# Convolutional Neural Networks Introduction

## Table of Contents



Augusto Damasceno

augustodamasceno@protonmail.com

augustodamasceno.org



Images in the dataset folder are from the "Vehicle Detection Image Set" by Baris Dincer on Kaggle.

Source: https://www.kaggle.com/datasets/brsdincer/vehicle-detection-image-set

License: Open Data Commons Database Contents License v1.0

SPDX-License-Identifier: ODC-DbCL-1.0

# Clean

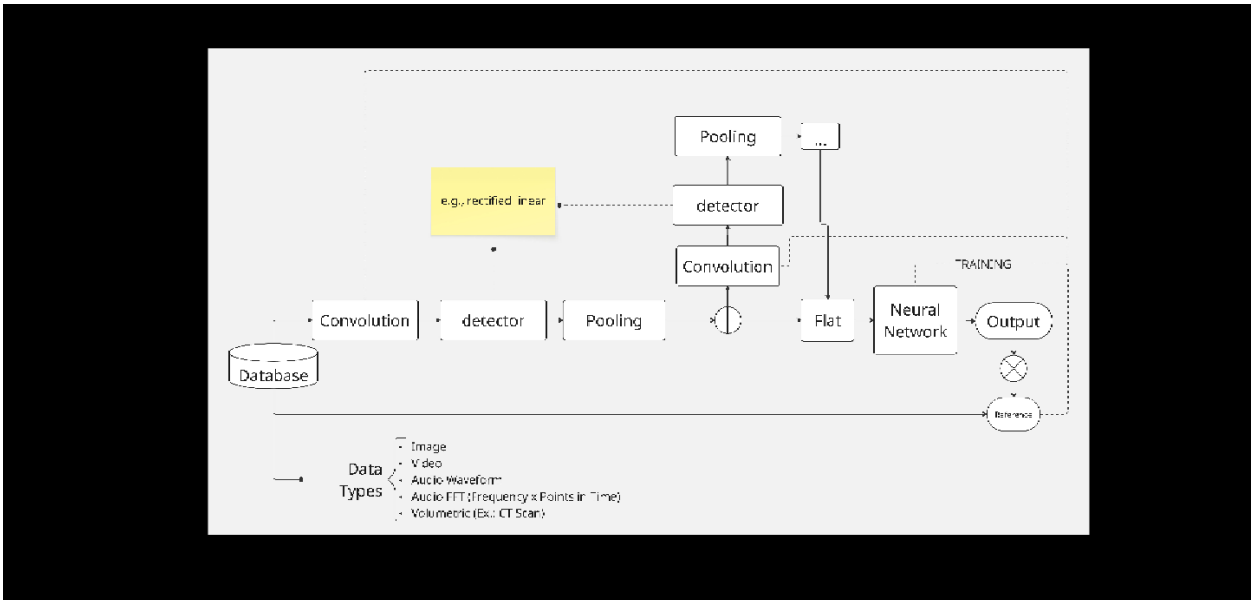
```
% Removes all variables from the workspace.
clear;
% Close all figures and
close all;
% Clear Command Window
clc
```

# Definition

Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers. [1]

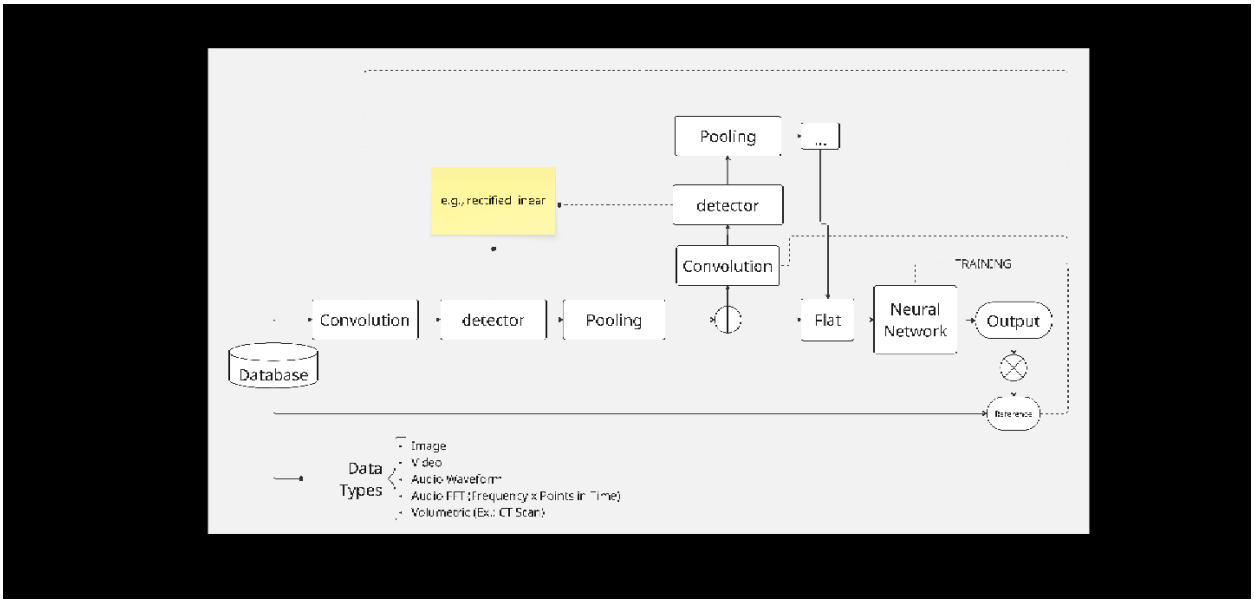
# Overview

```
imshow("overview.jpg");
```



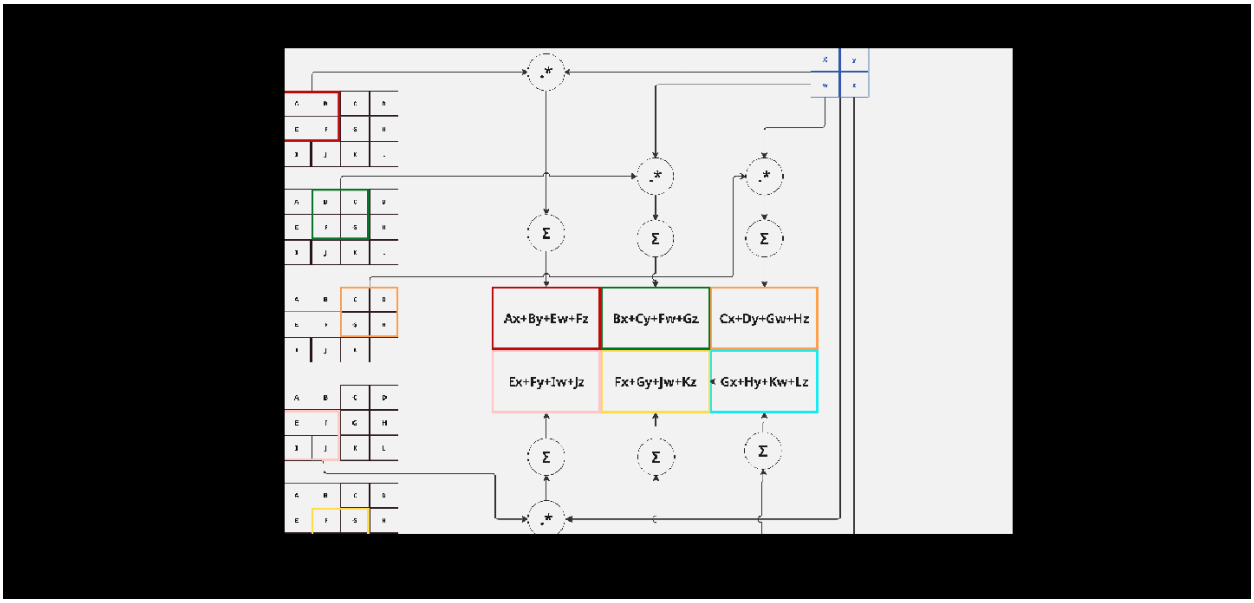

# Overview

```
imshow("overview.jpg");
```



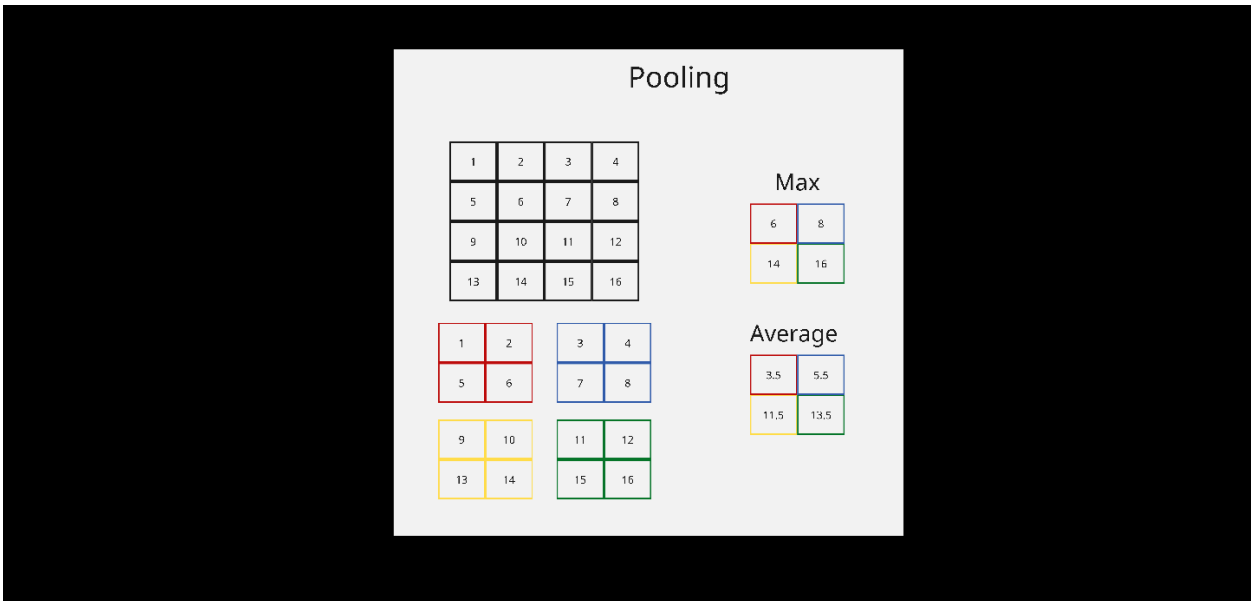
# Convolution

```
imshow("convolution.jpg");
```




# Pooling

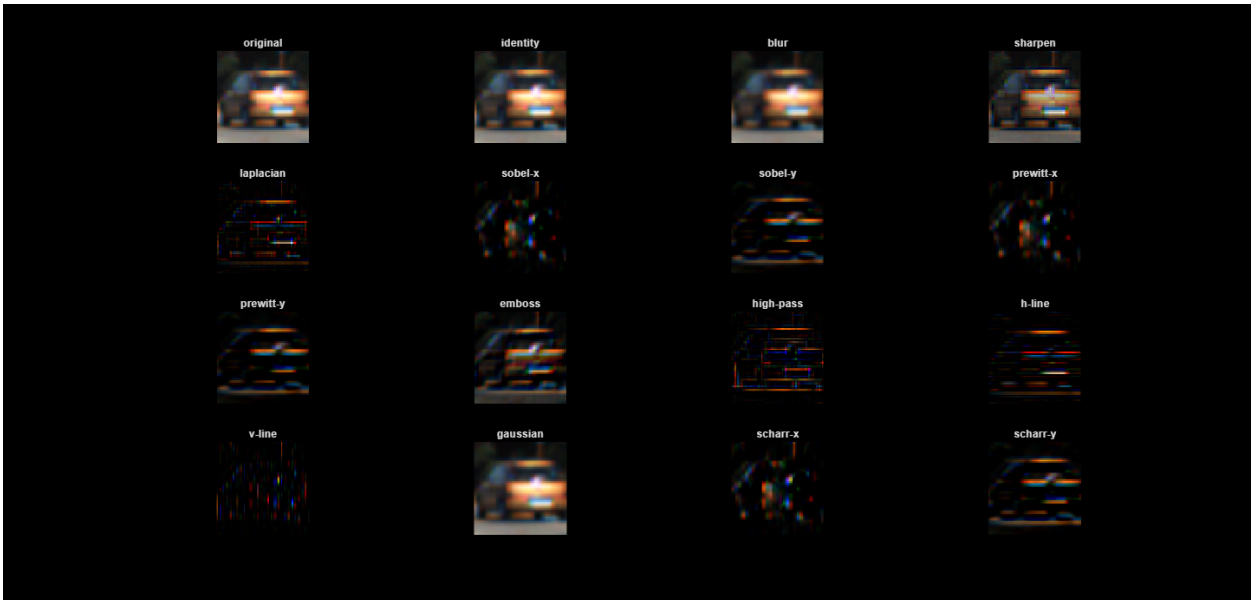
```
imshow("pooling.jpg");
```




# Convolution

```
kernels = {
    [0 0 0; 0 1 0; 0 0 0];              % identity
```

```
    ones(3)/9;                          % box blur
    [0 -1 0; -1 5 -1; 0 -1 0];          % sharpen
    [0 -1 0; -1 4 -1; 0 -1 0];          % laplacian
    [-1 0 1; -2 0 2; -1 0 1];           % sobel-x
    [-1 -2 -1; 0 0 0; 1 2 1];           % sobel-y
    [-1 0 1; -1 0 1; -1 0 1];           % prewitt-x
    [-1 -1 -1; 0 0 0; 1 1 1]            % prewitt-y
    [-2 -1 0; -1 1 1; 0 1 2];           % emboss
    [1 1 1; 1 -8 1; 1 1 1];             % high-pass (hp8)
    [-1 -1 -1; 2 2 2; -1 -1 -1];        % horizontal line
    [-1 2 -1; -1 2 -1; -1 2 -1]         % vertical line
    [1 2 1; 2 4 2; 1 2 1]/16            % gaussian
    [-3 0 3; -10 0 10; -3 0 3]/16       % scharr-x
    [-3 -10 -3; 0 0 0; 3 10 3]/16       % scharr-y
};

img = imread('dataset/vehicles/7.png');
img = im2double(img);
[H, W, C] = size(img);
dimStr = sprintf('Image dimensions: %d x %d x %d', H, W, C);

figure(1);
filter_names = {'identity','blur','sharpen','laplacian', ...
                'sobel-x','sobel-y','prewitt-x','prewitt-y', ...
                'emboss', 'high-pass', 'h-line', 'v-line', ...
                'gaussian', 'scharr-x', 'scharr-y'};

subplot(4,4,1), imshow(myim2uint8(img)), title('original')
if ~exist('samples','dir'), mkdir('samples'); end
 imwrite(myim2uint8(img), fullfile('samples', ...
            ['original.png']));
for n = 1:numel(kernels)
    K  = kernels{n};
    out = zeros(H-2,W-2, 3);
    for i = 2:H-1
        for j = 2:W-1
            p = img(i-1:i+1,j-1:j+1,:);
            out(i-1,j-1, 1) = max(sum(p(:,:,1).*K,'all'), 0);
            out(i-1,j-1, 2) = max(sum(p(:,:,2).*K,'all'), 0);
            out(i-1,j-1, 3) = max(sum(p(:,:,3).*K,'all'), 0);
        end
    end
    subplot(4,4,n+1), imshow(myim2uint8(out)), title(filter_names{n})
    imwrite(myim2uint8(out), fullfile('samples', ...
            ['convolution_' filter_names{n} '.png']));
end
```




# CNN Load Data

```
imds = imageDatastore(fullfile("dataset", {'vehicles','non-vehicles'}), ...
        'LabelSource','foldernames', 'FileExtensions','.png');

imgCell   = readall(imds);
trainLabs = imds.Labels;
trainImgs = cat(4, imgCell{:});
```

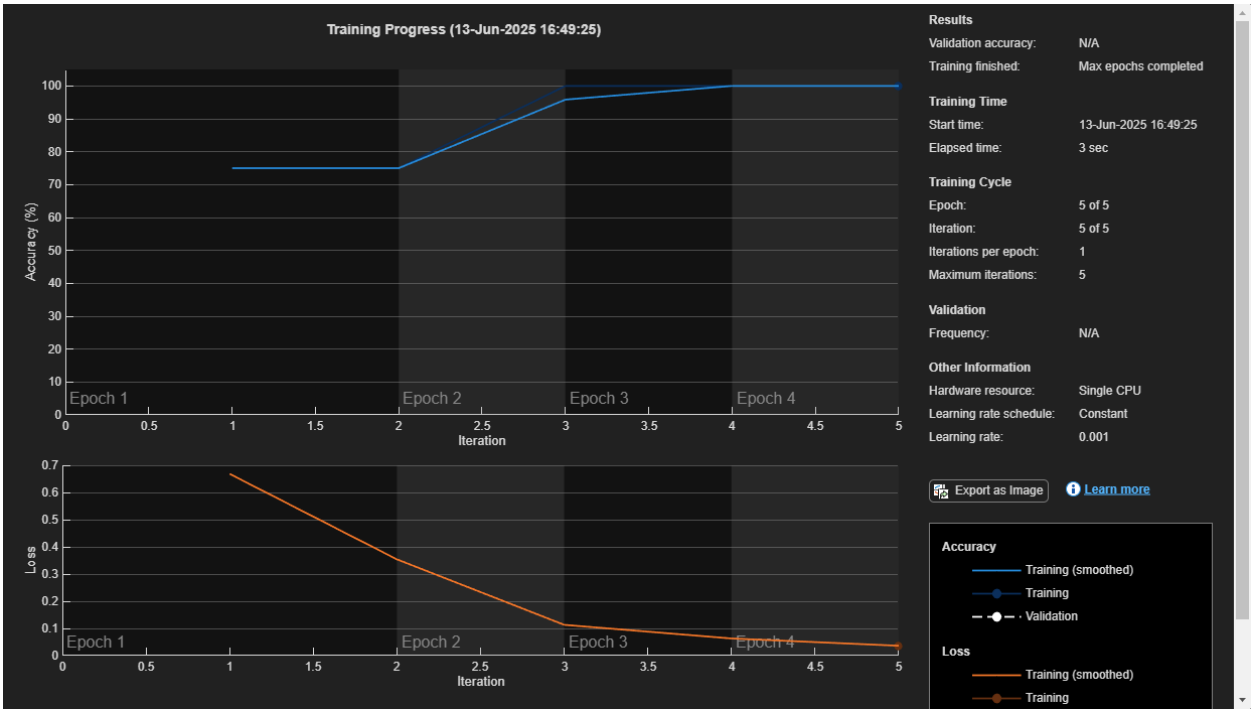
# CNN Architecture

```
layers = [
    imageInputLayer([64 64 3])
    convolution2dLayer(3, 8, "Padding","same")
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,"Stride",2)

    convolution2dLayer(3,16,"Padding","same")
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,"Stride",2)

    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer];

opts = trainingOptions("adam", ...
        "MaxEpochs",5, "MiniBatchSize",64, ...
        "Shuffle","every-epoch", ...
        "Plots","training-progress");
```

# CNN Training

```
net = trainNetwork(trainImgs, trainLabs, layers, opts);
```

```
Training on single CPU.
Initializing input data normalization.
|
========================================================================================
==========|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |
Base Learning  |
|         |             |    (hh:mm:ss)   |   Accuracy   |     Loss
|      Rate       |
|
========================================================================================
==========|
|       1 |           1 |       00:00:02 |       75.00% |       0.6688
|          0.0010 |
|       5 |           5 |       00:00:03 |      100.00% |       0.0363
|          0.0010 |
|
========================================================================================
==========|
Training finished: Max epochs completed.
```




# CNN Applying

```
predLabs = classify(net, imds);
```

```
tbl = table(imds.Files, imds.Labels, predLabs, ...
            'VariableNames',{'file','true','pred'});
writetable(tbl,'cnn_results.csv')
```

# References

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016, p. 330, para. 1. ISBN: 978-0-262-03561-3.

[2] A. Karpathy, J. Johnson, and L. Fei-Fei, CS231n: Convolutional Neural Networks for Visual Recognition. [Online]. Available: https://cs231n.stanford.edu/. [Accessed: Jun 9, 2025].

[3] G. Sanderson, "But What Is a Convolution?," **YouTube**, Nov. 18, 2022. [Online]. Available: https://youtu.be/KuXjwB4LzSA. [Accessed: Jun 9, 2025].

[4] G. Sanderson, "discrete.py," **GitHub**, 2022. [Online]. Available: https://github.com/3b1b/videos/blob/master/_2022/convolutions/discrete.py [Accessed: Jun 9, 2025].

*Published with MATLAB® R2025a*