Shape detection

by methods of Feature Extraction and CNN

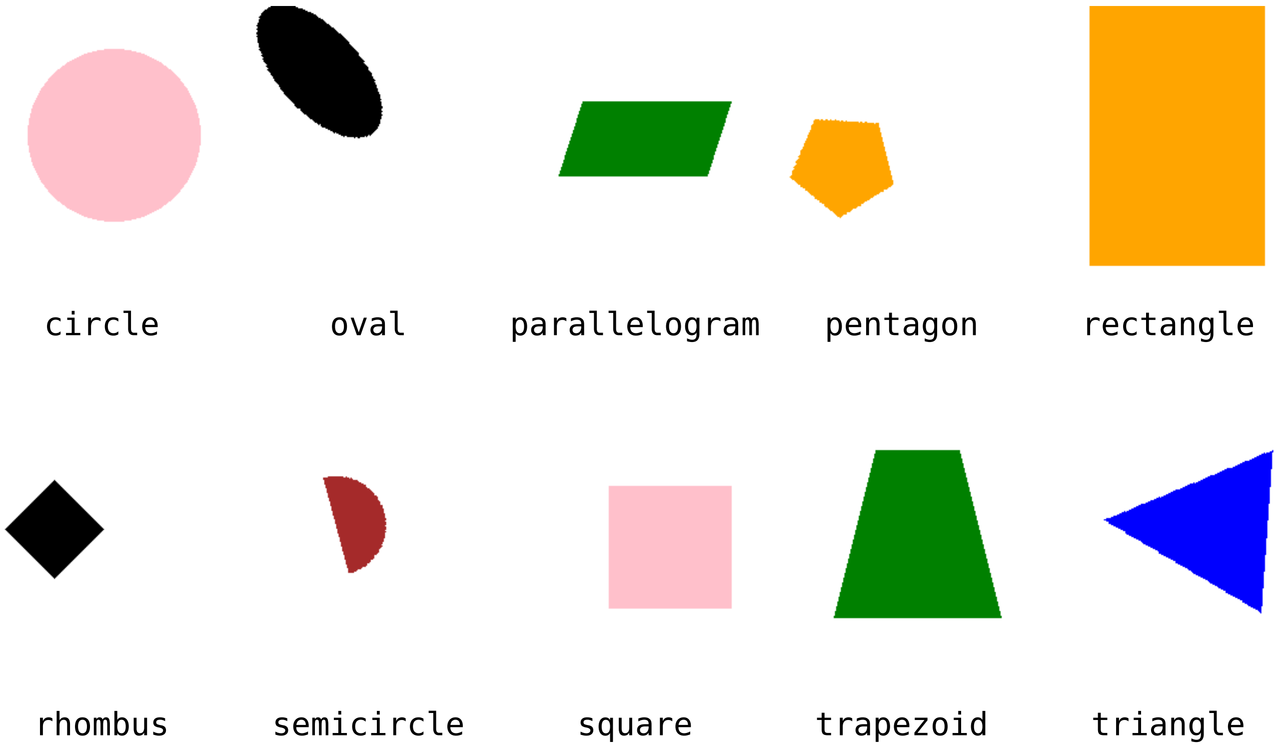
Sparsha Ray

Rishi Vora

IDC409 Project

8th November 2025

1.1 Samples


circle
oval
parallelogram
pentagon
rectangle
rhombus
semicircle
square
trapezoid
triangle

1.2 Stats

- 10 labels
- 12000 images for each label
- All images are 224x224 pixels.
- Each class contains shapes that are rotated, scaled and differently coloured.

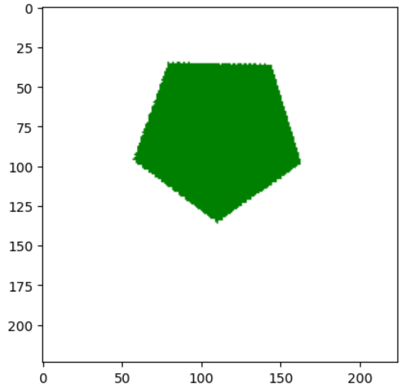
2.1 Overview

- The core idea is to extract the feature vector manually and then feed it to a classifier.
- The features that we chose are:
  - Number of corners
  - Solidity
  - Aspect ratio
  - Circularity
  - Hu Moments
- For classification, we chose Random Forests Classifier.

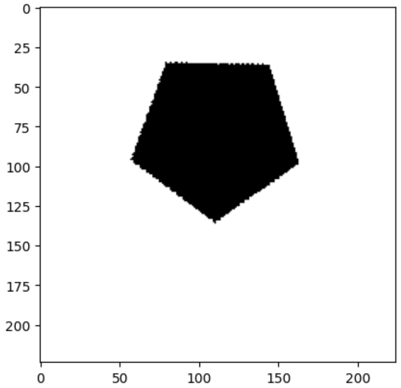
We use **OpenCV** for all image manipulation.

2.2 Preprocessing

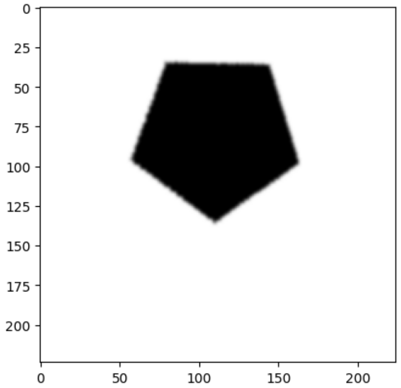
- Read the image
- Make it gray-scale
- Blur it to reduce noise
- Threshold it to make it binary



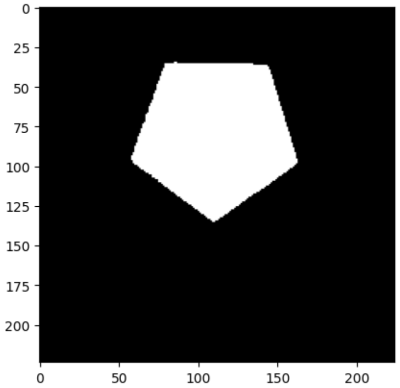
original



grayscaled

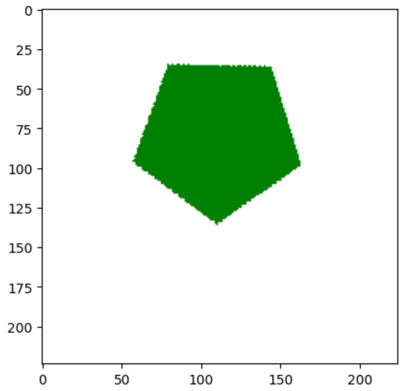


blurred

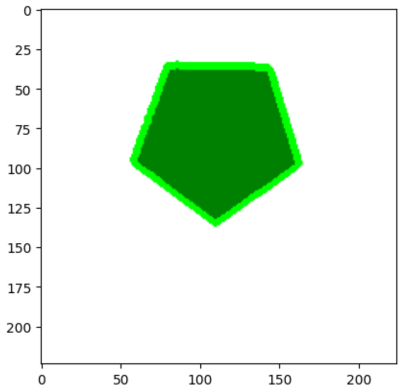


binary thresholded

2.3 Finding contour



original



with contour

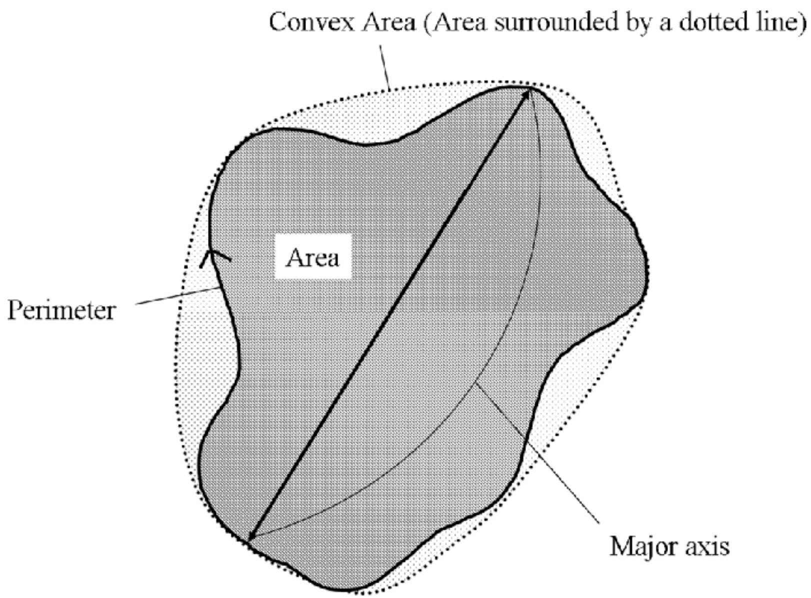
2.4 Feature extraction

1. Number of corners

- great first filter
- can categorize shapes into broad groups like:
  - `triangles` (3 corners)
  - `squares` + `rectangles` + `parallelogram` + `rhombus` (4 corners)
  - `semicircle` (2 corners)
  - `circle` + `oval` (no corners)

2.4 Feature extraction

2. Solidity

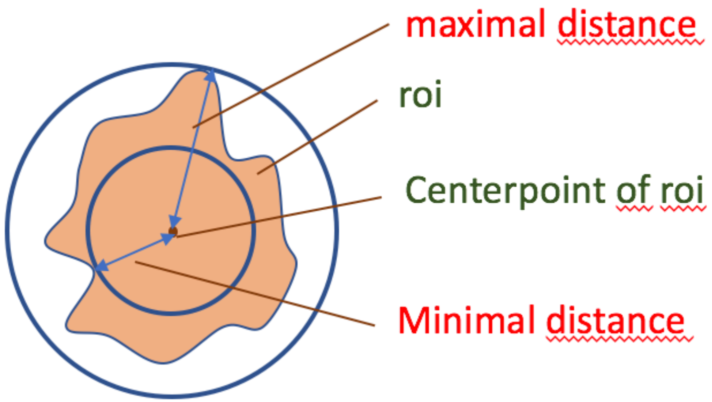


$$\text{Solidity} = \frac{\text{Convex hull area}}{\text{Area}}$$

- All the shapes we chose have solidity 1.0 but it is a great filter for concave shapes like stars.

2.4 Feature extraction

3. Circularity (Roundness)



$$\text{Circularity} = \frac{4\pi * \text{Area}}{\text{Perimeter}^2}$$

- A perfect `circle` will have a score of exactly 1.0.

2.4 Feature extraction

4. Aspect Ratio

- We draw the tightest upright box around the shape and divides its `width` by its `height`.

2.4 Feature extraction

4. Aspect Ratio

- We draw the tightest upright box around the shape and divides its `width` by its `height`.
- `square` and `circle` will have an aspect ratio of 1.0.
- `rectangle` and `oval` will have an aspect ratio not equal to 1.0.

2.4 Feature extraction

4. Aspect Ratio

- We draw the tightest upright box around the shape and divides its `width` by its `height`.
- `square` and `circle` will have an aspect ratio of 1.0.
- `rectangle` and `oval` will have an aspect ratio not equal to 1.0.
- But this can be easily fooled by rotation.

2.4 Feature extraction

5. Hu Moments

- Set of 7 numbers that can be considered unique “mathematical fingerprint” for a shape’s geometry.
- Derived from the field of Statistics

The central moments are defined as

$$\mu_{ij} = \sum_x \sum_y (x - \bar{x})^i (y - \bar{y})^j \, I(x, y)$$

where (x, y) are pixel coordinates, $(\bar{x}, \bar{y})$ is the centroid of the shape, and $I(x, y)$ are the intensity values.

Central moments are translation invariant.

2.4 Feature extraction

Using these, we get normalized central moments

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{\left(1+\frac{i+j}{2}\right)}}$$

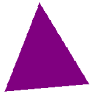
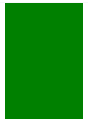
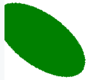
These are invariant to scaling and translation both.

Using these, M. K. Hu, derived a set of 7 moments that are invariant to translation, scaling, and rotation.

2.4 Feature extraction

$$
\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 &= (\eta_{20} - 3\eta_{12})^2 + (3\eta_{11} - \eta_{02})^2 \\
I_4 &= (\eta_{20} - \eta_{12})^2 + (\eta_{21} + \eta_{13})^2 \\
I_5 &= (\eta_{20} - 3\eta_{12})(\eta_{20} + \eta_{12})\left[(\eta_{20} + \eta_{12})^2 - 3(\eta_{21} + \eta_{13})^2\right] \\
&\quad + (3\eta_{21} - \eta_{13})(\eta_{21} + \eta_{13})\left[3(\eta_{20} + \eta_{12})^2 - (\eta_{21} + \eta_{13})^2\right] \\
I_6 &= (\eta_{20} - \eta_{12})(\eta_{20} + \eta_{12})^2 - (\eta_{21} + \eta_{33})^2] \\
&\quad + 4\eta_{11}(\eta_{20} + \eta_{12})(\eta_{21} + \eta_{13}) \\
I_7 &= (8\eta_{21} - \eta_{30})(\eta_{30} + \eta_{22})\left[(\eta_{30} + \eta_{22})^2 - 3(\eta_{21} + \eta_{33})^2\right] \\
&\quad - (\eta_{30} - 3\eta_{22})(\eta_{21} + \eta_{33})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{33})^2\right]
\end{aligned}
$$

2.4 Feature extraction

Images	# H[1]	# H[2]	# H[3]	# H[4]	# H[5]	# H[6]	# H[7]	images
triangle	0.711	3.114	2.334	4.563	6.96	6.068	7	
pentagon	0.791	5.32	6.332	7	-7	-7	-7	
rectangle	0.744	2.329	7	7	-7	7	7	
circle	0.798	7	7	7	-7	7	7	
trapezoid	0.753	2.831	2.999	6.745	7	6.987	7	
oval	0.747	2.203	4.935	6.275	-7	-6.873	7	
semicircle	0.691	1.882	2.905	4.12	-6.909	-5.058	-6.993	

2.5 Training

- Now we have 11 features (corners, solidity, aspect ratio, circularity, and 7 Hu moments.)
- We train a Random Forests Classifier with
  - 100 estimators
  - 120k samples
  - 80% training, 20% testing samples

2.6 Evaluation

2.6 Evaluation

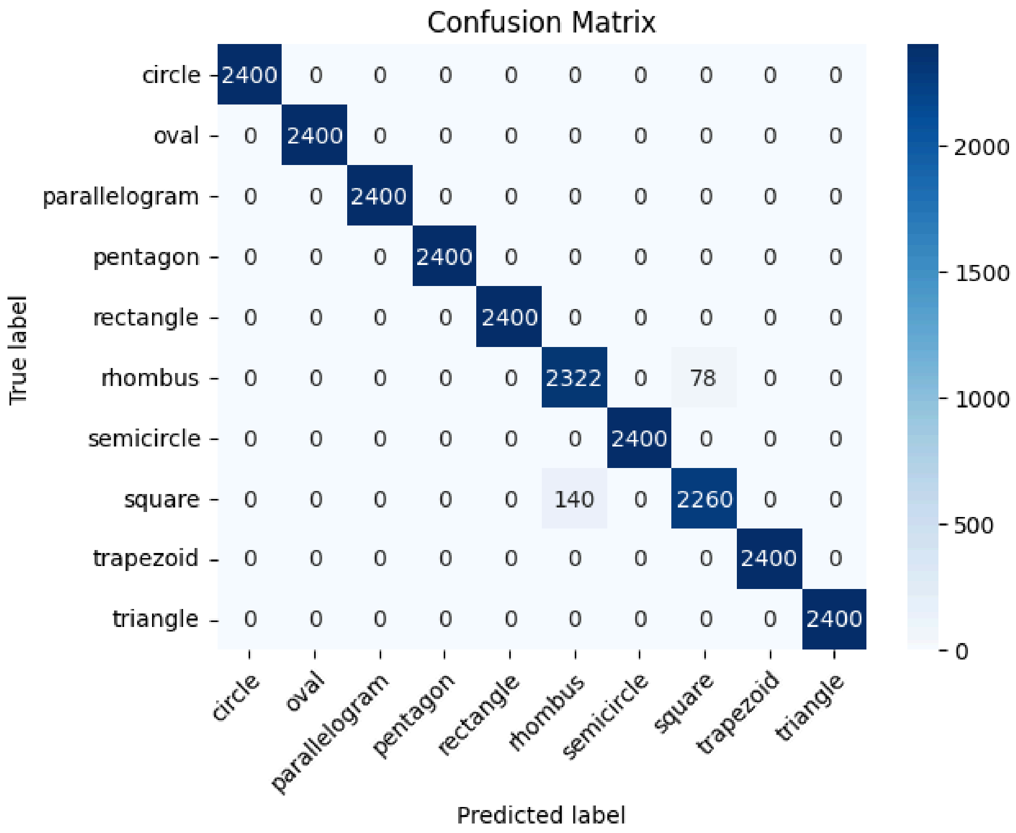
- Model Accuracy: **99.11%**

2.6 Evaluation

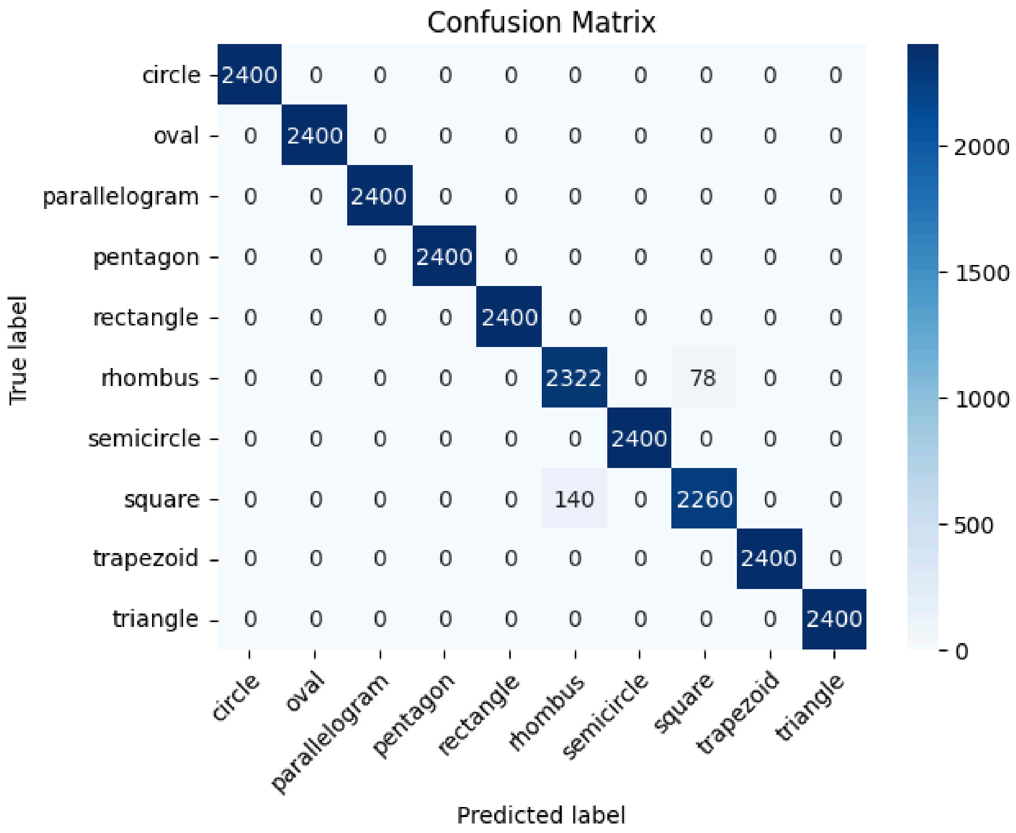
- Model Accuracy: **99.11%**

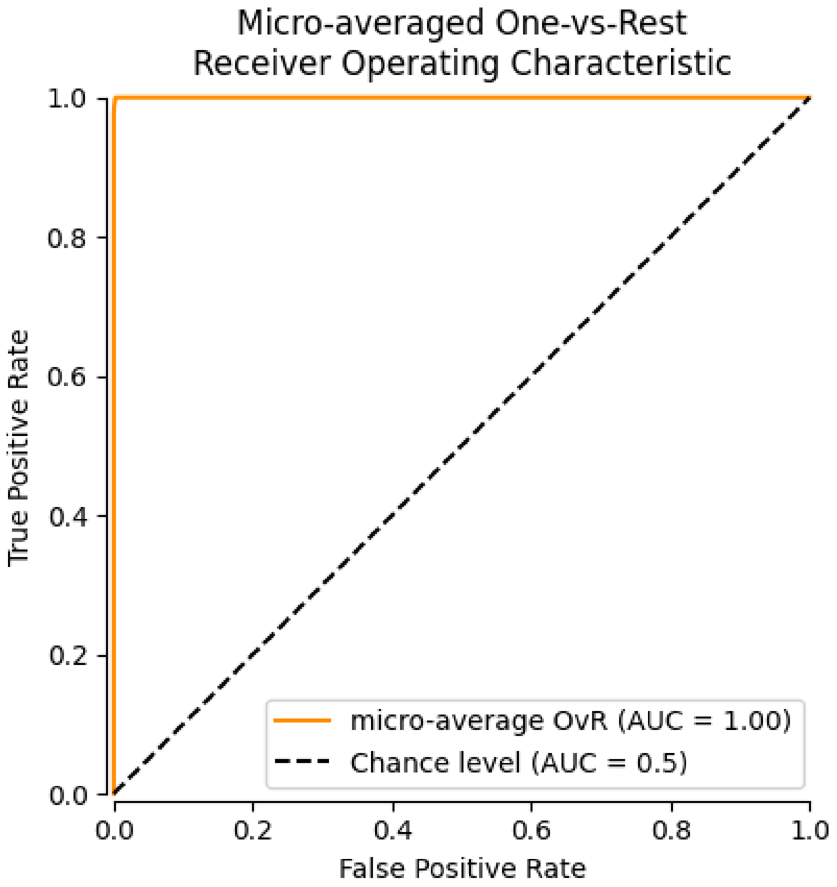
	precision	recall	f1-score	support
parallelogram	1.00	1.00	1.00	2400
triangle	1.00	1.00	1.00	2400
pentagon	1.00	1.00	1.00	2400
rectangle	1.00	1.00	1.00	2400
square	0.97	0.94	0.95	2400
circle	1.00	1.00	1.00	2400
trapezoid	1.00	1.00	1.00	2400
oval	1.00	1.00	1.00	2400
semicircle	1.00	1.00	1.00	2400
rhombus	0.94	0.97	0.96	2400
accuracy			0.99	24000
macro avg	0.99	0.99	0.99	24000
weighted avg	0.99	0.99	0.99	24000

2.6 Evaluation



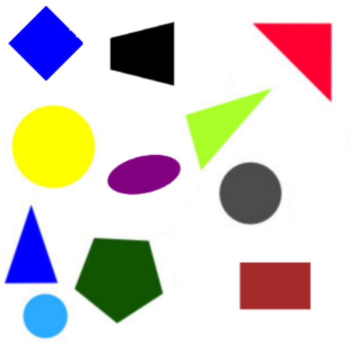
2.6 Evaluation





2.6 Evaluation

We also made a function recognize all present shapes from an image, by extracting all contours.



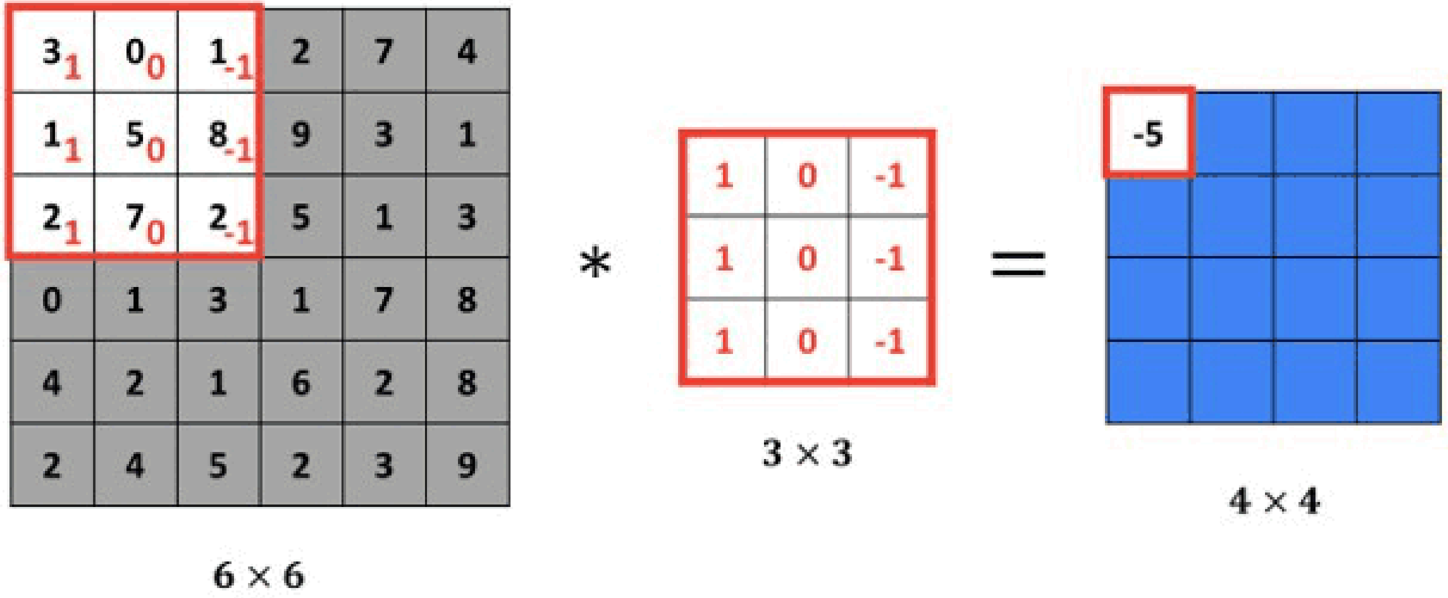
```python
all_shapes_features = extract_features_from_all_shapes(
    multi_shape_image, threshold_value=250
)
predicted_shapes = model.predict(all_shapes_features)
Counter(predicted_shapes)
```

```
Counter({'circle': 3, 'triangle': 3, 'pentagon': 2, 'rectangle': 1, 'oval': 1, 'trapezoid': 1})
```

It mostly correctly identified all the shapes.

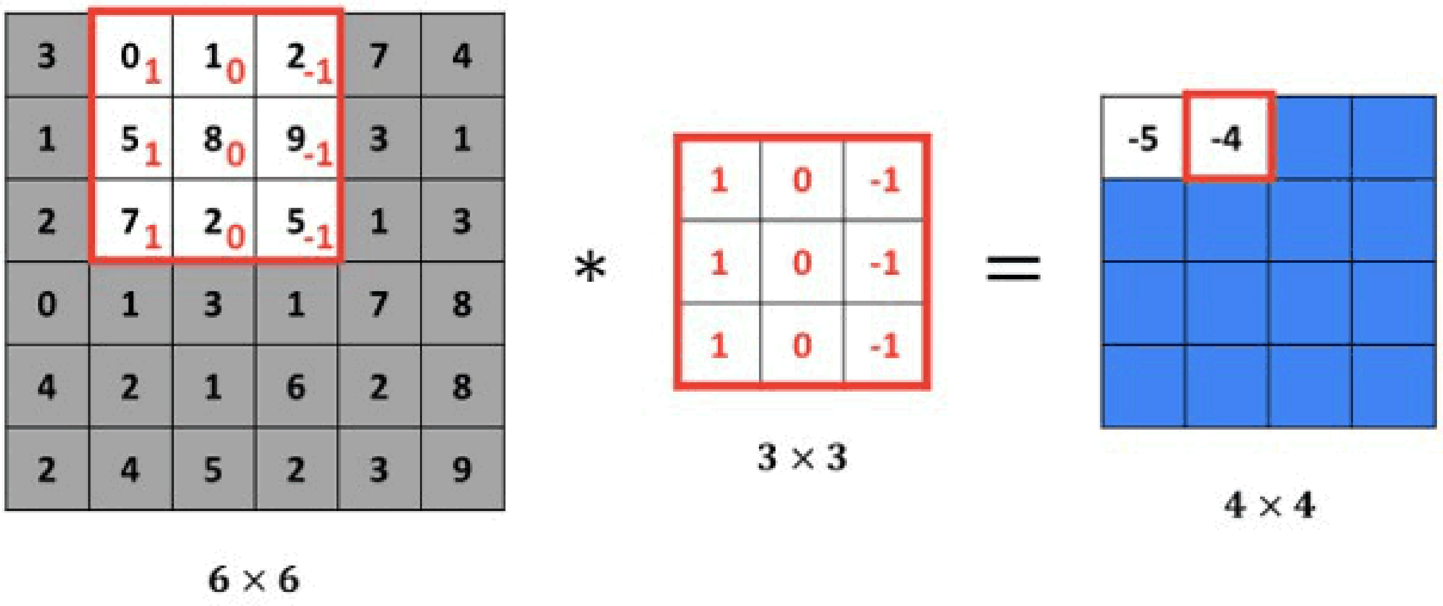
3.1 What is convolution ?

3.1 What is convolution ?



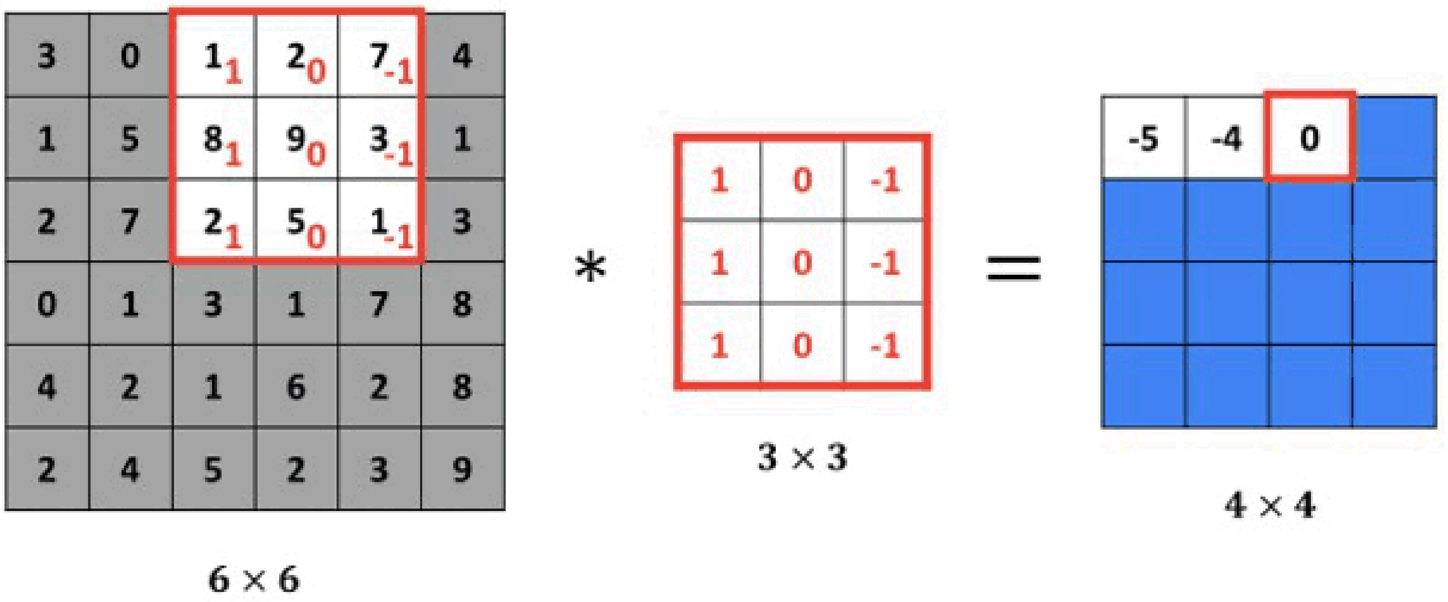
$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

3.1 What is convolution ?



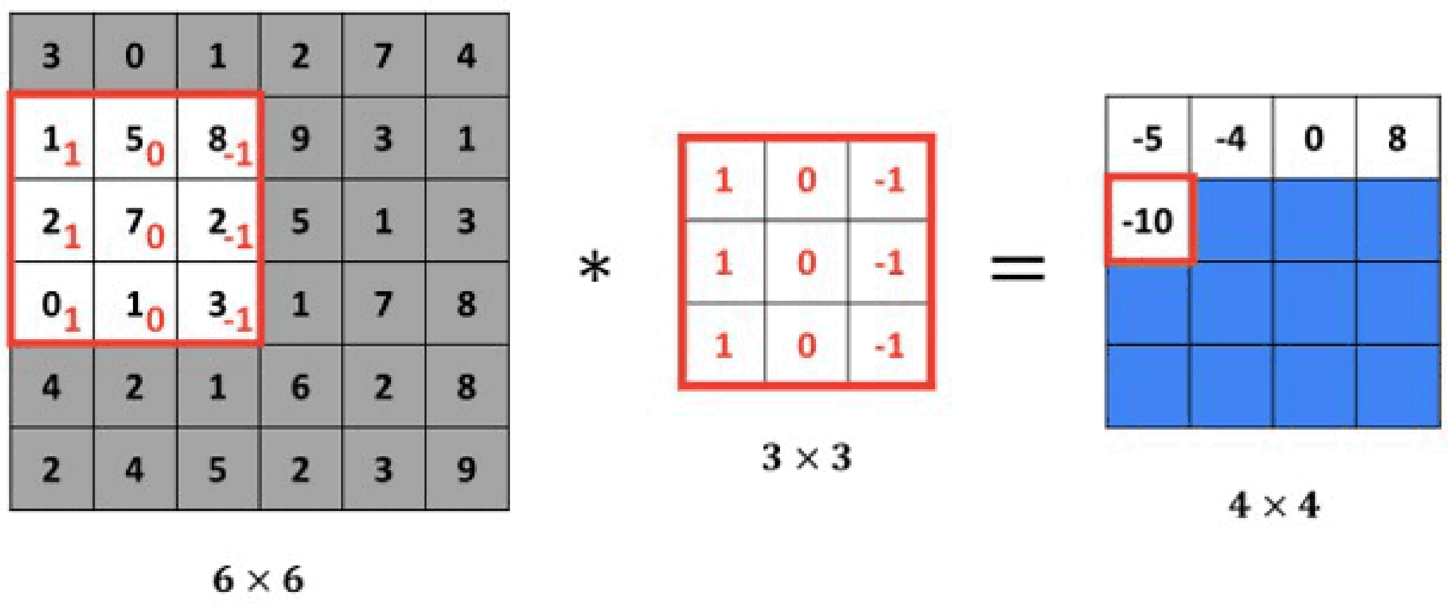
$$0 \times 1 + 5 \times 1 + 7 \times 1 + 1 \times 0 + 8 \times 0 + 2 \times 0 + 2 \times -1 + 9 \times -1 + 5 \times -1 = -4$$

3.1 What is convolution ?



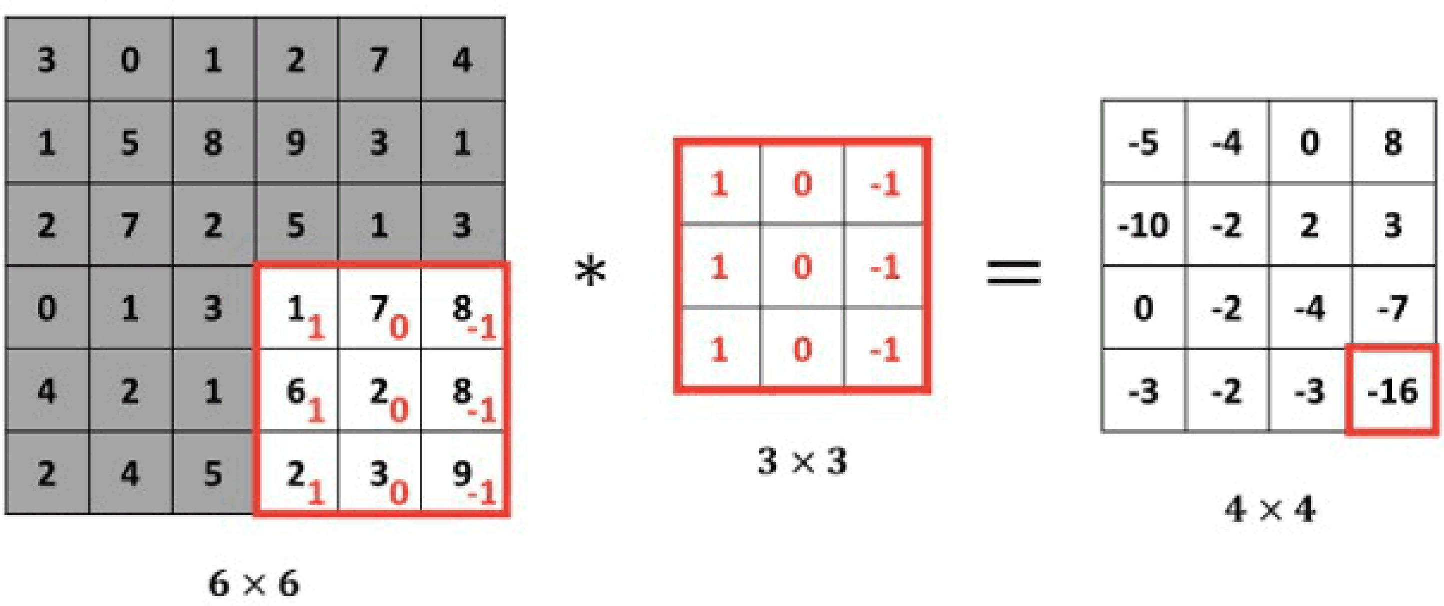
$$1 \times 1 + 8 \times 1 + 2 \times 1 + 2 \times 0 + 9 \times 0 + 5 \times 0 + 7 \times -1 + 3 \times -1 + 1 \times -1 = 0$$

3.1 What is convolution ?



$$1 \times 1 + 2 \times 1 + 0 \times 1 + 5 \times 0 + 7 \times 0 + 1 \times 0 + 8 \times -1 + 2 \times -1 + 3 \times -1 = -10$$

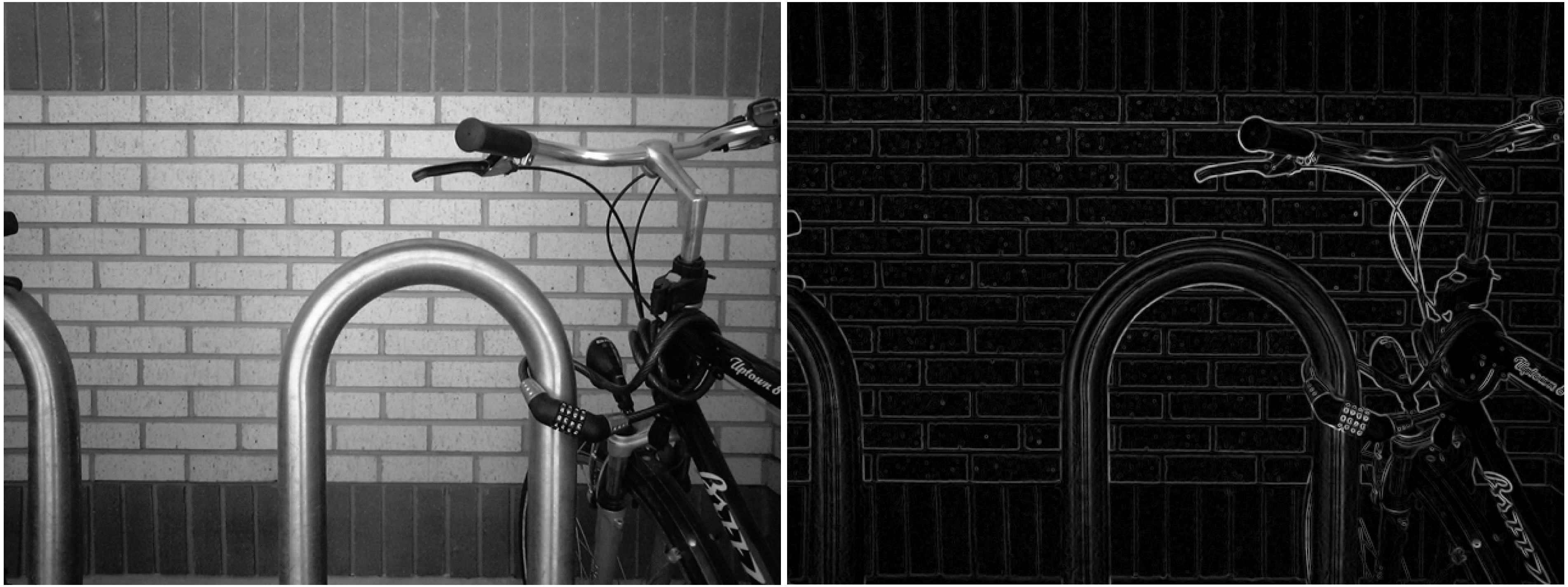
3.1 What is convolution ?



$$1 \times 1 + 6 \times 1 + 2 \times 1 + 7 \times 0 + 2 \times 0 + 3 \times 0 + 8 \times -1 + 8 \times -1 + 9 \times -1 = -16$$

3.2 What are filters ?

Consider the filter (or kernel) $\begin{pmatrix} -3+3i & +10i & 3+3i \\ -10 & 0 & 10 \\ -3-3i & -10i & 3-3i \end{pmatrix}$



3.3 How does CNNs work ?

3.3 How does CNNs work ?

- In CNNs, a image is passed through a series of filters (which the learnt as the model is trained) …

3.3 How does CNNs work ?

- In CNNs, a image is passed through a series of filters (which the learnt as the model is trained) …
- Which extract features like edges corners etc (called feature maps) …

3.3 How does CNNs work ?

- In CNNs, a image is passed through a series of filters (which the learnt as the model is trained) …
- Which extract features like edges corners etc (called feature maps) …
- As the feature maps are down-sampled and passed down through more and more layers of filters, specific filters starts to learn more complex higher order characteristics (like curvature, spacing between corners, parallel edges etc)

3.3 How does CNNs work ?

- In CNNs, a image is passed through a series of filters (which the learnt as the model is trained) …
- Which extract features like edges corners etc (called feature maps) …
- As the feature maps are down-sampled and passed down through more and more layers of filters, specific filters starts to learn more complex higher order characteristics (like curvature, spacing between corners, parallel edges etc)
- When these features are extracted, and the feature maps consists of only a few pixels, their values are passed down to a fully connected feed forward neural network (often with only 2 or 3 layers) which does the final classification task.

3.4 Architecture

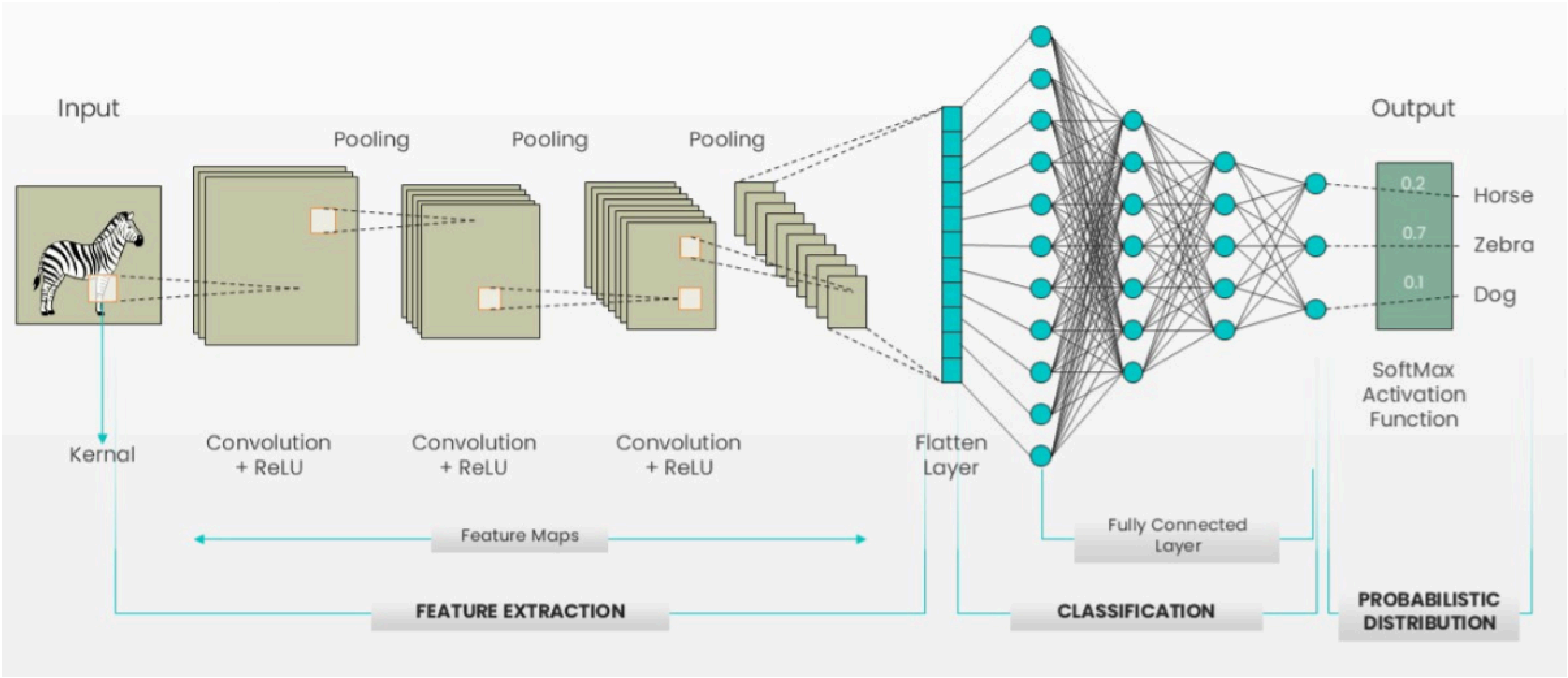


Image courtesy of Dwika Sudrajat's blog post

3.4 Architecture

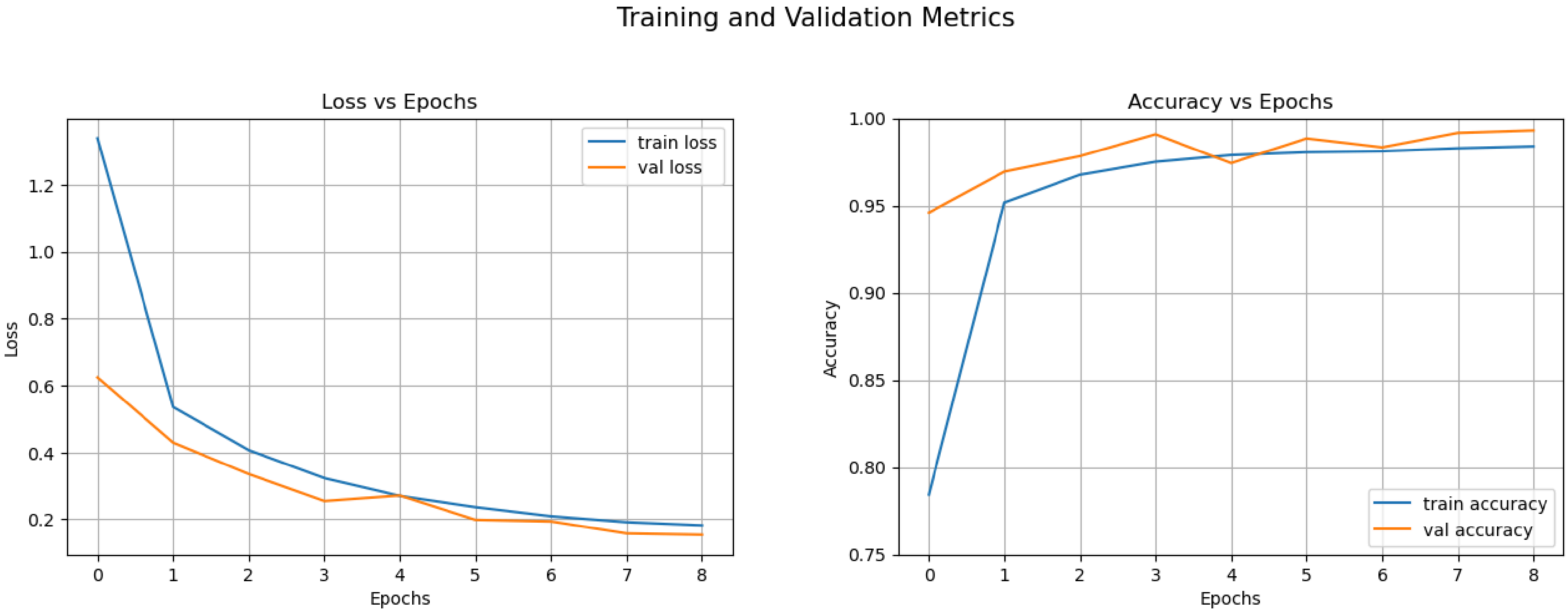
Layer (type)	Output Shape	Parameters
conv2d_5 (Conv2D)	(None, 124, 124, 8)	80
max_pooling2d_5 (MaxPooling2D)	(None, 62, 62, 8)	0
conv2d_6 (Conv2D)	(None, 60, 60, 16)	1,168
max_pooling2d_6 (MaxPooling2D)	(None, 30, 30, 16)	0
conv2d_7 (Conv2D)	(None, 28, 28, 32)	4,640
max_pooling2d_7 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_8 (Conv2D)	(None, 12, 12, 64)	18,496
max_pooling2d_8 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_9 (Conv2D)	(None, 4, 4, 128)	73,856
max_pooling2d_9 (MaxPooling2D)	(None, 2, 2, 128)	0

3.4 Architecture

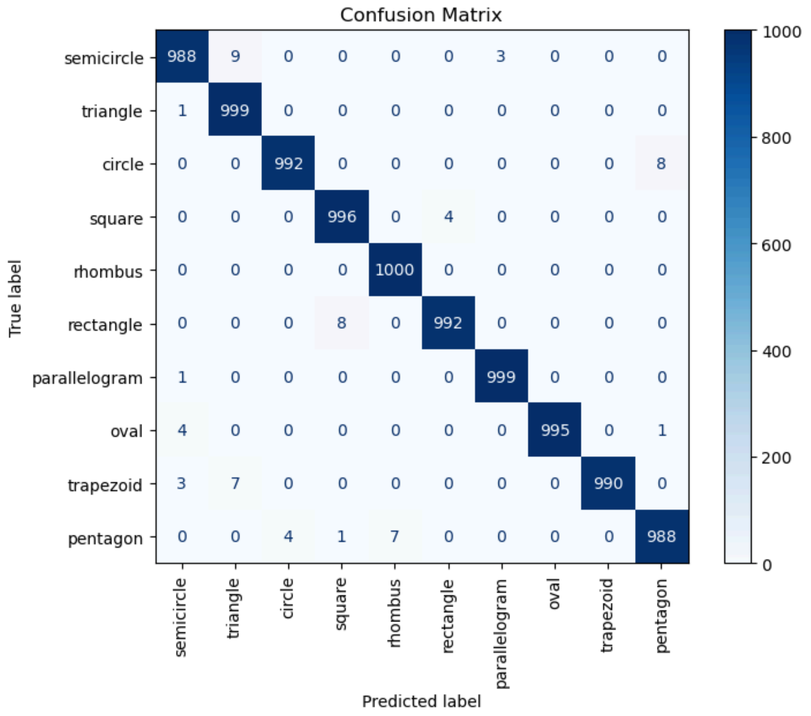
Layer (type)	Output Shape	Parameters
flatten_1 (Flatten)	(None, 512)	0
dense_3 (Dense)	(None, 128)	65,664
dense_4 (Dense)	(None, 64)	8,256
dense_5 (Dense)	(None, 10)	650

3.5 Training

Training is done with the `Adam` optimizer for 9 epochs only

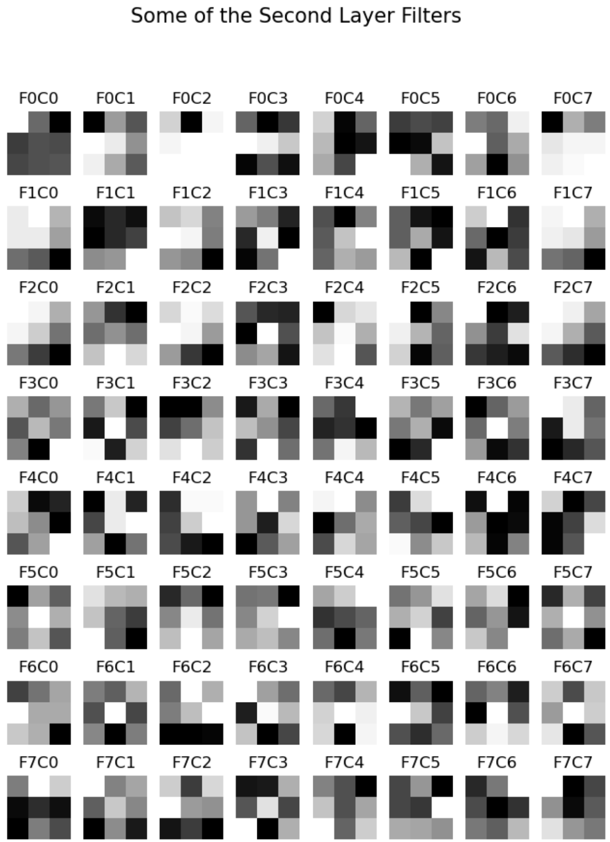


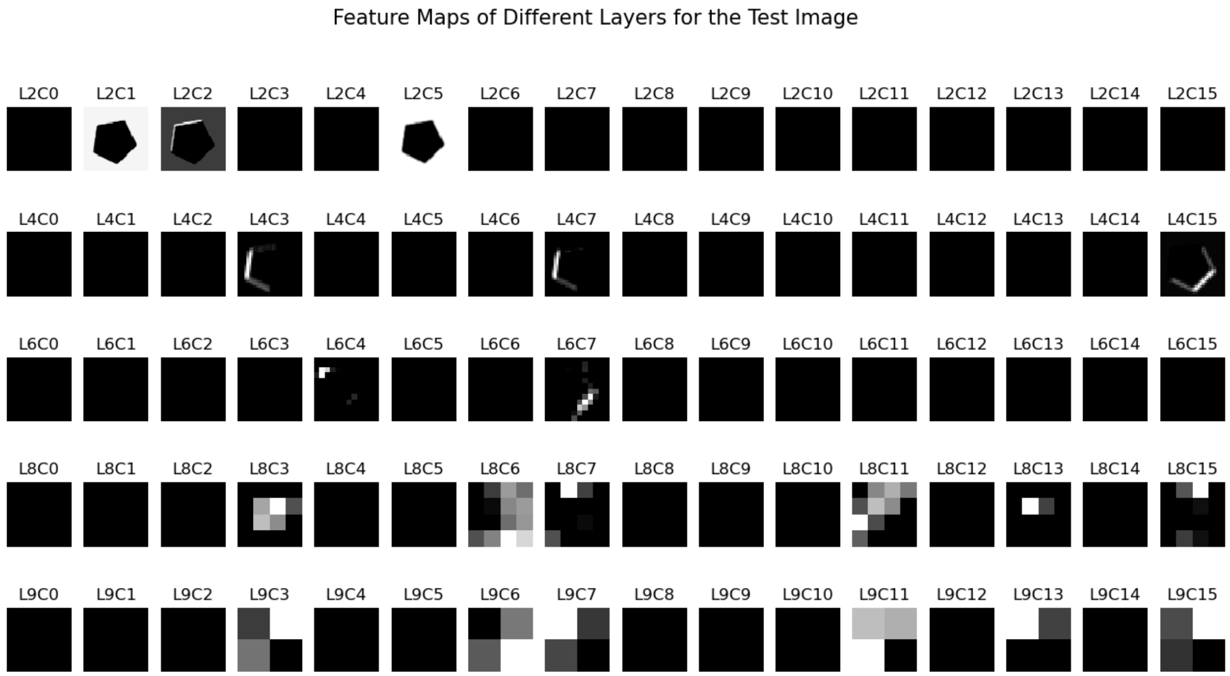
3.6 Results



Class	**Precision**	**Recall**	**f1-score**
semicircle	0.99	0.99	0.99
triangle	0.98	1.00	0.99
circle	1.00	0.99	0.99
square	0.99	1.00	0.99
rhombus	0.99	1.00	1.00
rectangle	1.00	0.99	0.99
parallelogram	1.00	1.00	1.00
oval	1.00	0.99	1.00
trapezoid	1.00	0.99	0.99
pentagon	0.99	0.99	0.99
Overall Accuracy			0.99

3.7 Interpretations





4.1 How does these two methods compare

- Both reaches similar accuracy, CNN is slower, Feature Extraction + Random Forests (RF) is much faster
- CNN learns the features by itself, FE+RF is given the features to classify
- FE+RF doesn't generalize well for complex shapes, CNN does

Thank you!

Questions?