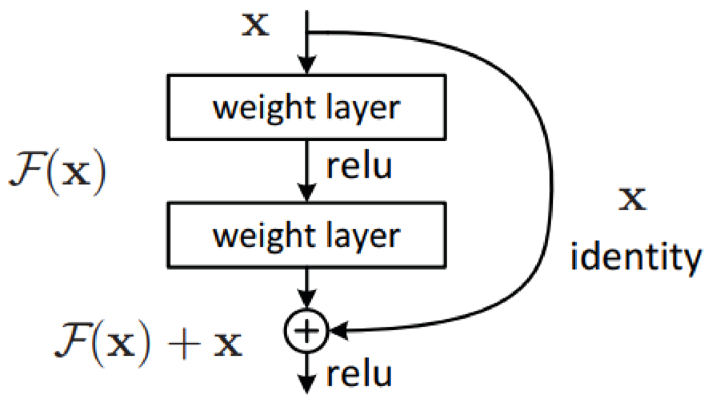


Figure 2. Residual learning: a building block.

ResNet combines a plain network with these **shortcuts,** resulting in improved performance. To ensure compatibility for adding outputs and inputs, ResNet uses techniques like zero-padding shortcuts and projection shortcuts. In deeper ResNet versions, bottleneck building blocks are used to manage parameter complexity while enabling deeper learning.

Additional refs:

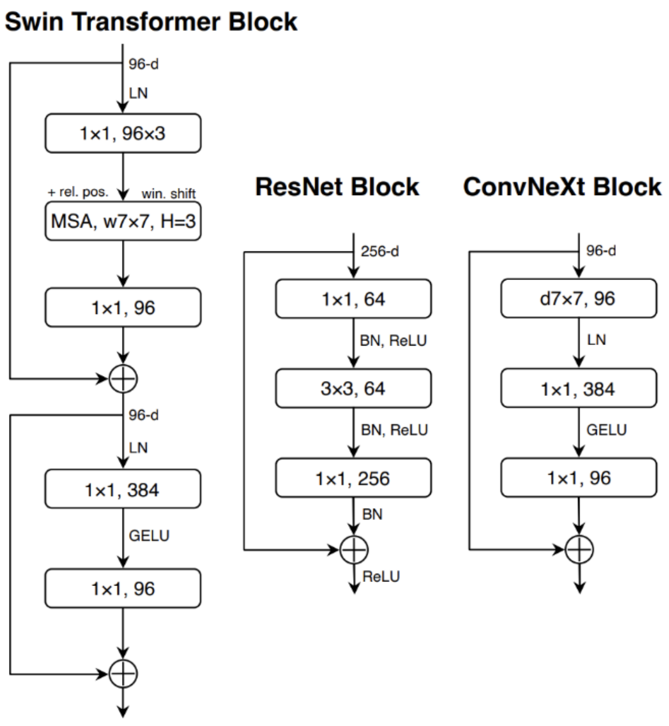
- *PyTorch docs*
- *ResNet: Deep Residual Learning for Image Recognition*
- *Resnet Architecture Source: ResNet Paper*
- *Hugging Face Documentation on ResNet*

5.2.5. ConvNext - A ConvNet for the 2020s (2022)

Vision Transformers (ViTs) have emerged as the new state-of-the-art for image recognition, surpassing pure CNN models. Interestingly, recent research suggests that CNNs can benefit from adopting design choices inspired by ViTs.

ConvNext represents a notable advancement in pure convolution models by integrating techniques from ViTs, achieving comparable accuracy and scalability results.

The author of the **ConvNeXT** starts building the model with a regular **ResNet (ResNet-50),** then modernizes and improves the architecture step-by-step to imitate the hierarchical structure of Vision Transformers.



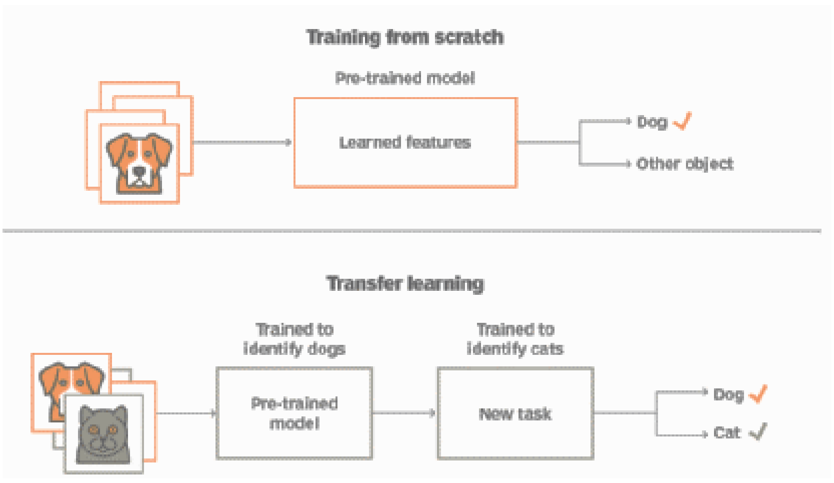
The key improvements are:

- **Training Techniques** : adopts training techniques from DeiT and Swin Transformers, including extending epochs to 300, using AdamW optimizer, and employing regularization methods like Stochastic Depth and Label Smoothing
- **Macro Design:** adjusting stage compute ratios to enhance model accuracy and replacing the stem with a Patchify convolutional layer for improved downsampling, resulting in a slight accuracy improvement.
- **ResNeXt-ify:** integrates ResNeXt principles, leveraging depthwise convolutions to balance FLOPs and accuracy.
- **Inverted Bottleneck**
- **Large Kernel Sizes**
- **Micro Design :** replacing ReLU with GELU, adjusting normalization layers, and adding a separate downsample layer…

5.3. Transfer Learning

5.3.1. Definition

Transfer learning, used in machine learning, is **the reuse of a pre-trained model on a new problem**. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another.



5.3.2. Transfer Learning and Fine-tuning

Transfer learning involves taking a **pre-trained model** and using it as a starting point for a new task. This typically involves freezing the weights of most or all layers in the pre-trained model and only training the final task-specific layers.

Fine-tuning, on the other hand, is a more flexible approach where we may unfreeze and retrain some of the earlier layers in addition to the final task-specific layers, especially if the initial results are not satisfactory. This allows for more adjustment and adaptation to the new task, leveraging both generic and task-specific features learned by the model.

5.3.3. Transfer Learning and Self-Training

Transfer learning involves using knowledge gained from training one model on a specific task and applying it to a related task. It typically leverages pre-trained models and labeled data to improve performance on a new task with limited labeled data.

On the other hand, **Self-training** involves training a model on both labeled and unlabeled data, allowing it to learn from the data itself without direct supervision.

While **transfer learning adapts existing knowledge** to a new task, **self-training focuses on learning from the available data,** which can be particularly useful when labeled data is scarce. Both approaches can be complementary and effective in improving model performance, depending on the specific task and data availability.

Ref: *https://arxiv.org/abs/2006.06882*

6. Vision Transformers

As the Transformers architecture scaled well in NLP, the same architecture was applied to images by creating small patches of the image and treating them as tokens. The result was a Vision Transformer (Vision Transformers).

6.1.1. CNN vs Vision Transformers

Inductive bias in machine learning refers to the assumptions guiding a model's predictions. CNNs rely on biases like translational equivariance and locality, whereas Vision Transformers lack these biases. However, Vision Transformers excel due to scalability and extensive training data, which compensate for the absence of these biases.

Training Vision Transformers requires a vast dataset to mitigate inherent biases. However, it's impractical for most to access millions of images necessary for optimal performance.

6.1.2. CNN vs Vision Transformers

Transfer learning involves leveraging features learned by pre-trained Vision Transformers on large datasets to improve model performance on smaller datasets. By freezing most weights and training only specific layers, significant improvements can be achieved with less training time and GPU consumption.

"Fine-tuning" the entire pre-trained model with a lower learning rate can also adapt it to perform well on datasets with domains different from the pre-trained model's dataset (very different datasets).

Additional ref:

- *Original Vision Transformers Paper:* An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale *Paper*
- *Swin Transformers Paper:* Swin Transformer: Hierarchical Vision Transformer using Shifted Windows *Paper*
- *A systematic empirical study in order to better understand the interplay between the amount of training data, regularization, augmentation, model size and compute budget for Vision Transformers:* How to train your Vision Transformers? Data, Augmentation, and Regularization in Vision Transformers *Paper*

6.2. Architectures of Transformers in Computer Vision

6.2.1. Vision Transformer (ViT)

ViT decomposes each image into a sequence of tokens (i.e. non-overlapping patches) with a fixed length and then applies multiple standard Transformer layers, consisting of Multi-head Self Attention and Position-wise Feed-forward module (FFN) to model global relations for classification.

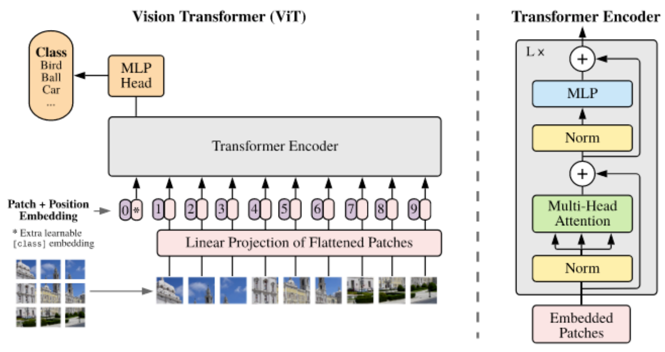
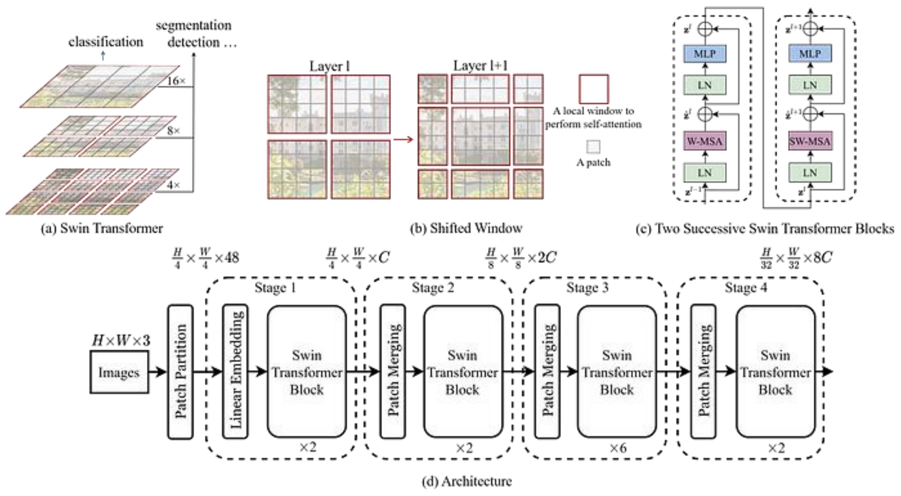


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable "classification token" to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

6.2.2. Swin Transformer



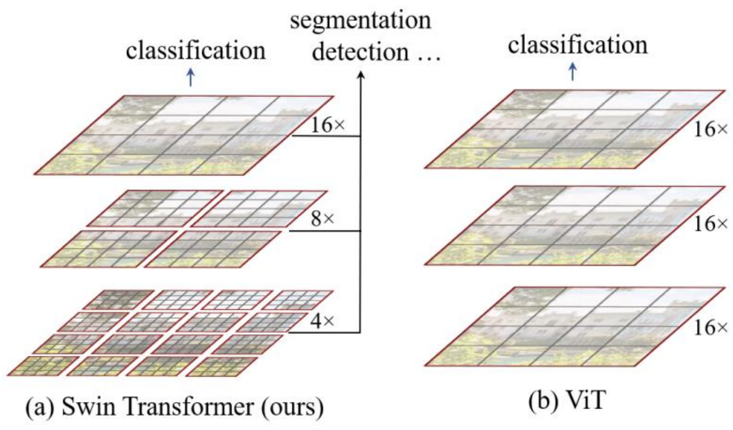
The **Swin Transformer**, revolutionizes computer vision with its hierarchical backbone architecture that employs shifted windows instead of sliding windows, reducing computational complexity while maintaining performance. It's optimized for low latency and high efficiency, making it ideal for tasks like image classification.

A **backbone**, in terms of deep learning, is a part of a neural network that does feature extraction. Additional layers can be added to the backbone to do a variety of vision tasks. Hierarchical backbones have tiered structures, sometimes with varying resolutions. This contrasts with the non-hierarchical **plain backbone** in VitDet model.

6.2.2.1. Shifted windows.

In the original **ViT** (Vision transformer) model, every patch attends to all other patches, which can be computationally heavy.

Swin simplifies this by making patches attend only to nearby patches in a window, similar to how CNNs work. This reduces the computational complexity from quadratic to linear, making it more efficient. Swin gradually merges information from neighboring patches, creating a hierarchical structure that helps handle images of different sizes effectively.



6.2.2.2. *Advantages*

- **Computational efficiency**: Swin is more performant than completely patch-based approaches like ViT.
- **Large datasets:** SwinV2 is one of the first 3B parameter models. As training size goes up, Swin outperforms CNN. Its extensive parameters enhance learning capacity, enabling the development of more intricate representations.

6.2.2.3. *Swin Versions & Applications*

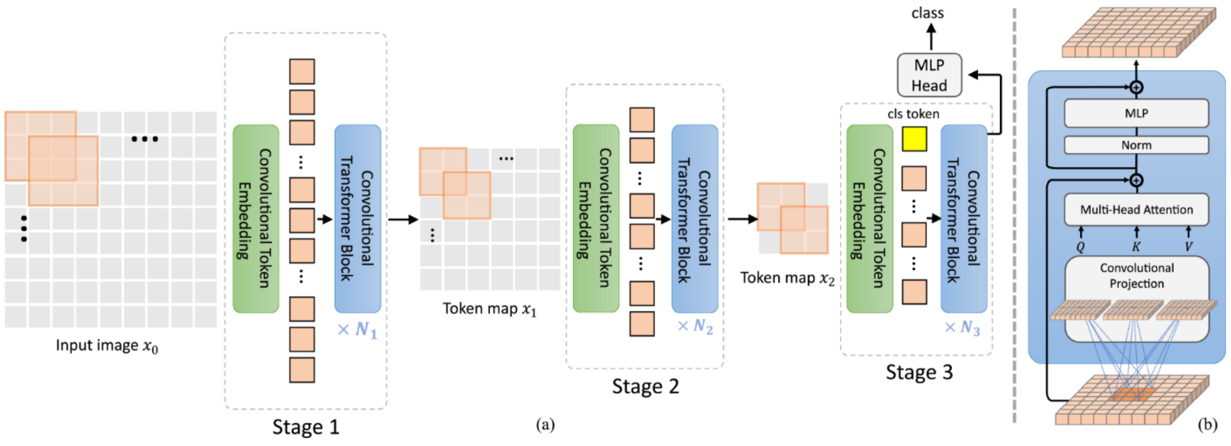
Swin Transformer V2 is a robust vision model with up to 3 billion parameters, designed for training with high-resolution images. It enhances the original Swin Transformer by ensuring stable training, enabling transfer learning from low to high resolution tasks, and employing SimMIM, a self-supervised training method that reduces the number of labeled images required for training.

Image Restoration:

- **SwinIR:** Is a model for turning low resolution images into high resolution images based on Swin Transformer.
- **Swin2SR:** Is another image restoration model. It is an improvement on SwinIR by incorporating Swin Transformer V2, applying the benefits of Swin V2 like training stability and higher image resolution capacity.

6.2.3. Convolutional Vision Transformer (CvT)

Convolutional Vision Transformer (CvT) is a variant of Vision Transformer (ViT) and extensively used for the Image Classification task in Computer Vision.



Convolutional Vision Transformer (CvT) employs all the benefits of CNNs (local receptive fields, shared weights, and spatial subsampling along with shift, scale, distortion invariance) while keeping merits of Transformers (dynamic attention, global context fusion, and better generalization).

CvT achieves superior performance while maintaining computational efficiency compared to ViT. Furthermore, due to built-in local context structure introduced by convolutions, CvT no longer requires a position embedding, giving it a potential advantage for adaption to a wide range of vision tasks requiring variable input resolution.

It involves two main components:

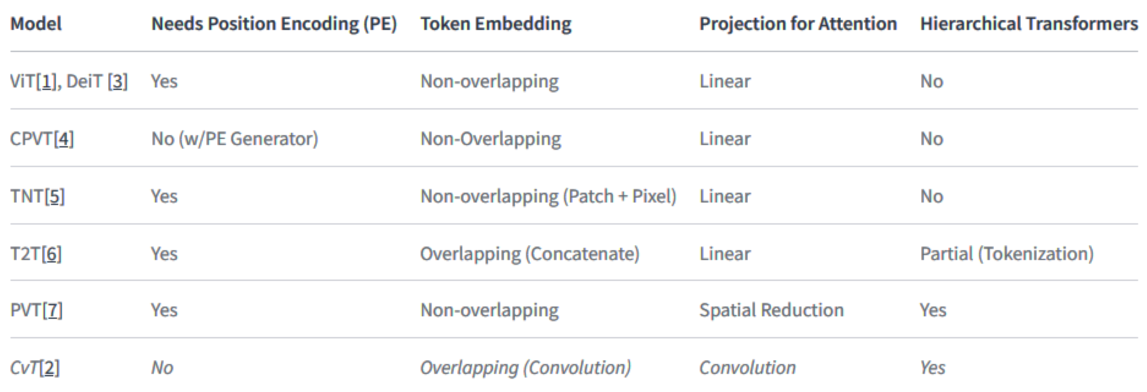
Convolutional Token Embedding: This step involves breaking down the input image into overlapping patches, converting them into tokens, and then passing them through a convolutional layer. This reduces the number of tokens while enhancing their feature richness, akin to traditional convolutional neural networks (CNNs). Unlike traditional Transformers, position information isn't explicitly added to tokens; instead, spatial relationships are captured solely through convolutional operations.

Convolutional Transformer Blocks: These blocks are stacked within each stage of CvT. Unlike ViT, which employs linear projections, CvT uses depth-wise separable convolutions (Convolutional Projection) to handle the "query," "key," and "value" components of the self-attention module. This maintains the benefits of Transformers while increasing efficiency. The "classification token" is only introduced in the final stage. Finally, a standard fully-connected layer processes the final classification token to predict the image class.

Key highlights of CvT:

- **Hierarchy of Transformers** containing a new **Convolutional token embedding**.
- Convolutional Transformer block leveraging a **Convolutional Projection**.
- **Removal of Positional Encoding** due to built-in local context structure introduced by convolutions.
- Fewer **Parameters** and lower **FLOPs** (Floating Point Operations per second) compared to other vision transformer architectures.

6.2.3.1. CvT Architecture Vs. other Vision Transformers



Model	Needs Position Encoding (PE)	Token Embedding	Projection for Attention	Hierarchical Transformers
ViT[1], DeiT [3]	Yes	Non-overlapping	Linear	No
CPVT[4]	No (w/PE Generator)	Non-Overlapping	Linear	No
TNT[5]	Yes	Non-overlapping (Patch + Pixel)	Linear	No
T2T[6]	Yes	Overlapping (Concatenate)	Linear	Partial (Tokenization)
PVT[7]	Yes	Non-overlapping	Spatial Reduction	Yes
CvT[2]	*No*	*Overlapping (Convolution)*	*Convolution*	*Yes*

Additional refs:

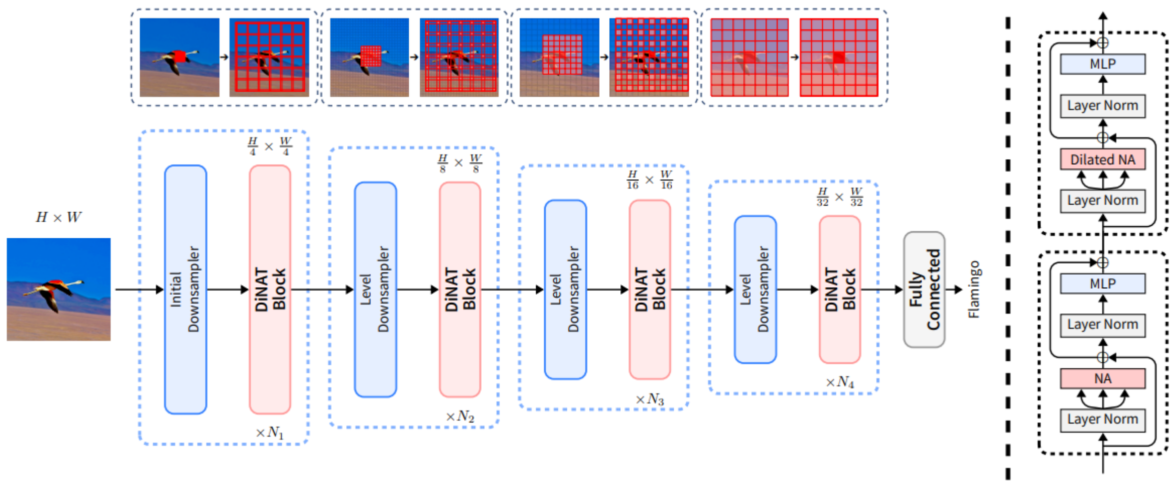
- *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*
- *CvT: Introducing Convolutions to Vision Transformers*
- *Training data-efficient image transformers & distillation through attention*
- *Conditional Positional Encodings for Vision Transformers*
- *Transformer in Transformer*
- *Tokens-to-Token ViT: Training Vision Transformers from Scratch on ImageNet*
- *Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions*
- *Implementation of CvT*

6.2.4. Dilated Neighborhood Attention Transformer (DINAT)

DiNAT stands for Dilated Neighborhood Attention Transformer, a cutting-edge hierarchical vision transformer designed to boost the performance of deep learning models in visual recognition tasks. Unlike traditional transformers, which can become computationally expensive as models grow larger, DiNAT introduces a novel attention mechanism called Dilated Neighborhood Attention (DiNA).

DiNA builds upon the concept of **Neighborhood Attention (NA),** a local attention mechanism. However, it enhances NA by incorporating sparse global attention without adding extra computational burden. This innovation allows DiNA to effectively capture more global context, significantly expand the receptive field, and model longer-range inter-dependencies in images while maintaining computational efficiency.

In the architecture of DiNAT, both NA and DiNA are combined, offering the best of both worlds: preserving locality and translational equivariance while also capturing extensive global context. This unique blend results in a transformer model that excels in understanding images and surpasses the performance of strong baseline models like NAT, Swin, and ConvNeXt across a range of visual recognition tasks.



Dilated Neighborhood Attention Transformer Architecture

DiNAT Block

6.2.4.1. *Neighborhood Attention*

Neighborhood Attention (NA) is a computer vision technique utilized in DiNAT, designed to enhance the understanding of relationships between pixels in an image efficiently.

It operates on two key principles:

- **Local Relationships:** NA enables each pixel to consider information from its immediate surroundings, akin to how we perceive a scene by focusing on nearby objects first before assessing the entire view.
- **Receptive Field:** NA dynamically expands the scope of pixels' attention to encompass more distant neighbors when necessary, facilitating a broader understanding of the image without significant computational overhead.

Additional refs:
- *DiNAT Paper*
- *Hugging Face DiNAT Transformer*
- *Neighborhood Attention(NA)*
- *SHI Labs*
- *OneFormer Paper*
- *Hugging Face OneFormer*

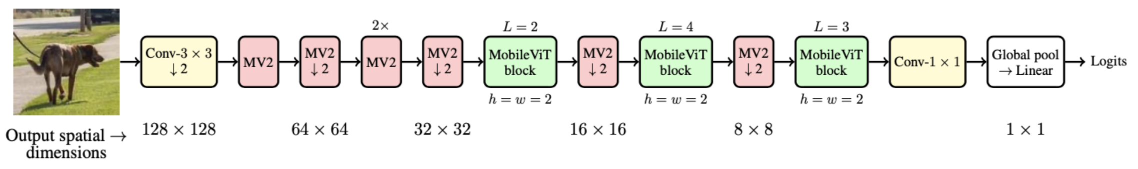
6.2.5. MobileViT v2

"MobileViT: Light-weight, General-purpose, and Mobile-friendly Vision Transformer"

MobileViT addresses the limitations of computationally intensive Vision Transformer architectures for mobile devices by combining the advantages of transformers and CNNs.

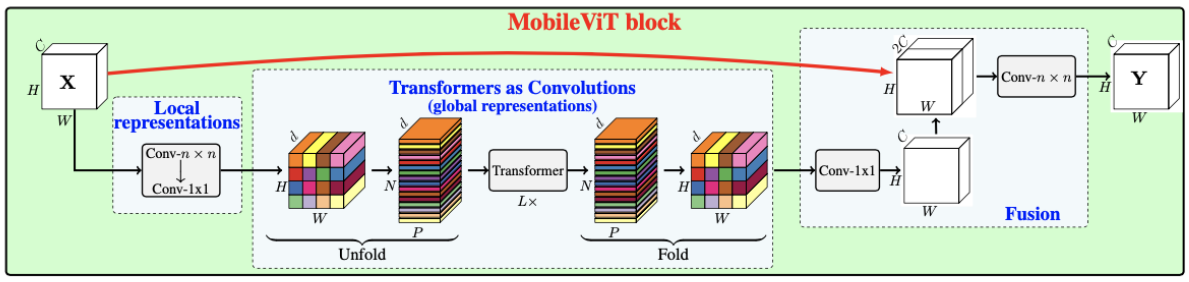
Developed by Apple, it enhances the MobileNet architecture by introducing the **MobileViT Block and separable self-attention**. These additions enable low-latency, lightweight architecture, parameter reduction, and efficient deployment of vision ML models on resource-constrained devices, offering a balance between performance and computational efficiency.

6.2.5.1. MobileViT Architeture



(b) **MobileViT**. Here, Conv-$n \times n$ in the MobileViT block represents a standard $n \times n$ convolution and MV2 refers to MobileNetv2 block. Blocks that perform down-sampling are marked with $\downarrow 2$.

The MobileViT block combines CNN's local processing and global processing, as seen in transformers. It uses a combination of convolutions and a transformer layer, allowing it to capture spatially local information and global dependencies in the data.

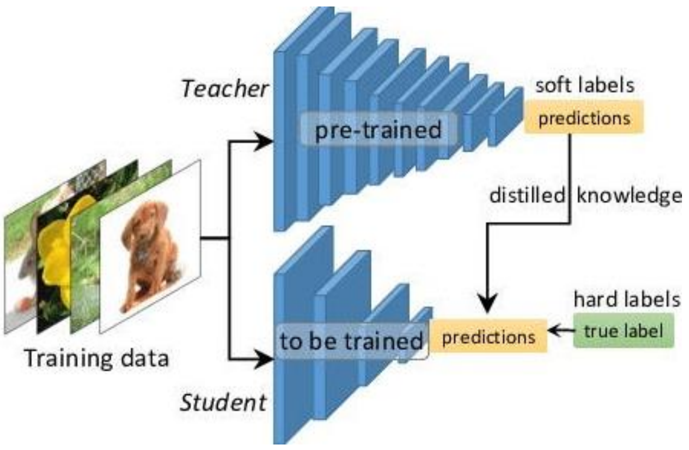


It processes multi-channel images by performing N by N convolutions, creating linear combinations, and projecting flattened patches through a transformer, followed by pointwise convolutions and recombination with the original RGB image.

This approach allows for a receptive field of H x W (the entire input size) while modeling non-local dependencies and local dependencies through retaining patch locational information. This can be seen in the unfolding and refolding of the patches.

6.3. Knowledge Distillation with Vision Transformers

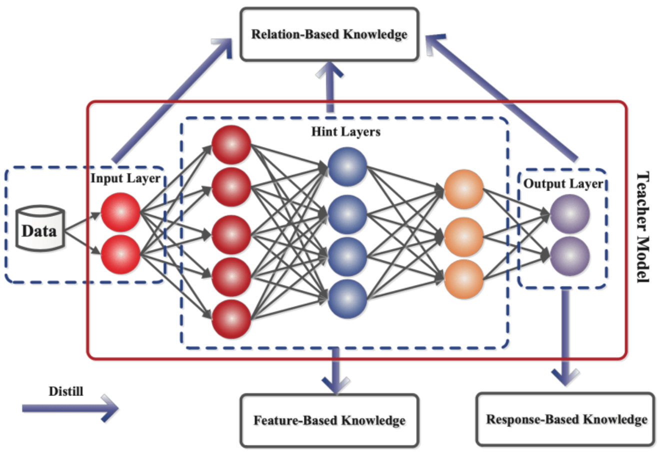
Knowledge distillation is a method where a smaller student model learns from a larger teacher model by mimicking its predictions (method behind distilGPT and distilbert).



This is done by adding a distillation loss to the training process, which encourages the student's output to match the teacher's predictions. The distillation loss measures the difference between the distributions of the teacher's and student's outputs, typically using the **Kullback-Leibler Divergence**.

$$L_{\text{distillation}} = \text{KL}(\sigma(z_t; T = t), \sigma(z_s; T = t))$$

The student's overall loss combines this distillation loss with the standard cross-entropy loss over the ground-truth labels, helping the student learn from the teacher while maintaining efficiency.



The different forms of knowledge that can be categorized into three different types:

- **Response-based knowledge:** involves the student model learning from the final output layer of the teacher model by minimizing the difference between their **logits**. In image classification tasks, soft targets derived from the teacher model's probability distribution over output classes are used, modulated by a temperature parameter, aiding supervised learning for the student model.
- **Feature-based knowledge:** focus on intermediate layers which learn to discriminate specific features and this knowledge can be used to train a student model. The distillation loss function achieves this by minimizing the difference between the feature activations of the teacher and the student models.

- **Relation-based knowledge**: captures the relationship between feature maps. This relationship can be modeled as correlation between feature maps, graphs, similarity matrix, feature embeddings, or probabilistic distributions based on feature representations.

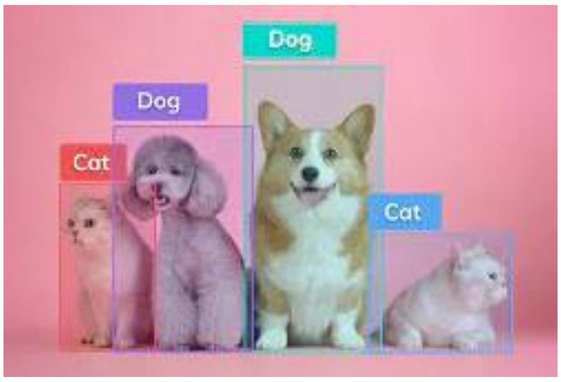
https://neptune.ai/blog/knowledge-distillation

7. Typical CV Tasks

7.1. Object Detection

Object detection is the task of identifying and locating specific objects within digital images or video frames. It has far-reaching implications across diverse sectors, including self-driving cars, facial recognition systems, and medical diagnosis tools.

Classification identifies objects based on distinct attributes, while localization pinpoints an object's position in an image. **Object detection merges** these by locating entities and assigning class labels simultaneously.

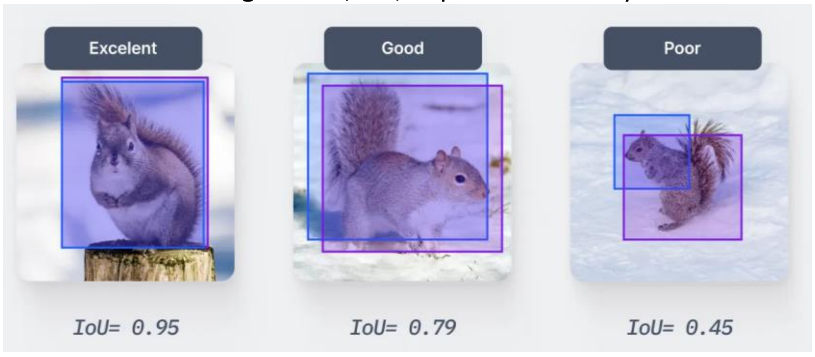


7.1.1. Evaluating an Object Detection Model

Object detection is primarily a supervised learning task. This means that the dataset is composed of images and their corresponding **bounding boxes**, which serve as the ground truth.

Several metrics are available for evaluating your model, with the most common ones including:

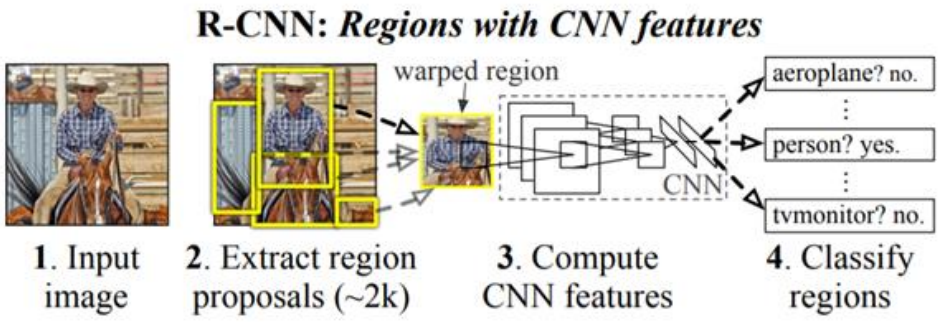
- **The Intersection over Union (IoU) or Jaccard index** measures the overlap between predicted and reference labels as a percentage ranging from 0% to 100%. Higher IoU percentages indicate better alignments, i.e., improved accuracy.



- **Mean Average Precision (mAP)** estimates object detection efficiency using both precision (correct prediction ratio) and recall (true positive identification ability). Calculated across varying IoU thresholds, mAP functions as a holistic assessment tool for object detection algorithms. Helpful for determining the model's performance in localization.

7.1.2. CNN Models

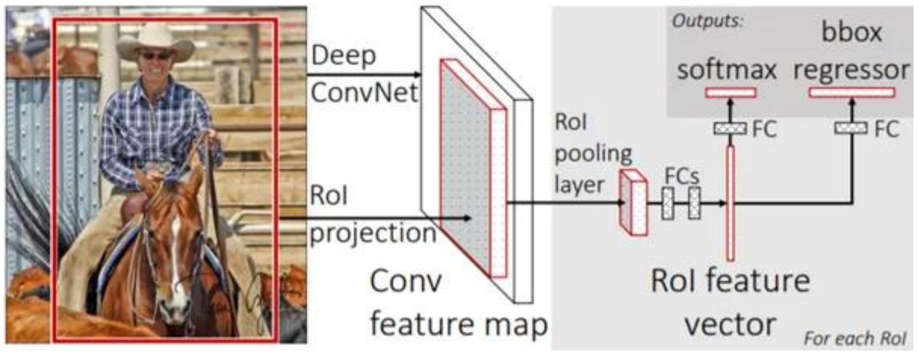
7.1.2.1. R-CNN



Before **R-CNN**, algorithms like Exhaustive Search were utilized, but they required significant computational power and time to locate objects accurately in images.

R-CNN introduced a solution by employing **Selective Search** to generate approximately 2000 region proposals, effectively reducing the number of locations to consider.

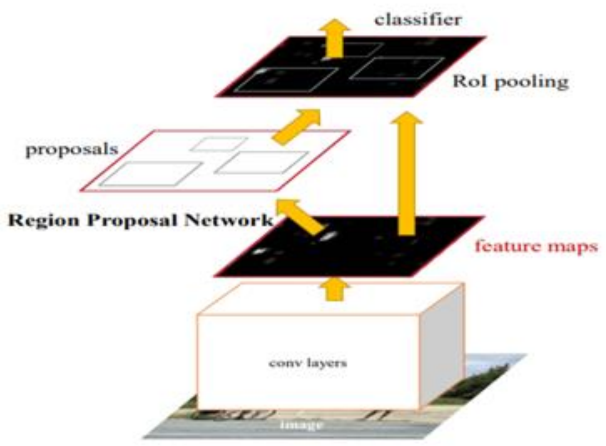
7.1.2.2. Fast R-CNN



Fast R-CNN improve by simplifying the process. Instead of passing region proposals individually to the CNN, the entire image is fed to generate a feature map.

From this map, we identify regions of interest and pass them through a **fully-connected layer**. Using a Softmax layer, we predict both classes and offset values for bounding boxes. Fast R-CNN is quicker than R-CNN because it performs the convolution operation just once per image, enhancing efficiency.

7.1.2.3. Faster-CNN

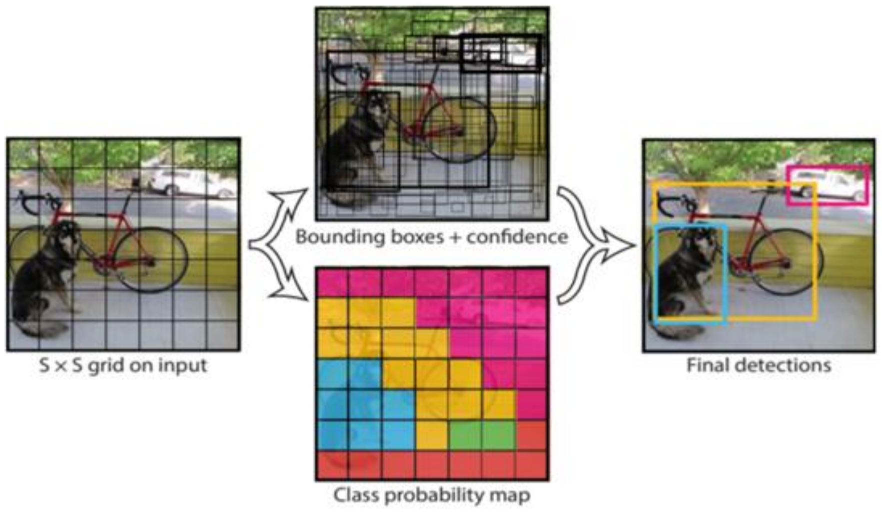


Faster R-CNN combines a Deep CNN for proposing regions with Fast R-CNN for utilizing these proposed regions. It's notably faster due to the introduction of **Region Proposal Network (RPN),** which replaces Selective Search.

RPN guides Fast R-CNN on where to focus by generating rectangular object proposals with associated objectness scores. Similar to Fast R-CNN, a single CNN processes the entire image to produce a **feature map.**

RPN operates on this map to generate proposals, reshaped using RoI pooling for predicting classes and bounding box offsets. This integrated approach enhances speed and efficiency in object detection tasks.

7.1.2.4. YOLO & YOLOv4



YOLO, which stands for "You Only Look Once," predicts bounding boxes and class probabilities directly from entire images in one evaluation, being faster than R-CNN family.

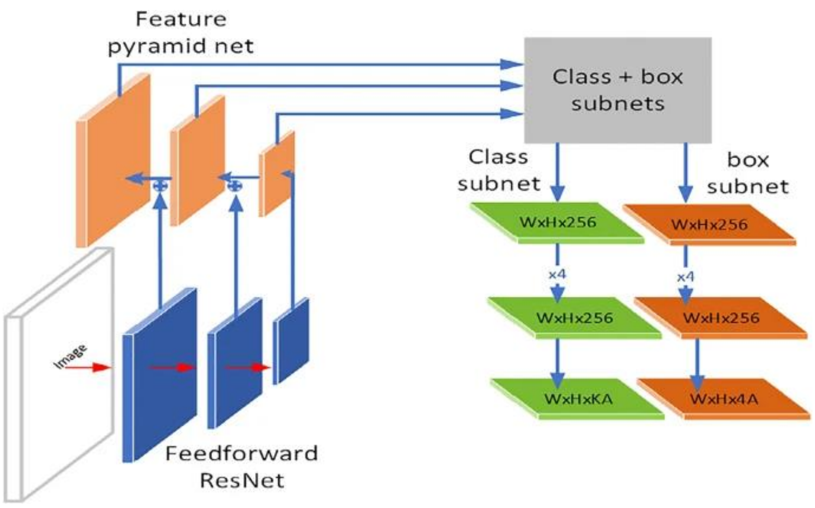
It works by **dividing the image into a grid,** and **each grid cell predicts bounding boxes** and confidence scores for those boxes. This approach allows YOLO to identify objects and their locations efficiently and accurately in images.

The network struggles with small objects and this was solved with later versions; Yolov2, Yolov3, Yolov4. **Yolov4** is state-of-art for object detection. It is extremely fast on real-time detection applications.

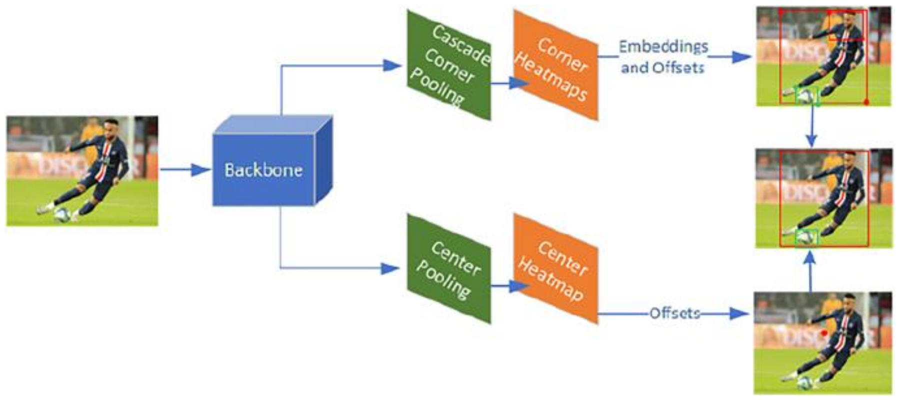
7.1.2.5. SSD, RetinaNet & CenterNet

Beside those, more models were proposed like:

- **SSD (Single Shot Detector):** takes an input image and passes it through VGG-16 to extract feature maps at various scales. Then, using convolutional filters, it predicts both object categories and offsets for bounding box locations.
- **RetinaNet :** is a single-stage object detection method that achieves the accuracy of two-stage methods by utilizing the Focal Loss to address sample imbalance. It combines ResNet for feature extraction and Feature Pyramid Network (FPN) for multi-scale feature utilization. This allows RetinaNet to accurately detect objects across various scales while maintaining efficiency.



- **CenterNet:** building upon the principles of CornerNet. Unlike traditional methods that rely on predefined anchor boxes, CenterNet is anchor-free, meaning it doesn't require prior assumptions about object shapes and sizes. Instead of focusing solely on corner points, CenterNet utilizes triples to define objects, enhancing the perception of internal object information. This is achieved through Cascade Corner Pooling, which extracts boundary information and progressively incorporates internal details for better object representation.



7.1.3. DEtection TRansformer (DETR)

DETR (DEtection TRansformer) is a modern object detection algorithm introduced in 2020.

A traditional object detection model like YOLO consists of hand-crafted features like anchor box priors, which requires initial guesses of object locations and shapes, affecting downstream training. Post-processing steps are then used to remove overlapping bounding boxes, which require careful selection of its filtering heuristics. Unlike those **DETR simplifies** the detector by using an encoder-decoder transformer after the feature extraction backbone to directly predict bounding boxes in parallel, requiring minimal post-processing.

DETR's architecture comprises three key components:

- a CNN backbone like ResNet for feature extraction
- a transformer encoder-decoder
- a feed-forward network (FFN) for final detection predictions.

The CNN backbone extracts features from the input image, which are then processed by the transformer encoder to capture global context using multi-head self-attention. The transformer decoder employs parallel decoding of object embeddings and independently predicts bounding box coordinates and class labels using object queries.

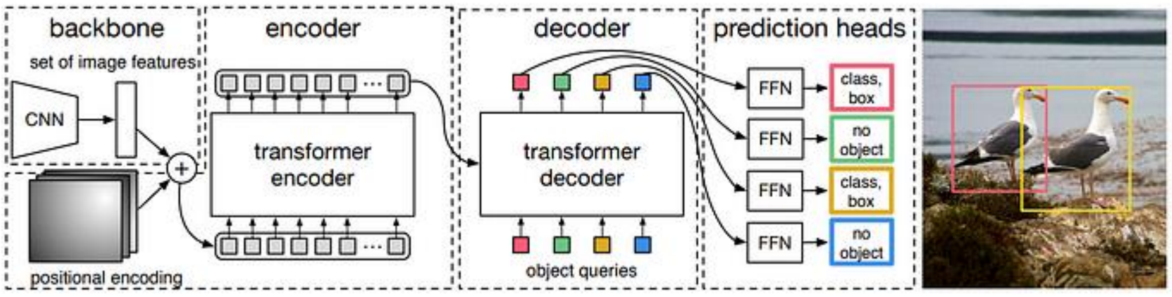


Fig. 2: DETR uses a conventional CNN backbone to learn a 2D representation of an input image. The model flattens it and supplements it with a positional encoding before passing it into a transformer encoder. A transformer decoder then takes as input a small fixed number of learned positional embeddings, which we call *object queries*, and additionally attends to the encoder output. We pass each output embedding of the decoder to a shared feed forward network (FFN) that predicts either a detection (class and bounding box) or a "`no object`" class.

Deformable DETR and Conditional DETR address key challenges in the original DETR model, aiming to improve convergence speed and enhance small object detection.

Deformable Attention: Traditional attention mechanisms in DETR are inefficient, leading to slow convergence and suboptimal small object detection. Deformable Attention reduces this by focusing attention on specific sampling points around each reference point, learned dynamically from the input. This allows the model to attend to relevant areas, improving performance.

Conditional DETR: To accelerate convergence, Conditional DETR introduces Conditional Cross-Attention in the decoder. This enhances the localization of bounding box regression by incorporating content queries (cq) and spatial queries (pq) separately. By combining information from decoder embeddings and object queries, Conditional DETR improves the model's ability to localize objects accurately, resulting in faster convergence and enhanced performance.

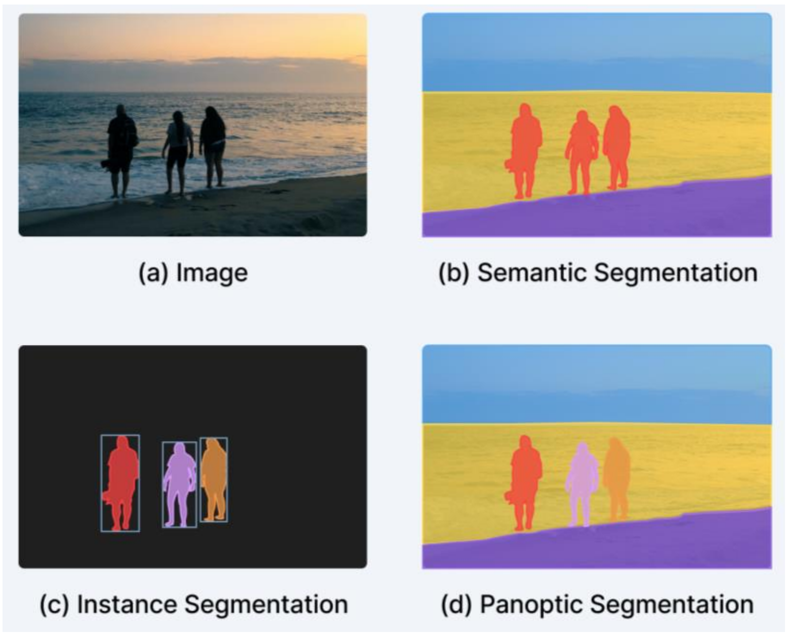
Additional refs:
- *Hugging Face Object Detection Guide*
- *Object Detection in 20 Years: A Survey*
- *Papers with Code - Real-Time Object Detection*
- *Papers with Code - Object Detection*
- *SSD: Single Shot MultiBox Detector*
- *YOLOv4: Optimal Speed and Accuracy of Object Detection*
- *A survey: object detection methods from CNN to transformer*
- *DETR*
- *YOLO*
- *YOLOv3*
- *Conditional DETR*
- *Deformable DETR*
- *facebook/detr-resnet-50*
- *https://medium.com/@faheemrustamy/detection-transformer-detr-vs-yolo-for-object-detection-baeb3c50bc3*

7.2. Image Segmentation

Image segmentation is dividing an image into **meaningful segments**. It's all about creating masks that spotlight each object in the picture. The intuition behind this task is that it can be viewed as a classification for each pixel of the image.

Different types of segmentations can be applied:

- **Semantic Segmentation:** each pixel is assigned the most likely class. For instance, in semantic segmentation, the model doesn't differentiate between individual cats but focuses on pixel-level classification.
- **Instance Segmentation:** This type involves identifying each instance of an object with a unique mask. It combines aspects of object detection and segmentation to differentiate between individual objects of the same class.
- **Panoptic Segmentation:** A hybrid approach that combines elements of semantic and instance segmentation. It assigns a class and an instance to each pixel, effectively integrating the what and where aspects of the image.


(a) Image
(b) Semantic Segmentation
(c) Instance Segmentation
(d) Panoptic Segmentation

Note: Recent models allow you to achieve the three segmentation types with a single model. *Mask2former* is a new model by Meta that achieves the three segmentation types with only a Panoptic dataset.

7.2.1. Evaluating Image Segmentation

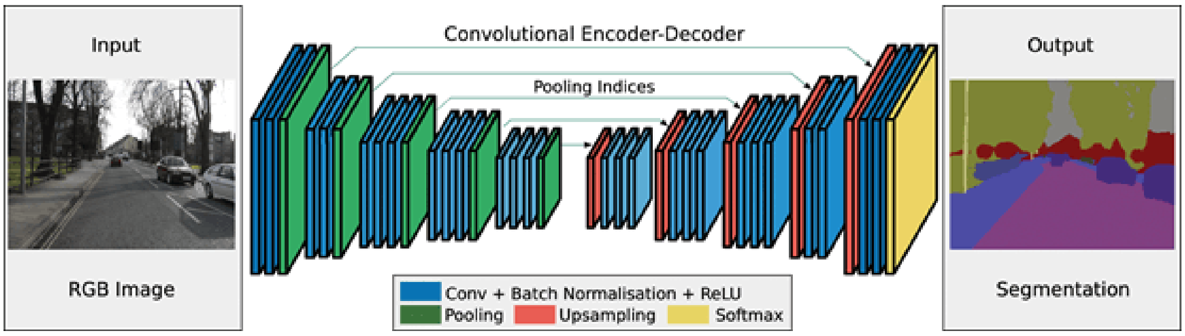
Segmentation is supervised learning, utilizing datasets of images paired with corresponding masks as ground truth. Evaluation metrics, including various measures, assess model performance against these ground truth annotations:

- **The Intersection over Union (IoU) or Jaccard index metric** is the ratio between the intersection and the union of the predicted mask and the ground truth. IoU is arguably the most common metric used in segmentation tasks. Its advantage lies in being less sensitive to class imbalance, making it often a good choice when you begin modeling.
- **Pixel accuracy**: Pixel accuracy is calculated as the ratio of the number of correctly classified pixels to the total number of pixels. While being an intuitive metric, it can be misleading due to its sensitivity to class imbalance.
- **Dice coefficient**: It's the ratio between the double of the intersection and the sum of the predicted mask and the ground truth. The dice coefficient is simply the percentage of overlap between the prediction and the ground truth. It's a good metric to use when you need sensibility to small differences between the overlap.

7.2.2. Some main CNN models for Image Segmentation

7.2.2.1. Encoder-Decoder

In image segmentation, the fundamental architecture comprises an encoder and a decoder. The encoder extracts image features using filters, while the decoder generates the final output, typically a segmentation mask outlining the objects. This basic architecture, or its variations, is common across many segmentation models.



7.2.2.2. U-Net

U-Net is a convolutional neural network initially designed for segmenting biomedical images, shaped like the letter U, hence its name. Its architecture consists of two components: the **contracting path and the expansive path.**

The contracting path focuses on capturing context through two consecutive three-by-three convolutions, followed by rectified linear units and two-by-two max-pooling for downsampling.

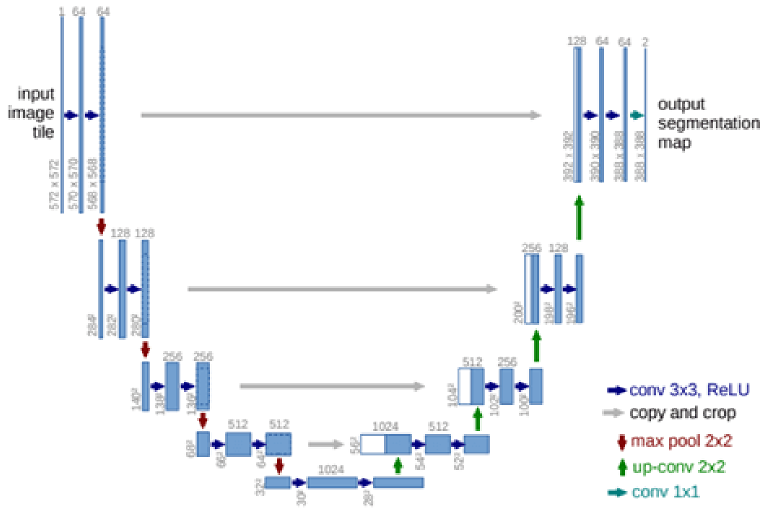


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

7.2.2.3. Gated-SCNN

This architecture consists of **a two-stream CNN architecture**. In this model, a separate branch is used to process image shape information. The shape stream is used to process boundary information.

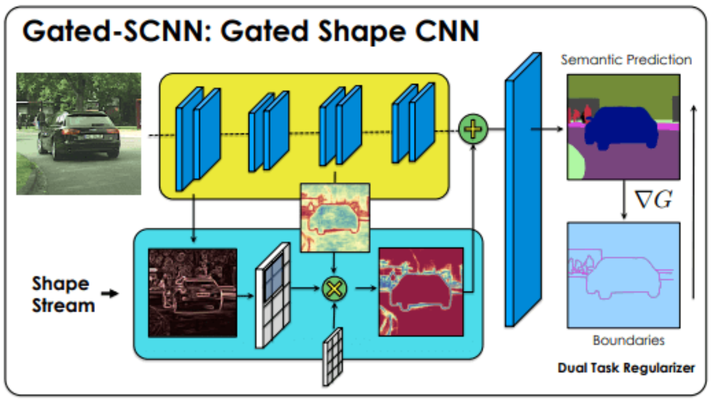


Figure 1: We introduce *Gated-SCNN* (GSCNN), a new two-stream CNN architecture for semantic segmentation that explicitly wires shape information as a separate processing stream. GSCNN uses a new gating mechanism to connect the intermediate layers. Fusion of information between streams is done at the very end through a fusion module. To predict high-quality boundaries, we exploit a new loss function that encourages the predicted semantic segmentation masks to align with ground-truth boundaries.

7.2.2.4. Mask R-CNN

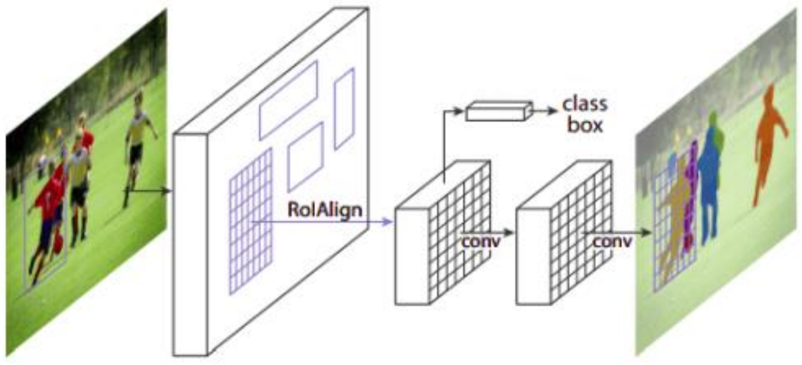


Figure 1. The **Mask R-CNN** framework for instance segmentation.

This architecture combines object classification and localization using both bounding boxes and semantic segmentation, where each pixel is categorized into specific classes. It extends the Faster R-CNN model, which comprises a deep convolutional network for region proposal and a detector to utilize these regions.

The final output includes class labels and bounding boxes for each region of interest, providing comprehensive object detection and localization.

7.2.3. CNNs vs Transformers for Segmentation

Before the emergence of **Vision Transformers, CNNs (Convolutional Neural Networks**) have been the go-to choose for image segmentation. Models like U-Net and Mask R-CNN captured the details that are needed to distinguish different objects in an image, making them state-of-the-art for segmentation tasks.

Despite their excellent results over the past decade, CNN-based models have some limitations, which Transformers aim to solve:

- **Spatial limitations:** CNNs struggle to capture long-range dependencies in images, affecting segmentation accuracy. Transformers like Vision Transformers (ViTs) consider the entire image at once, enabling them to understand complex relationships between distant parts and improving object delineation.
- **Task-specific components:** CNN-based methods often rely on hand-designed (e.g., non-maximum suppression, spatial anchors) to encode prior knowledge about segmentation tasks. Adding complexity and requiring manual tuning. ViT-based approaches simplify segmentation by eliminating the need for these components, making optimization easier.
- **Segmentation task specialization:** CNN-based models typically treat semantic, instance, and panoptic segmentation as separate tasks, resulting in specialized architectures and research efforts. Recent ViT-based models like MaskFormer, SegFormer or SAM offer a unified approach, handling multiple segmentation tasks within a single framework.

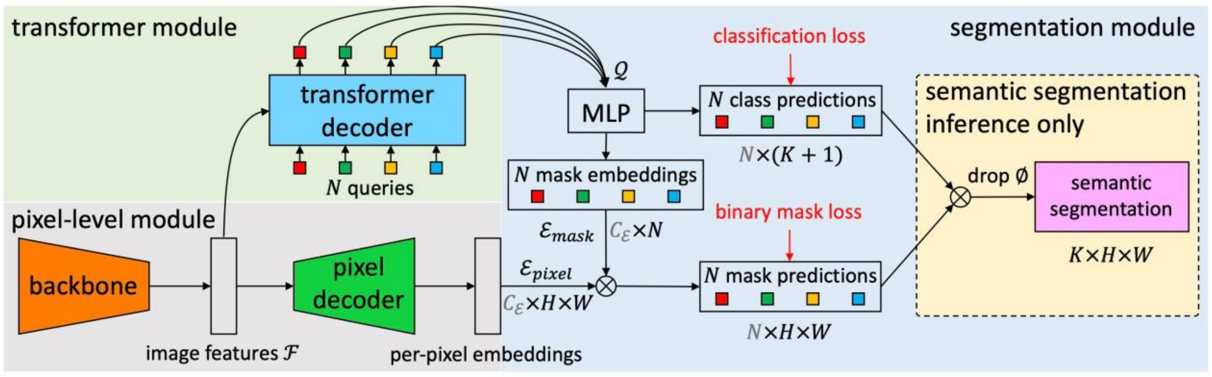
7.2.4. Transformer-based image segmentation

MaskFormer is a model designed for semantic segmentation, predicting segmentation masks for each class in an image while combining semantic and instance segmentation in one architecture.

The architecture consists of three main components:

1. **Pixel-level Module**: Utilizes a backbone to extract features from the input image and a pixel decoder to generate embeddings for each pixel.
2. **Transformer Module**: Employs a standard Transformer decoder to compute embeddings for each segment based on the image features and learnable positional embeddings. This module captures global information about each segment.
3. **Segmentation Module**: Generates class probability predictions and mask embeddings for each segment using a linear classifier and a Multi-Layer Perceptron (MLP) respectively. These mask embeddings, combined with per-pixel embeddings, are used to predict binary masks for each segment.

During training, the model is optimized using a binary mask loss, similar to DETR, and a cross-entropy classification loss per predicted segment.



7.2.5. Universal Image Segmentation Transformers

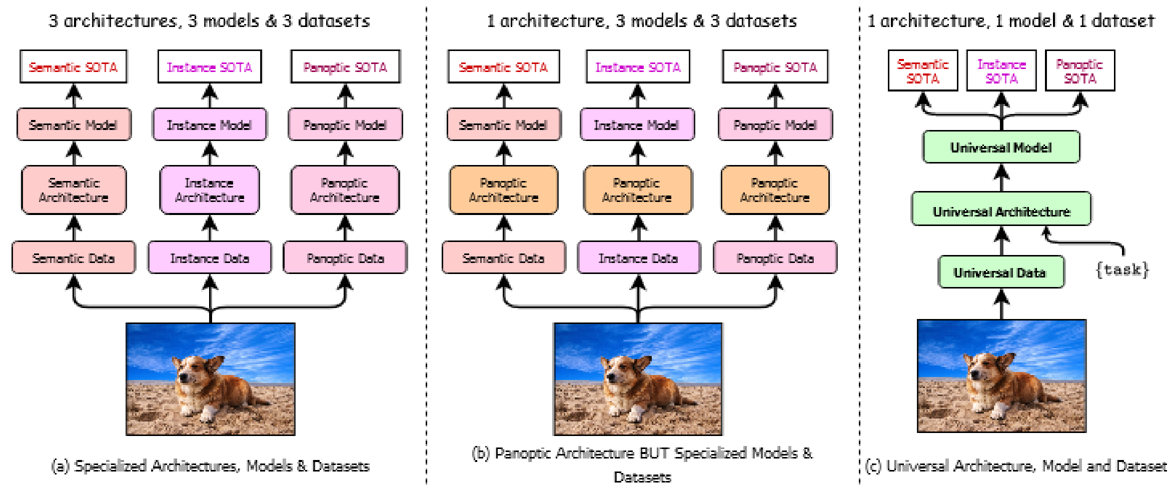
7.2.5.1. *Mask2Former*

Mask2Former, an evolution of MaskFormer, introduces two key improvements:

- **Mask Attention:** Unlike MaskFormer's cross-attention mechanism, Mask2Former utilizes masked attention in its Transformer decoder. This allows the model to focus only on localized features centered around predicted segments, leading to faster convergence and improved performance by considering relevant parts of the input sequence.
- **Multi-scale High-Resolution Features:** To address small objects, Mask2Former incorporates multi-scale feature representation. It includes features at different scales or resolutions to capture fine-grained details and broader context information effectively. By processing these features with specific Transformer decoder layers, the model can handle objects of varying sizes more efficiently.

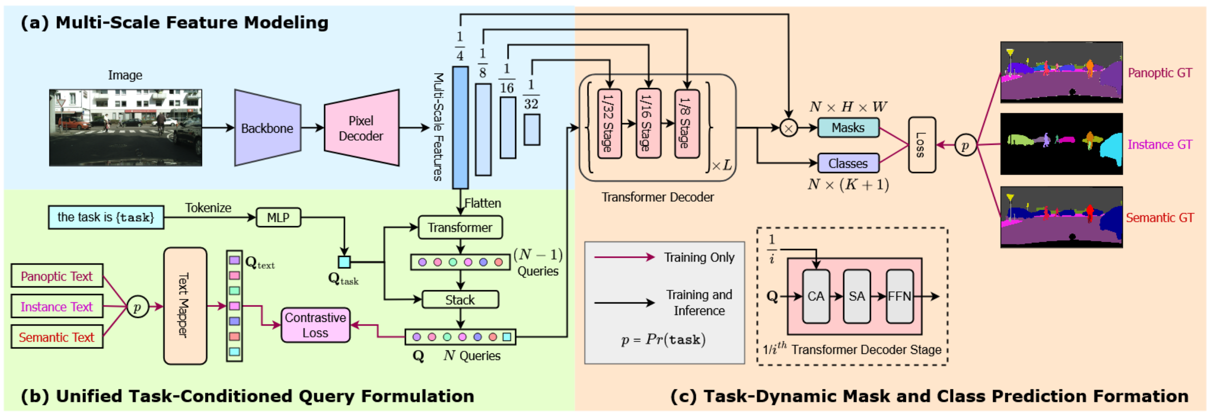
Overall, Mask2Former revolutionizes image segmentation by simultaneously tackling multiple tasks, such as panoptic, semantic, and instance segmentation. Its advancements, including masked attention and multi-scale features, outperform specialized architectures and offer streamlined development and improved performance.

7.2.5.2. OneFormer: One Transformer to Rule Universal Image Segmentation


3 architectures, 3 models & 3 datasets
Semantic SOTA
Instance SOTA
Panoptic SOTA
Semantic Model
Instance Model
Panoptic Model
Semantic Architecture
Instance Architecture
Panoptic Architecture
Semantic Data
Instance Data
Panoptic Data
(a) Specialized Architectures, Models & Datasets
1 architecture, 3 models & 3 datasets
(b) Panoptic Architecture BUT Specialized Models & Datasets
1 architecture, 1 model & 1 dataset
Universal Model
Universal Architecture
Universal Data
{task}
(c) Universal Architecture, Model and Dataset

OneFormer is a revolutionary image segmentation framework that addresses the complexity of traditional segmentation methods. Instead of relying on separate models for different segmentation tasks, like identifying objects or labeling regions.

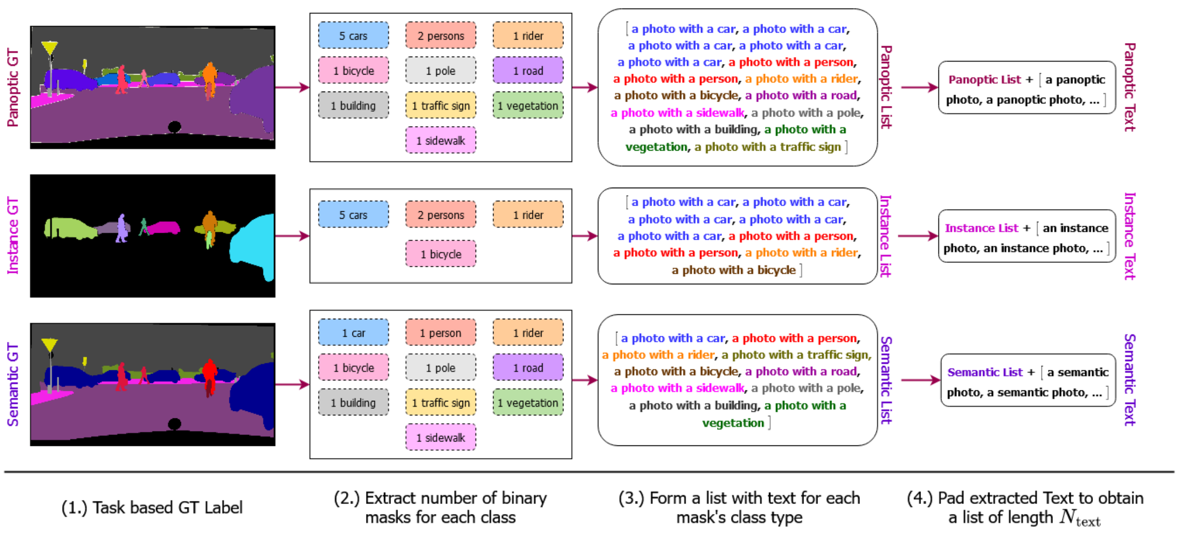
OneFormer introduces a unified approach. It utilizes a multi-task strategy, enabling a single model to handle semantic, instance, and panoptic segmentation tasks without separate training. This is achieved through task-conditioned joint training, where the model dynamically adapts to different tasks during both training and inference.



The key innovation of OneFormer lies in its ability to leverage panoptic annotations, which provide unified ground truth information for all segmentation tasks. Additionally, the framework **incorporates query-text contrastive learning** to better distinguish between tasks and improve overall performance.

This breakthrough simplifies the training process and yields superior results across various datasets, making OneFormer a highly effective and versatile solution for image segmentation.

How OneFormer do the trick?


Panoptic GT
Instance GT
Semantic GT
5 cars
2 persons
1 rider
1 bicycle
1 pole
1 road
1 building
1 traffic sign
1 vegetation
1 sidewalk
1 car
1 person
[a photo with a car, a photo with a car, a photo with a car, a photo with a car, a photo with a car, a photo with a person, a photo with a person, a photo with a rider, a photo with a bicycle, a photo with a road, a photo with a sidewalk, a photo with a pole, a photo with a building, a photo with a vegetation, a photo with a traffic sign]
[a photo with a car, a photo with a car, a photo with a car, a photo with a car, a photo with a car, a photo with a person, a photo with a person, a photo with a rider, a photo with a bicycle]
[a photo with a car, a photo with a person, a photo with a rider, a photo with a traffic sign, a photo with a bicycle, a photo with a road, a photo with a sidewalk, a photo with a pole, a photo with a building, a photo with a vegetation]
Panoptic List
Instance List
Semantic List
Panoptic List + [a panoptic photo, a panoptic photo, ...]
Instance List + [an instance photo, an instance photo, ...]
Semantic List + [a semantic photo, a semantic photo, ...]
Panoptic Text
Instance Text
Semantic Text
(1.) Task based GT Label
(2.) Extract number of binary masks for each class
(3.) Form a list with text for each mask's class type
(4.) Pad extracted Text to obtain a list of length N_{text}

OneFormer employs several innovative techniques to enhance image segmentation:

- **Task-Dynamic Mask:** This feature enables the model to adapt its focus based on the specific segmentation task at hand. It decides whether to concentrate on the overall scene, identify individual objects, or do a combination of both, enhancing its flexibility.
- **Task-Conditioned Joint Training:** Instead of training separately for different segmentation tasks, OneFormer employs a unified training approach. It uniformly samples the segmentation task during training, allowing the model to learn across various tasks simultaneously. By conditioning the architecture on the specific task through the task token, OneFormer streamlines the training process and reduces the need for task-specific architectures.
- **Query-Text Contrastive Loss:** This technique helps OneFormer understand the differences between tasks and classes by comparing visual features extracted from the image (queries) with corresponding text descriptions. It reduces confusion between different classes and improves task-specific learning.

By integrating these methods, OneFormer becomes a versatile and efficient tool for image segmentation, offering a universal approach to tackle various segmentation tasks and outperform existing frameworks.

Additional refs:

- *Segment Anything Paper*
- *Fine-tuning Segformer blog post*
- *Mask2former blog post*
- *Hugging Face's documentation on segmentation tasks*
- *Stanford's lecture on segmentation.*
- *https://medium.com/@HannaMerqui/maskformer-per-pixel-classification-is-not-all-you-need-for-semantic-segmentation-1e2fe3bf31cb*
- *OneFormer Paper*
- *HuggingFace OneFormer model*