Week 5

Deep Learning II

Convolutional Neural Networks

Dr Anagi Gamachchi

Discipline of Information Systems and Business Analytics,
Deakin Business School

AI Application at Woolworth

NEWS ANALYSIS MARKETS RETAIL SUPPLY TECHNOLOGY PACKAGING EVENTS SUBSCRIBE ADVERT

Woolworths scans fresh produce with AI

By Liam O'Callaghan | 18 February 2020

Woolworths is aiming to speed up scanning for shoppers with a new artificial intelligence (AI) powered scale that automatically identifies loose fresh produce.

The scales, developed by Australian company Tiliter, are currently being trialled in three Sydney stores in conjunction with the retailer's Scan&Go app.

Consumers can place their fruit or vegetables on the scale surface and then the technology will detect the type of produce and identify the variety.

Woolworths is aiming to speed up scanning for shoppers with a new artificial intelligence (AI) powered scale that automatically identifies loose fresh produce.

Source:https://www.fruitnet.com/produce-plus/woolworths-scans-fresh-produce-with-ai/180927.article

Discussion Question

How can Machine identify the correct type of fruits?

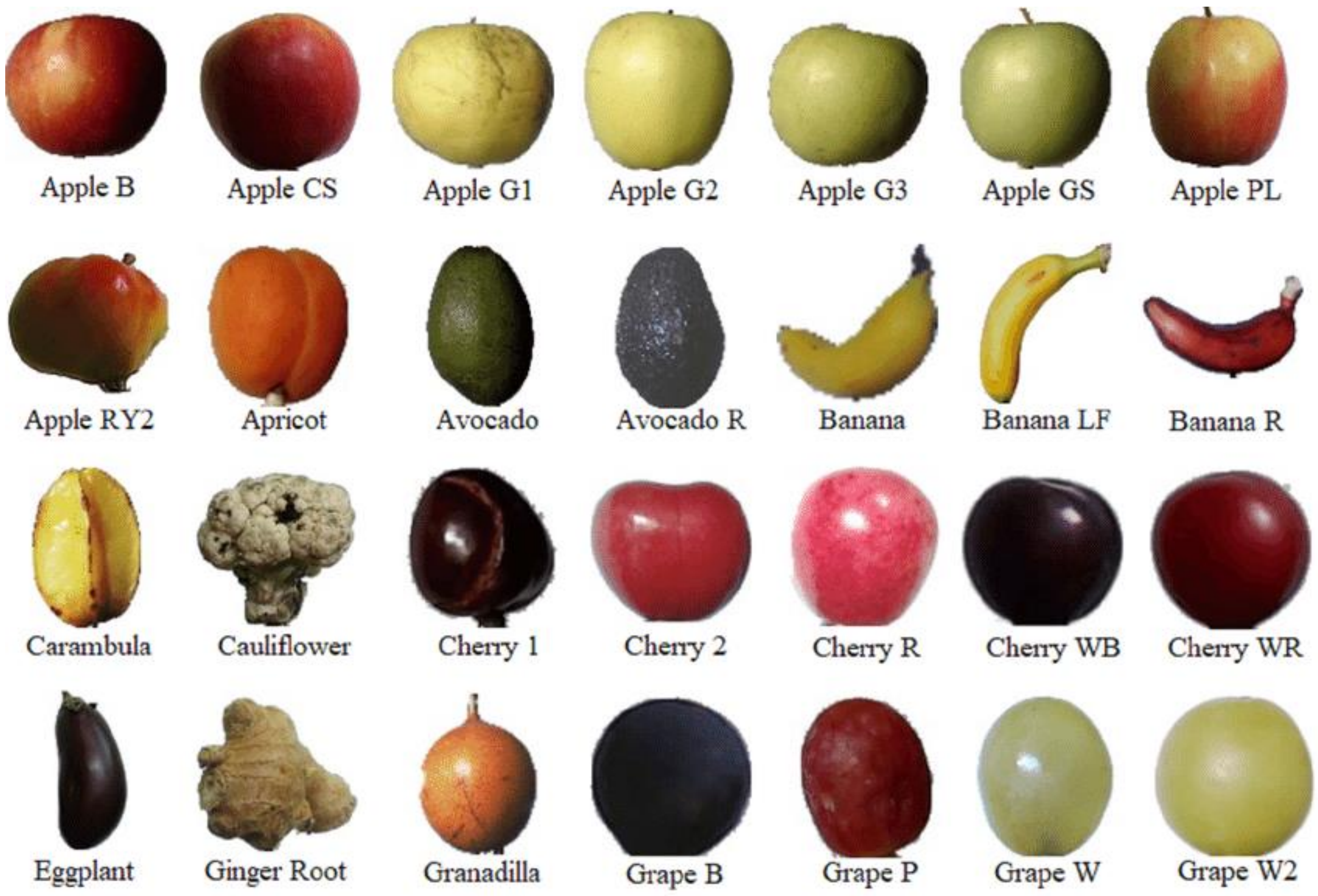
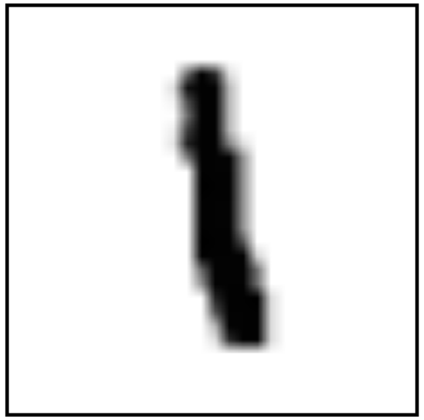


Image Representation

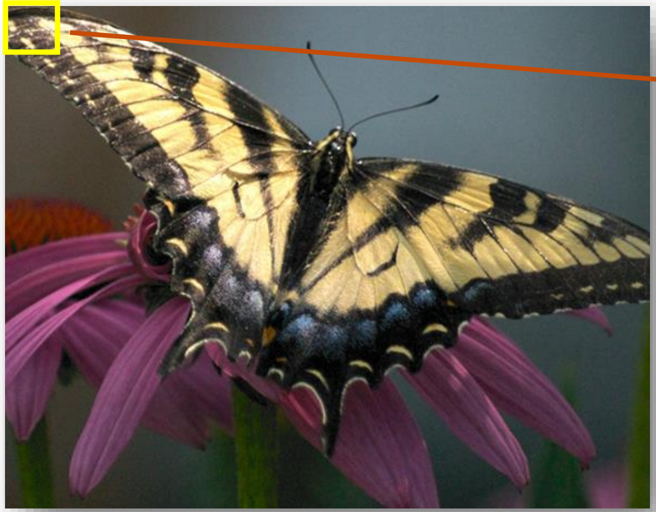
Representation of visual data (images) is based on matrices of pixels

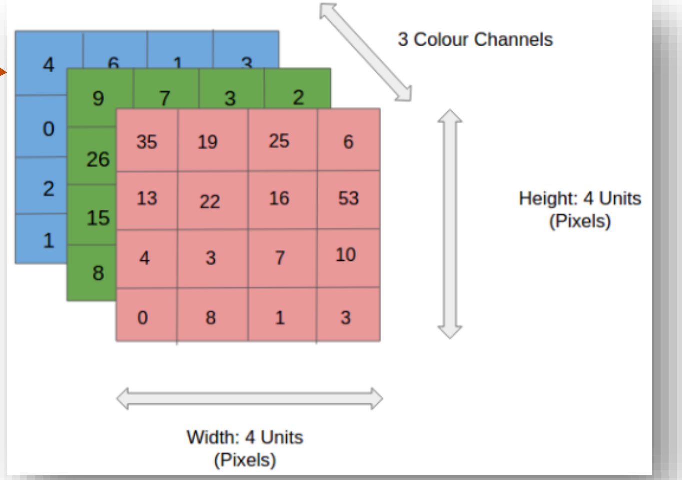
Pixel: Is the smallest unit of an image

- Has an address in the matrix (**row**, **column**)
- Has an intensity…
  - **Greyscale images**: **0** for black and 255 **for** white
  - **RGB images**: 256 shades of red, green, and blue colors



$$\simeq \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$





Resolution of the screen (image container): Is the size of the matrix in terms of the number of rows and columns (pixels)

How do Convolutional Neural Networks (CNN) work?

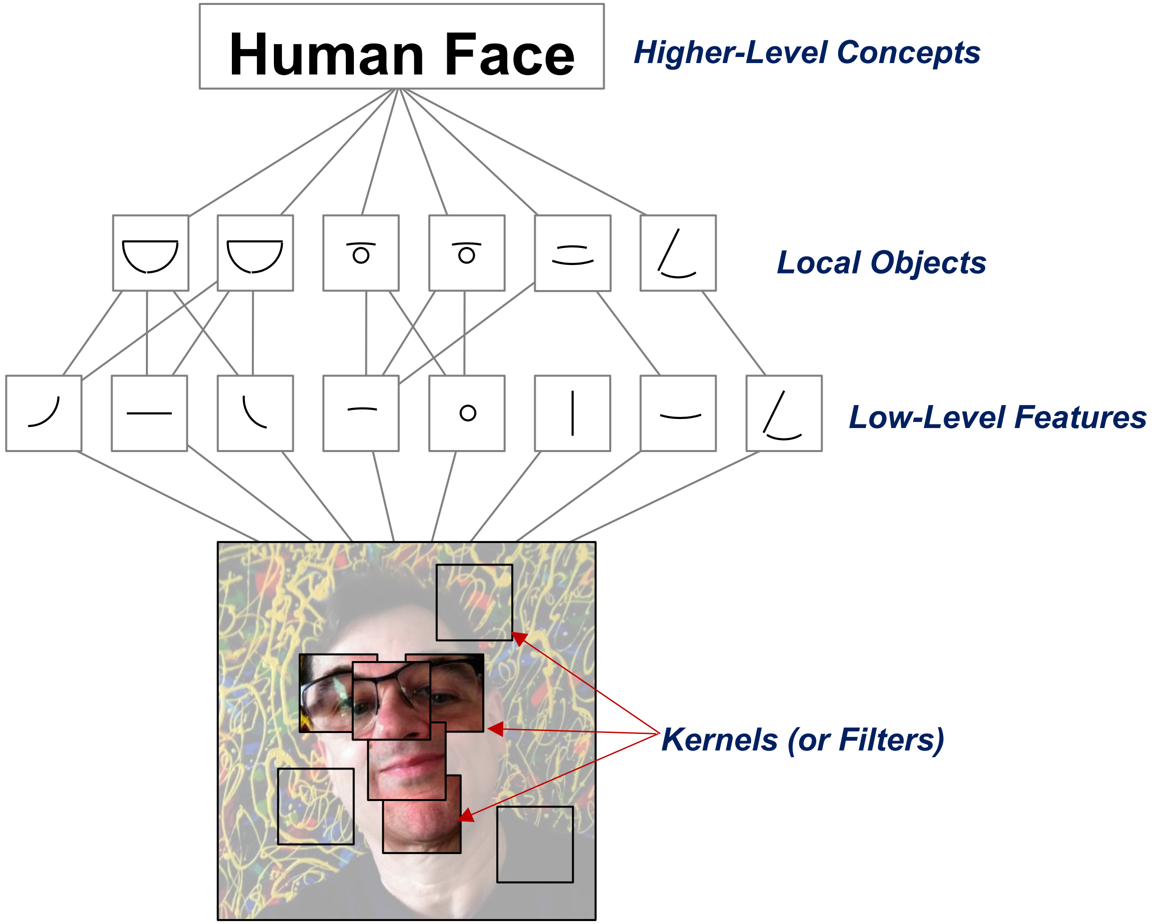
*The Deep Learning magic is in meaning (**representation**) forming at different neural network layers*

Images are scanned by tiny **filters** (also called **kernels**).

They are responsible for matching and identifying **low-level visual features**, such as edges of different direction, shades or colours.

Once detected at the next level these features are being combined into **local objects**, such as eyes, mouths, noses or glasses.

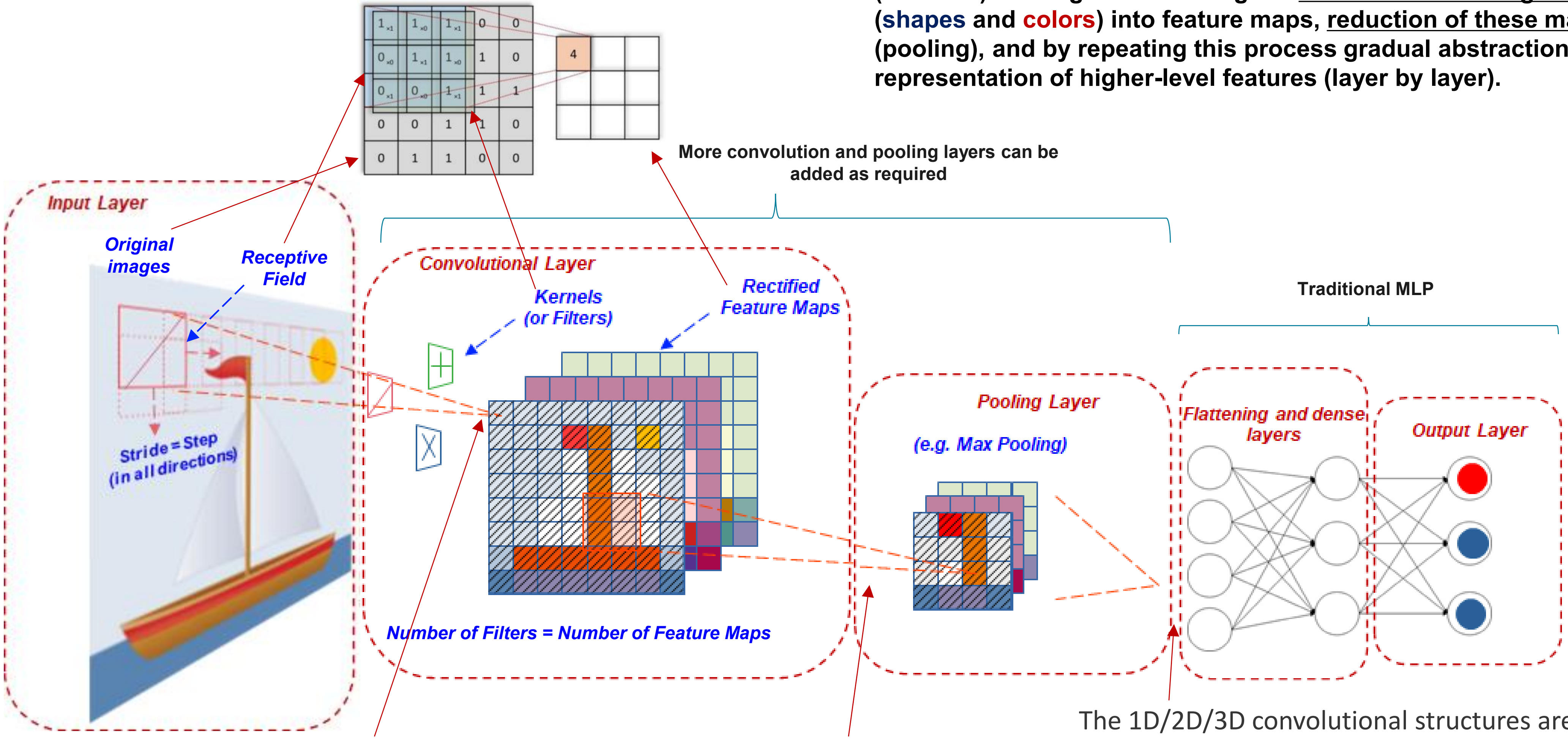
These local objects in turn combine into **high-level concepts**, such as a "human face".



A **kernel** is a numeric matrix, used to identify *lines* and *edges* in different direction, sharpening or blurring lines, identifying *colours*, etc.
Kernels are discovered automatically.

Architecture of CNN

CNNs **link different types of layers which aim at using small kernels (or filters) striding over the image to <u>extract various image features</u> (shapes and colors) into feature maps, <u>reduction of these maps</u> (pooling), and by repeating this process gradual abstraction and representation of higher-level features (layer by layer).**



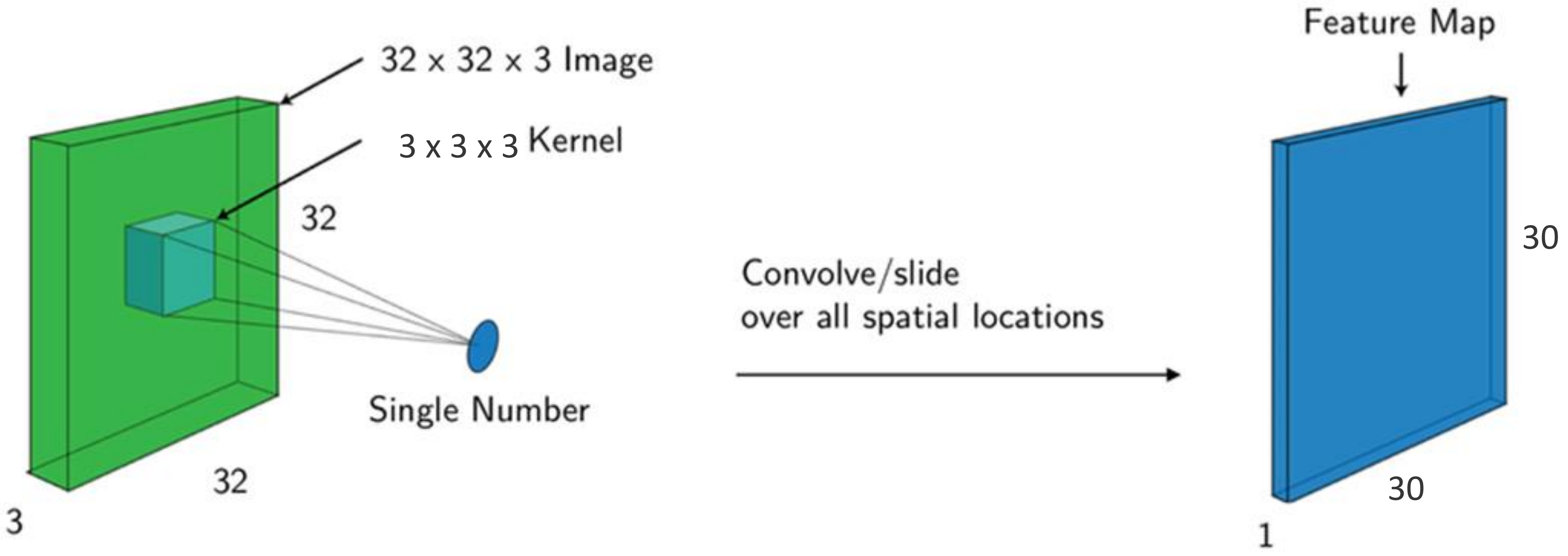
The **convolution operation** is the weighted sum of the <u>receptive field</u> and the <u>kernel</u> (usually 3x3 or 5x5), across all channels (RGB).

Max-pooling operation takes the maximum value from each channel within a window (usually 2x2) of the feature map, which is sliding over it (with a stride/step of 2 pixels), thus <u>*reducing the image dimension*</u> (by half).

The 1D/2D/3D convolutional structures are **flattened**, and a series of dense layers are used to output classification, estimation or decision making.

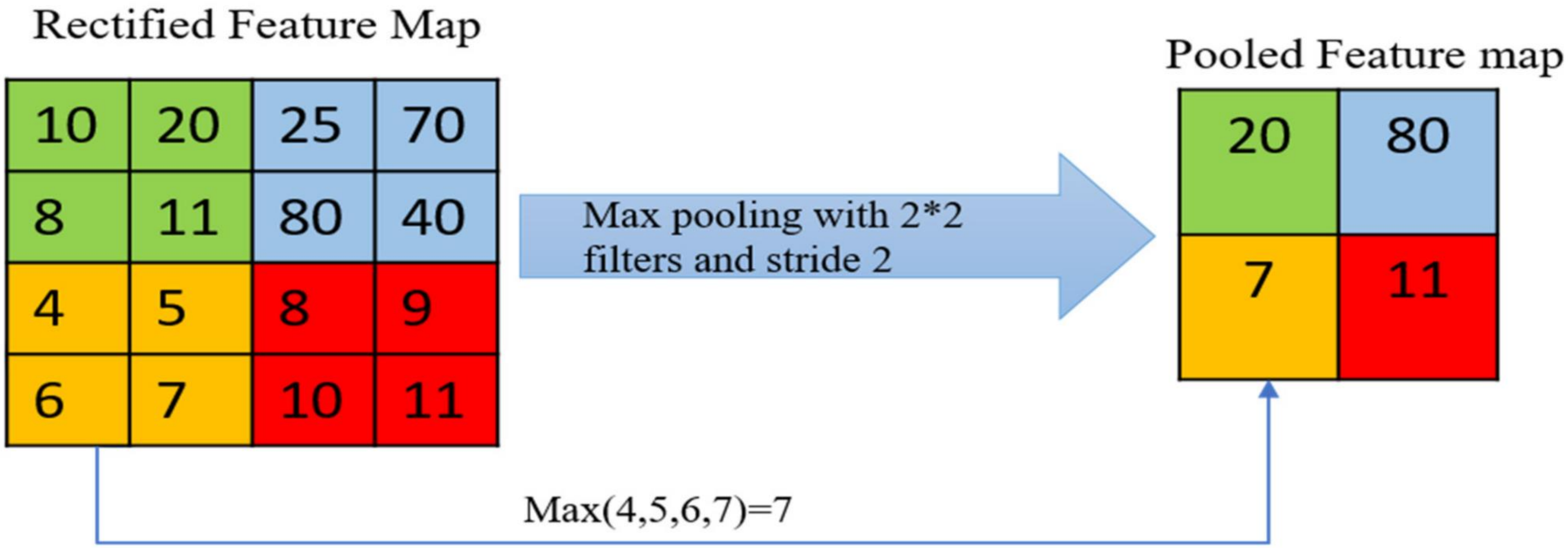
Convolution Layer

Stride = 1



https://www.sciencedirect.com/topics/engineering/convolutional-layer

Max Pooling



https://www.mdpi.com/2076-3417/12/17/8643

CNN Applications

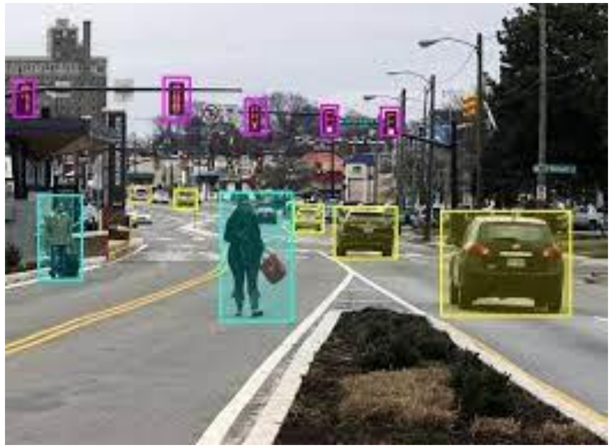
Applications of CNN include:

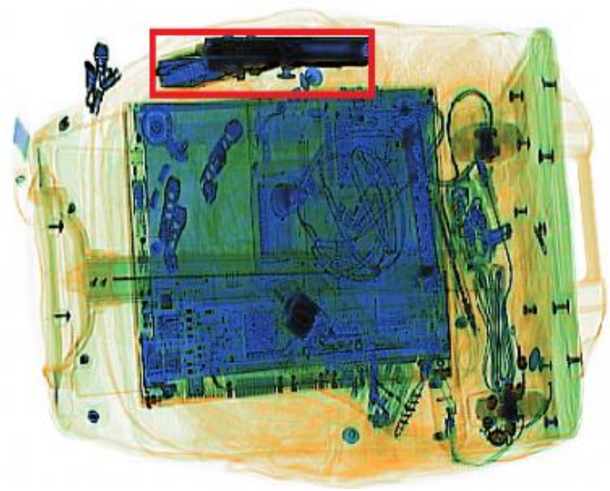
- Object Detection (e.g., fruits)
- Hand-writing recognition (e.g. transcription)
- Facial recognition (e.g. social media tagging)
- Emotion recognition (e.g. in lie detectors)
- Scene labelling (e.g. self-driving cars)
- Action recognition (e.g. suspicious behaviour)
- Motion detection (e.g. security cameras)
- MRI / CT diagnosis (e.g. cardiac imaging)
- Satellite image processing (e.g. for planning)
- Land feature recognition (e.g. missile systems)
- Echo-sound processing (e.g. oil prospecting)
- Object detection (e.g. airport scanners)
- Object tracking (e.g. players/ball in sport)
- Colouring and noise removal (e.g. photography)

...

Handwriting recognition for postal automation

Scene labelling self-driving car





Threaten Object-Detection in Bag Scanning at Airport

CNN in Python

❑ Data Set:

CIFAR10 data sets includes images of consists of 60000 **32x32** colour images in 10 classes, with 6000 images per class

Your task is to:

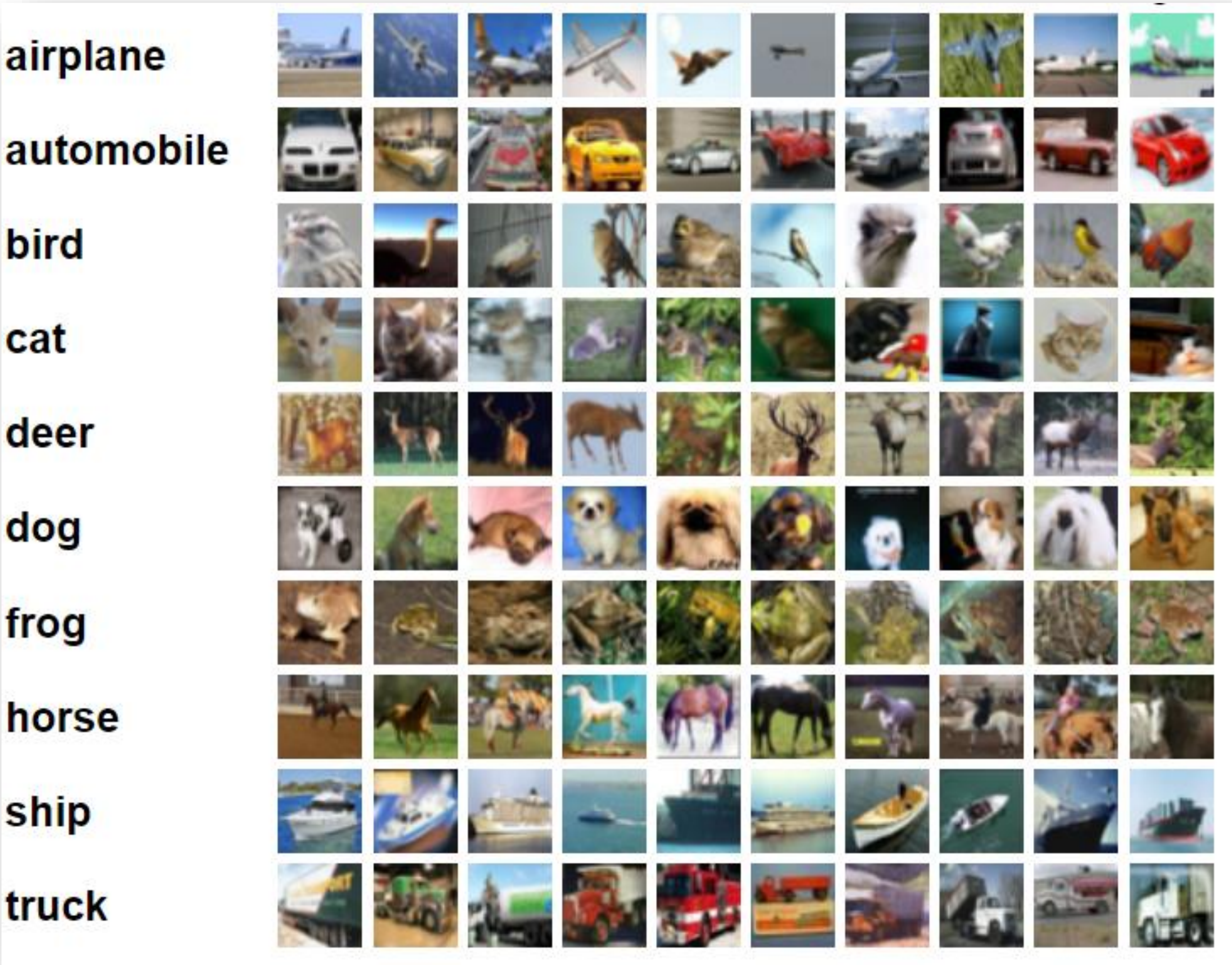
Recognize the corresponding object in the images.

❑ Tool:

Python + Tensorflow (with Keras)

❑ Method:

Convolutional Neural Nets (CNN)



EFMD EQUIS ACCREDITED
AACSB ACCREDITED
DEAKIN BUSINESS SCHOOL

CNN in Python

Data Loading

```python
from tensorflow.keras.datasets import cifar10

# Data parameters
img_rows, img_cols = 32, 32
channels = 3

num_classes = 10
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

```python
plot_images(x_train[0:20], cols=5,figsize=[7,7])
```

Data Preparation

```python
x_train /= 255
x_test /= 255
```

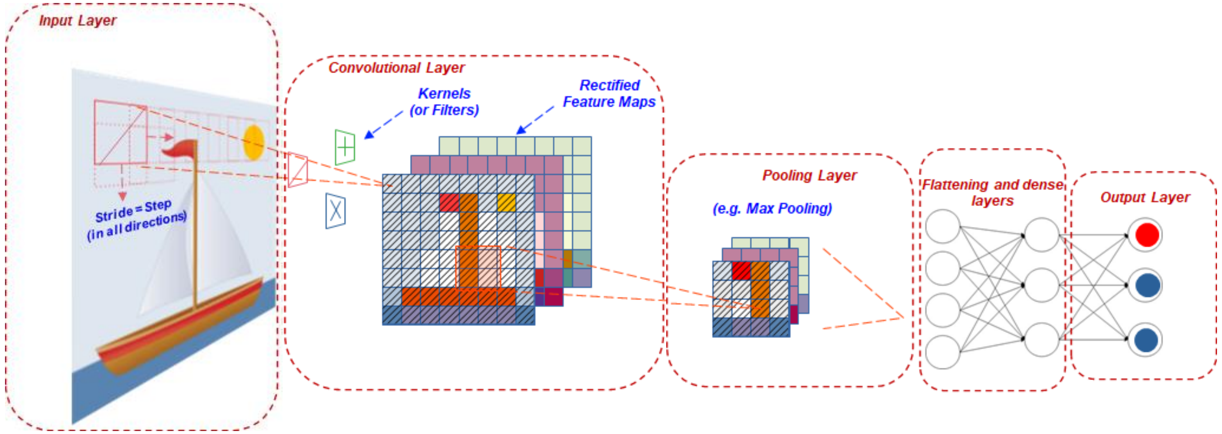
Why?

```python
# convert class vectors to binary class matrices
y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)

print('Train shape: x=', x_train.shape, ', y=', y_train.shape)
print('Test shape: x=', x_test.shape, ', y=', y_test.shape)
```

```
Train shape: x= (50000, 32, 32, 3) , y= (50000, 10)
Test shape: x= (10000, 32, 32, 3) , y= (10000, 10)
```

Why 3? Why 10?

CNN in Python (cont.)




```python
def model_2():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3),
                     strides=(1, 1),
                     activation='relu',
                     input_shape=(img_rows, img_cols, channels)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    model.summary()
    return model
```

Number of filter

1 Convolution and 1 Pooling Layers

Reduced to a quarter

Prevent overfitting, randomly dropping some connections with preceding layer

1 Flattening and 1 Dense Layers

1 Output Layer

Deep Learning CNN model is created

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 30, 30, 32)	896 = (3 x 3 x 3 + 1) * 32
max_pooling2d_1 (MaxPooling2D)	(None, 15, 15, 32)	0
dropout_2 (Dropout)	(None, 15, 15, 32)	0
flatten_1 (Flatten)	(None, 7200) = 15 x 15 x 32	0
dense_2 (Dense)	(None, 128)	921728 = (7200 + 1) * 128
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 10)	1290 = (128 + 1) * 10

Total params: 923,914
Trainable params: 923,914
Non-trainable params: 0

CNN in Python (cont.)

- We will experiment with various *optimizers* or algorithms searching for the best set of network *weights* and *biases*
- The error function is called *loss* to guide the optimiser
- Other *metrics* can also be used measure the net performance, e.g. accuracy

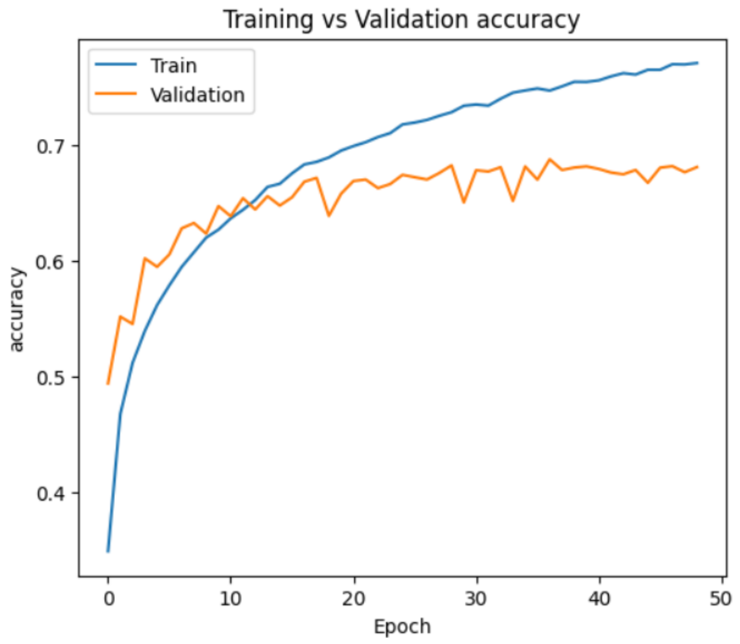
```
model.compile(loss=categorical_crossentropy,
              optimizer=RMSprop(learning_rate=0.001,weight_decay=1e-6),
              metrics='accuracy')

hist = model.fit(x_train, y_train,
        batch_size=128,
        epochs=100,
        verbose=2,
        validation_data=(x_test, y_test),
        validation_split=0.2,
        callbacks=keras_callbacks)
```

optimizers

Performance metric

Train the model and evaluate against test set.

```
Epoch 1/100
391/391 - 3s - loss: 2.1041 - accuracy: 0.2456 - val_loss: 1.8539 - val_accuracy: 0.3309
Epoch 2/100
391/391 - 2s - loss: 1.8945 - accuracy: 0.3132 - val_loss: 1.7669 - val_accuracy: 0.3608
Epoch 3/100
```



The training process terminated after 49 epochs.

```
Adadelta(lr=0.001, rho=0.95, epsilon=1e-07)
Adadelta(lr=0.05, rho=0.99, epsilon=1e-07)
Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-07)
```

Other Optimizers

Final Performance measures:

```
Train loss: 0.2722
Train accuracy: 0.9236

Test loss: 1.098
Test accuracy: 0.6806
```

CNN in Python (cont.)

Evaluation Metrics

```
The result of Kappa is : 0.645
The result of the classification report is:
              precision    recall  f1-score   support

    airplane       0.71      0.72      0.72      1000
  automobile       0.71      0.86      0.78      1000
        bird       0.62      0.50      0.55      1000
         cat       0.54      0.49      0.51      1000
        deer       0.70      0.55      0.61      1000
         dog       0.54      0.65      0.59      1000
        frog       0.70      0.79      0.74      1000
       horse       0.74      0.75      0.75      1000
        ship       0.81      0.77      0.79      1000
       truck       0.75      0.72      0.73      1000

    accuracy                           0.68     10000
   macro avg       0.68      0.68      0.68     10000
weighted avg       0.68      0.68      0.68     10000
```

Confusion Matrix

True label \ Predicted label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	724	34	44	21	17	12	13	9	74	52
automobile	16	862	1	11	1	5	4	2	20	78
bird	84	16	500	59	65	116	82	51	15	12
cat	20	28	56	489	44	215	80	39	11	18
deer	28	6	71	77	551	55	100	94	12	6
dog	19	8	45	143	32	650	37	44	13	9
frog	7	19	42	66	24	35	792	6	4	5
horse	22	10	28	30	53	76	9	752	2	18
ship	68	73	8	8	2	15	6	3	770	47
truck	35	159	7	10	3	16	6	18	30	716

Discussion

```python
def model_1():
    model = Sequential()
    model.add(Flatten(input_shape=(img_rows, img_cols, channels)))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(10, activation='softmax'))
    model.summary()
    return model
```

Is this a CNN model?

```
The result of Kappa is : 0.387
The result of the classification report is:
              precision    recall  f1-score   support

    airplane       0.60      0.36      0.46      1000
  automobile       0.55      0.58      0.56      1000
        bird       0.37      0.29      0.33      1000
         cat       0.30      0.28      0.29      1000
        deer       0.47      0.29      0.36      1000
         dog       0.42      0.30      0.35      1000
        frog       0.40      0.66      0.50      1000
       horse       0.53      0.48      0.51      1000
        ship       0.59      0.56      0.58      1000
       truck       0.38      0.66      0.48      1000

    accuracy                           0.45     10000
   macro avg       0.46      0.45      0.44     10000
weighted avg       0.46      0.45      0.44     10000
```

```python
def model_2():
    model = Sequential()
    model.add(Conv2D(32, kernel_size=(3, 3),
                     activation='relu',
                     input_shape=(img_rows, img_cols, channels)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    model.summary()
    return model
```

Which model would perform better? And Why?

```
The result of Kappa is : 0.645
The result of the classification report is:
              precision    recall  f1-score   support

    airplane       0.71      0.72      0.72      1000
  automobile       0.71      0.86      0.78      1000
        bird       0.62      0.50      0.55      1000
         cat       0.54      0.49      0.51      1000
        deer       0.70      0.55      0.61      1000
         dog       0.54      0.65      0.59      1000
        frog       0.70      0.79      0.74      1000
       horse       0.74      0.75      0.75      1000
        ship       0.81      0.77      0.79      1000
       truck       0.75      0.72      0.73      1000

    accuracy                           0.68     10000
   macro avg       0.68      0.68      0.68     10000
weighted avg       0.68      0.68      0.68     10000
```

How to improve prediction performance?

Working with Real Digital Photos

```
drive.mount('/content/drive')

# Set the paths to the folders containing the image files
city_path = '/content/drive/MyDrive/Colab Notebooks/dataset/city'
country_path = '/content/drive/MyDrive/Colab Notebooks/dataset/country'
```


```
# Iterate through the files in the first folder
for file in os.listdir(city_path):
  # Check if the file is a jpeg or jpg file
  if file.endswith('.jpeg') or file.endswith('.jpg'):
    # Load the image data from the file using TensorFlow
    img = tf.io.read_file(os.path.join(city_path, file))
    img = tf.image.decode_jpeg(img)
    img = tf.image.resize(img, (100, 100))
    # Assign a label to the file
    label = 'City'
    # Add the image data and label to the data list
    data.append((img, label))
```

In this lecture, we have covered:

- The concepts and architectures of CNN.
- Experiments with various model architectures of CNN in Python.
- Discussion on the applications of these deep learning techniques.

Summary