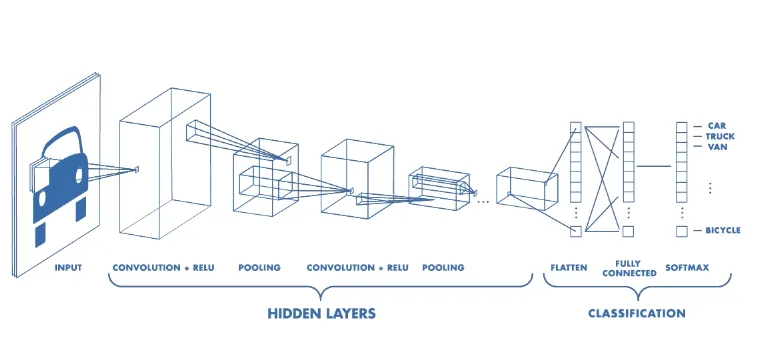
CNN architecture

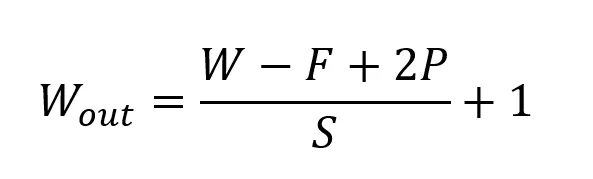


**Convolution layer**

Spatial size -> F

Stride ->s

padding-> p



**Pooling Layer**

The pooling layer replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights.

Several pooling functions like max pooling.

**Fully Connected Layer**

Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer as seen in regular FCNN.

Advantages:

Efficient image processing:This is because they use a technique called convolution, which involves applying a filter to an image to extract features that are relevant to the task at hand.

High accuracy rates:This is because they can learn to recognize complex patterns in images by analyzing large datasets.

Robust to noise – CNNs are also robust to noise, which means that they can still recognize patterns in images even if they are distorted or corrupted. This is because they use multiple layers of filters to extract features from images, which makes them more resilient to noise than other types of algorithms.

Transfer learning – CNNs also support transfer learning, which means that they can be trained on one task and then used to perform another task with little or no additional training.

Automated feature extraction – Finally, CNNs automate the feature extraction process, which means that they can learn to recognize patterns in images without the need for manual feature engineering.

Disadvantages:

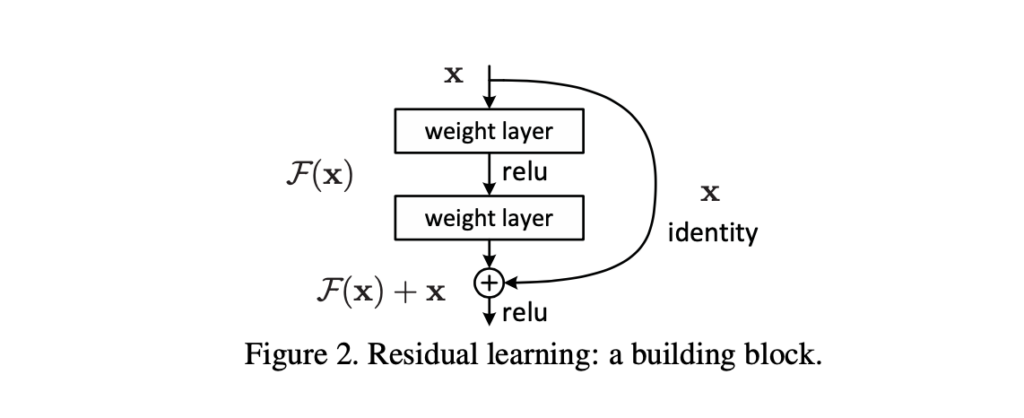
This is because CNNs typically have a large number of layers and parameters, which require a lot of processing power and memory to train and run. This can make them impractical for use in some applications where resources are limited.

Difficulty with small datasets – CNNs also require large datasets to achieve high accuracy rates. This is because they learn to recognize patterns in images by analyzing many examples of those patterns. CNNs also require large datasets to achieve high accuracy rates.

Limited ability to generalize-This means that they may perform poorly on images that are very different from those in the training dataset. This can be a problem in applications where the CNN needs to work with a wide variety of images.

**ResNet**

ResNet is known for its ability to handle very deep networks, which is crucial for image processing tasks that often require complex feature extraction. Deep ResNet models, such as ResNet-50, ResNet-101, and ResNet-152, have demonstrated superior performance on challenging image classification benchmarks like ImageNet.



ResNet's architecture allows for training very deep neural networks without the problem of vanishing gradients.

ResNet models pre-trained on large-scale datasets, such as ImageNet, can be used as powerful feature extractors for transfer learning. This is particularly beneficial when dealing with limited labeled data, as the pre-trained models capture general visual representations that can be fine-tuned for specific tasks. The residual connections in ResNet enable more efficient training by allowing gradients to flow directly through the network. ResNet's ability to learn residual functions allows for better generalization and robustness. The residual blocks capture residual mappings, enabling the model to focus on learning the residual details of the data while maintaining the overall identity of the input.

Cons:

ResNet's deep architecture requires a large number of parameters, leading to increased memory usage during training and inference. This can pose challenges when working with limited computational resources or deploying models on resource-constrained devices.

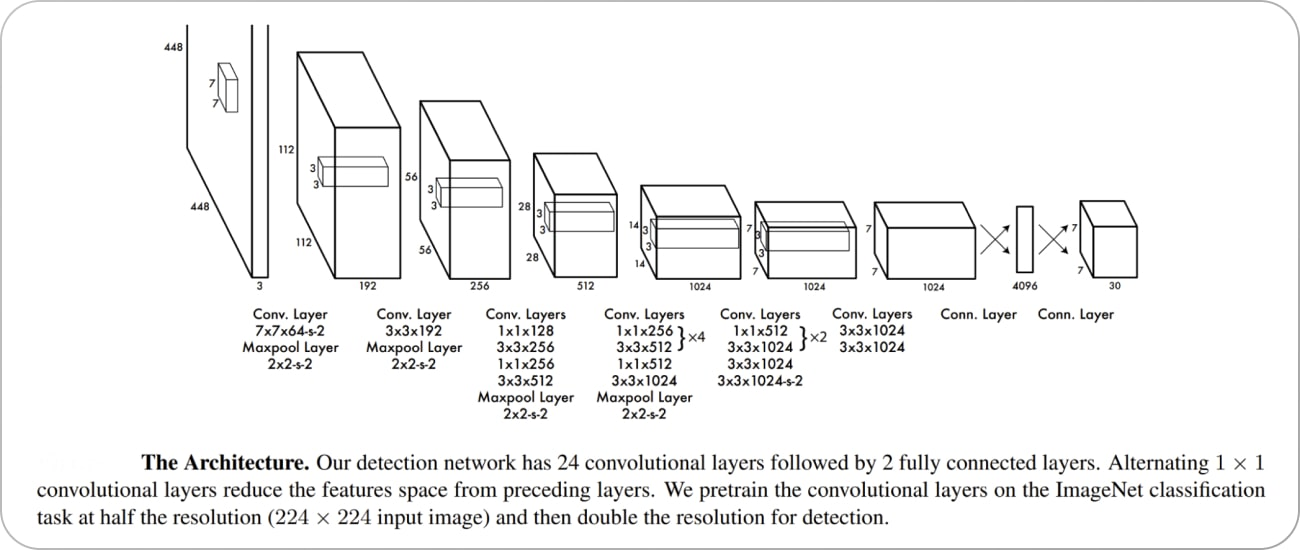
Deep networks like ResNet are prone to overfitting when trained on small datasets.

While ResNet offers different architectural variants with varying depths, it can be challenging to determine the most suitable variant for a specific task.

The black-box nature of deep learning models limits their interpretability, which may be a concern in certain domains where interpretability is crucial.

**Yolo**

YOLO is a single-shot detector that uses a fully convolutional neural network (CNN) to process an image.



Pros:

Real-time Performance: YOLO is known for its exceptional speed, capable of processing images and videos in real-time, making it suitable for time-sensitive applications.

Simplicity: YOLO's architecture is relatively straightforward, involving a single pass through the network, which simplifies the implementation and reduces computational overhead.

YOLO achieves competitive accuracy in object detection, because of its ability to capture multi-scale features and utilize anchor boxes for precise bounding box predictions.

YOLO accurately localizes objects by directly regressing the bounding box coordinates, which leads to precise and consistent localization results.

YOLO can detect a wide range of object classes across various domains, making it suitable for diverse applications such as surveillance, robotics, and autonomous driving.

Cons:

Smaller Object Detection: YOLO may struggle to detect small objects due to the fixed grid size and limited resolution of the feature maps, which can lead to lower accuracy for small and densely packed objects.

Difficulty with Aspect Ratios: YOLO's fixed anchor boxes may struggle with objects that have extreme aspect ratios, as the predefined anchor boxes may not adequately capture their shapes.Localization Accuracy for Overlapping Objects: YOLO may face challenges accurately localizing overlapping objects, as it assigns a single grid cell responsibility for detecting objects within its boundaries, leading to potential confusion when multiple objects overlap.Training Data Imbalance: YOLO can suffer from data imbalance, where certain object classes are significantly underrepresented in the training data. This can impact the model's ability to accurately detect and classify rare or less frequent classes.High Memory Usage: The deeper variants of YOLO, such as YOLOv4, require substantial memory resources, which can limit deployment on resource-constrained devices.