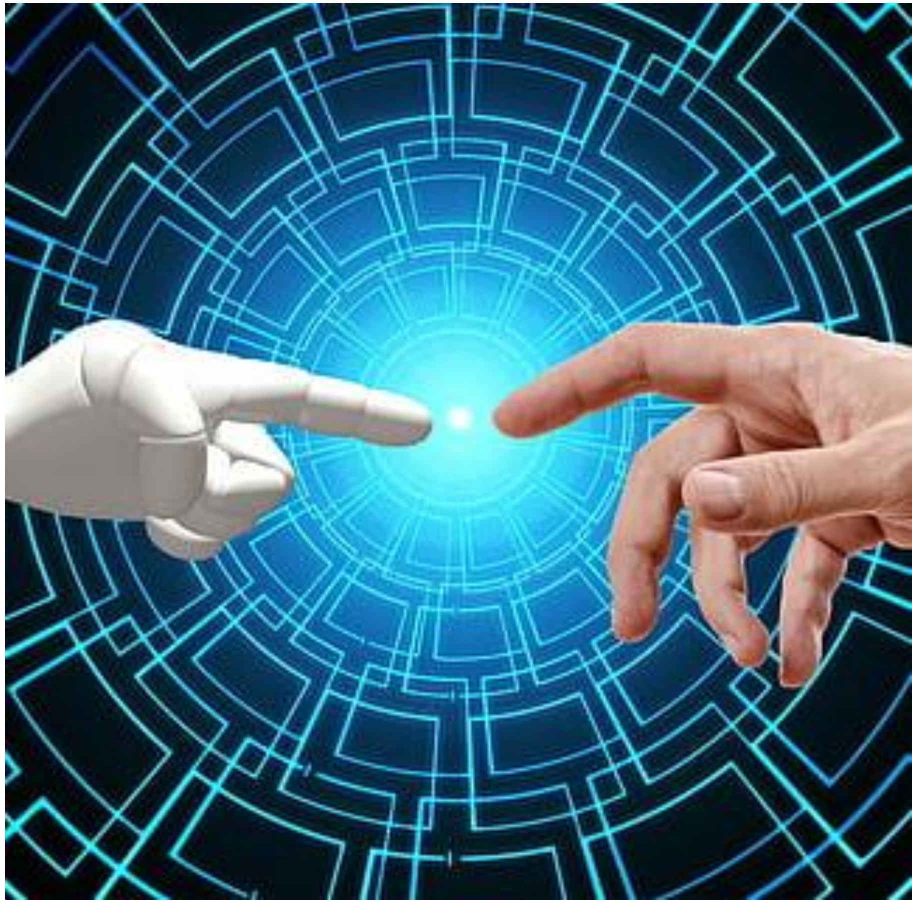
Artificial Intelligence (AI) for Engineering

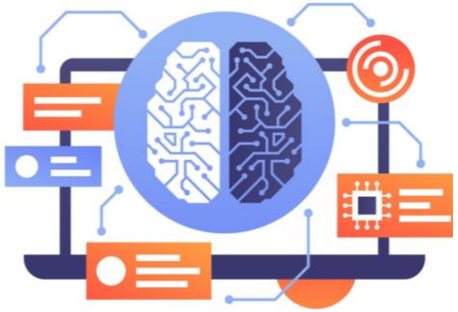
COS40007

Dr. Abdur Forkan

Senior Research Fellow, AI and Machine Learning

Digital Innovation Lab



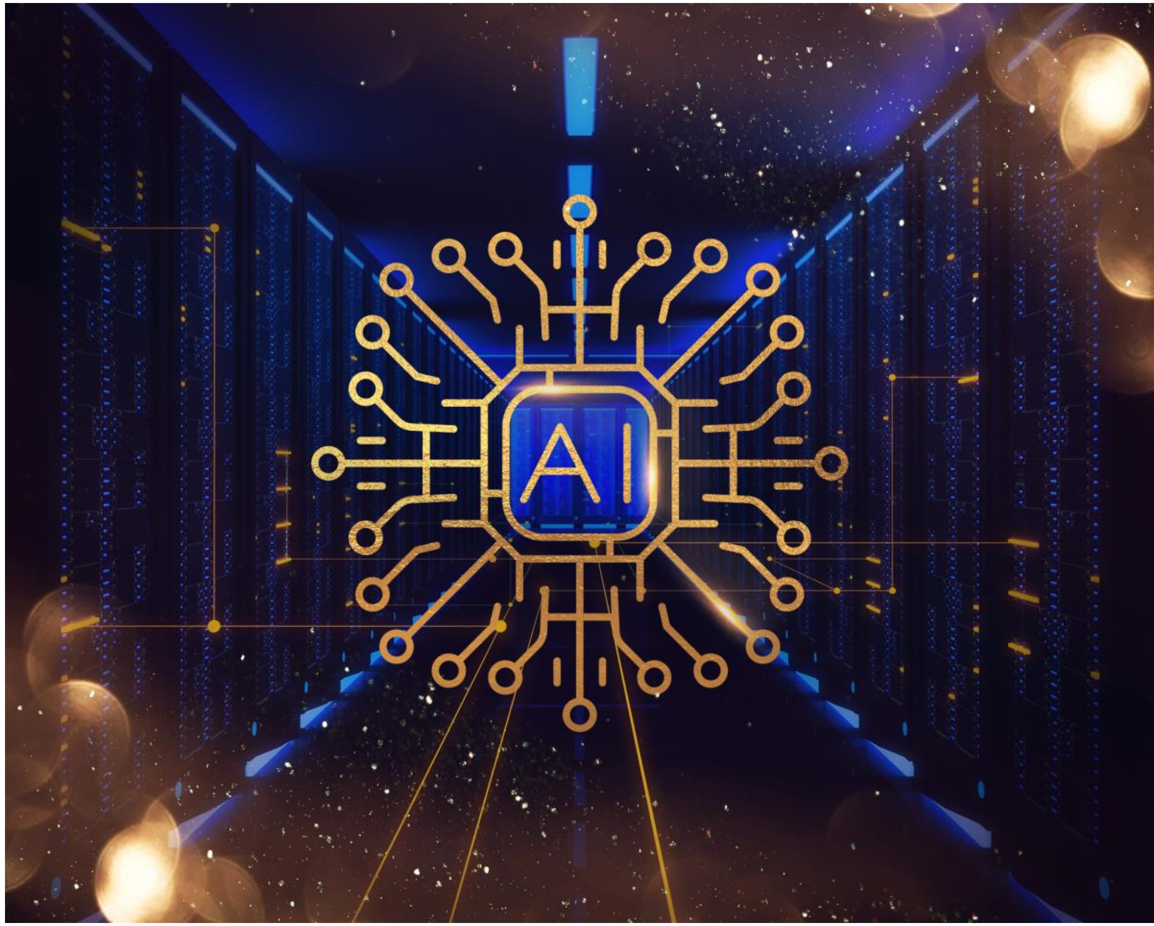


Seminar 5: 28th August 2024

SWINBURNE UNIVERSITY OF TECHNOLOGY

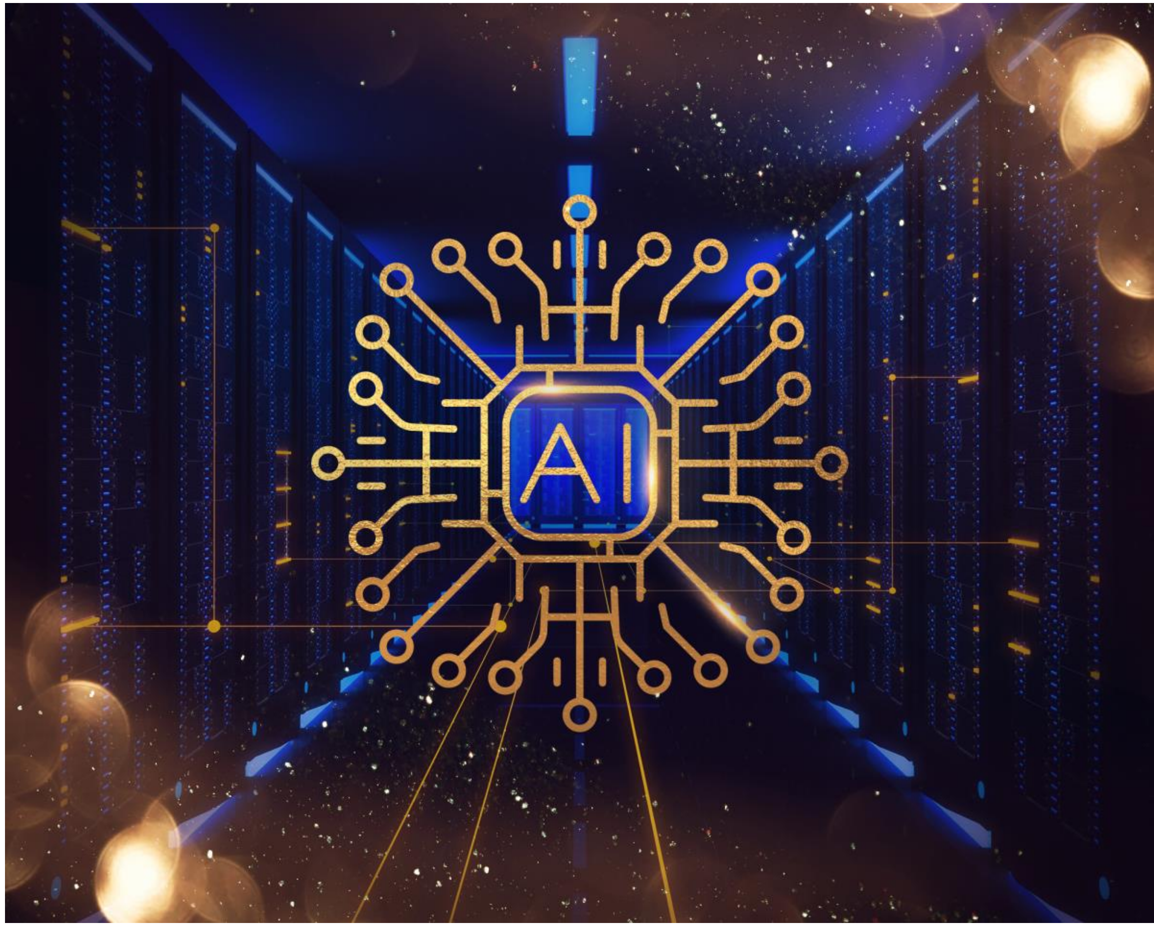
Overview

- ❑Deep Learning
- ❑Basics of CNN
- ❑Transfer Learning
- ❑R CNN
- ❑Examples of deep learning with keras and tensorflow



Required Reading

- Chapter 8, 10, 12 of "Applied Machine Learning and AI for Engineers"
- Chapter 14 of "Machine Learning with Pytorch and Scikit-Learn"



At the end of this you should be able to

- Understand what is deep learning
- Understand what is CNN and relevant functionalities
- Understand how to create CNN, train it and use it for image classification
- Understand how to use transfer learning using RestNet
- Understand how to do object recognition using Mask RCNN

Deep Learning

Deep learning use cases

- computer vision
- speech recognition
- image processing
- bioinformatics
- social network filtering
- drug design
- Recommendation systems
- Bioinformatics
- Mobile Advertising
- Many others

Image classification

SWIN BUR NE
SWINBURNE UNIVERSITY OF TECHNOLOGY

CNN

It is a class of deep learning.

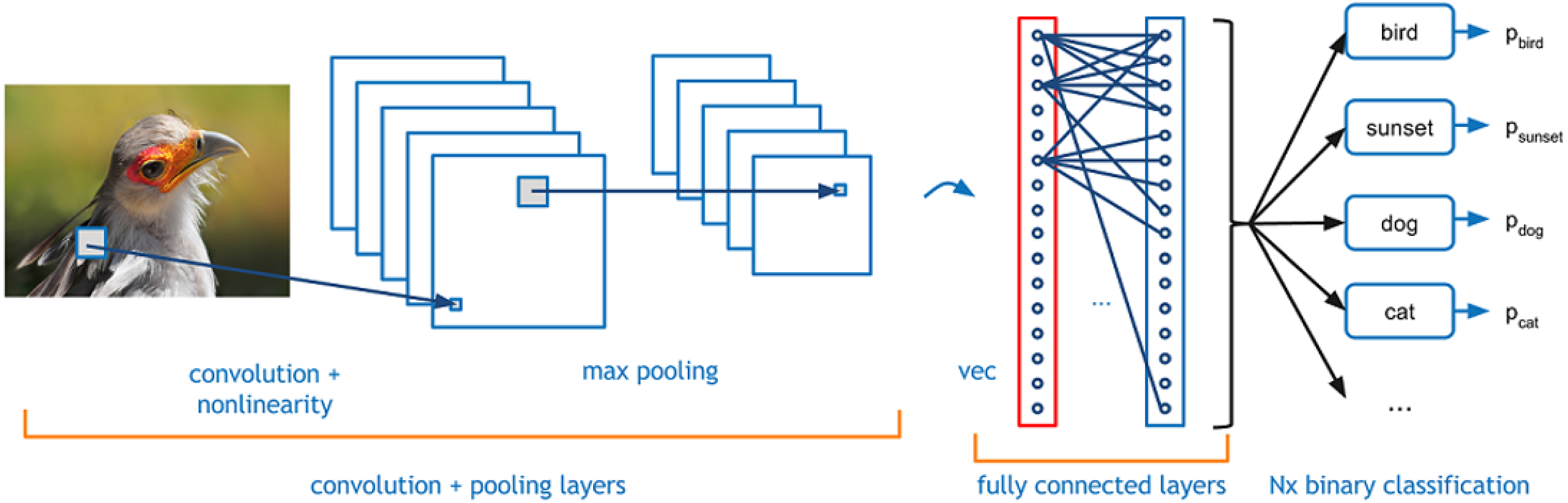
Convolutional neural network (ConvNet's or CNNs) is one of the main categories to do images recognition, images classifications, objects detections, recognition faces etc.,

It is similar to the basic neural network. CNN also have learnable parameter like neural network i.e., weights, biases etc.

There 3 basic components to define CNN

- The Convolution Layer
- The Pooling Layer
- The Output Layer (or) Fully Connected Layer

CNN Architecture



Number of Layers

- Deeper networks is always better, at the cost of more data and increased complexity of learning.
- You should initially use fewer filters and gradually increase and monitor the error rate to see how it is varying.
- Very small filter sizes will capture very fine details of the image. On the other hand having a bigger filter size will leave out minute details in the image.

Types of CNN

- The five major areas which can be addressed using CNN.
    - Image Classification
    - Object Detection
    - Object Tracking
    - Semantic Segmentation
    - Instance Segmentation

CNN Types: Image classification

- In an image classification we can use the traditional CNN models or there also many architectures designed by developers to decrease the error rate and increasing the trainable parameters.
    - LeNet (1998)
    - AlexNet (2012)
    - ZFNet (2013)
    - GoogLeNet19 (2014)
    - VGGNet 16 (2014)
    - ResNet(2015)

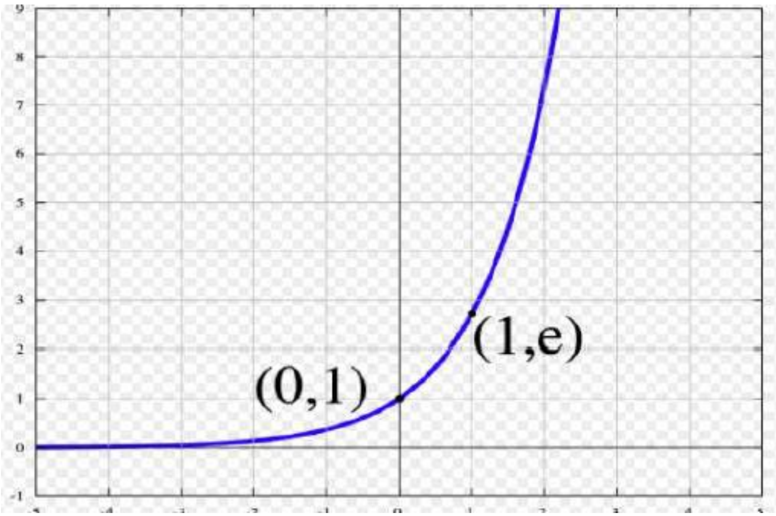
CNN Types: Object Detections

Object Detection:

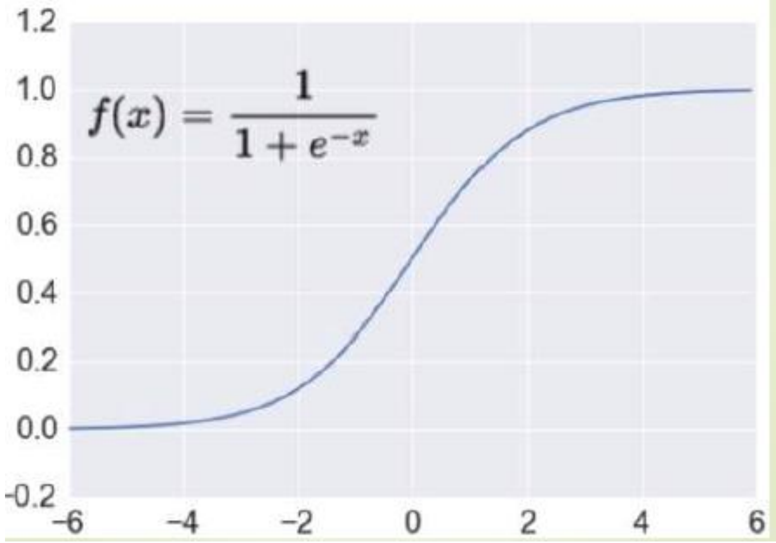
- Here the implementation of CNN is different compared to the previous image classification.
- Here the task is to identify the objects present in the image, therefore traditional implementation of CNN may not help.
    - R CNN
    - Fast R CNN
    - Faster R CNN
    - Mask R CNN
    - YOLO

Activation Functions

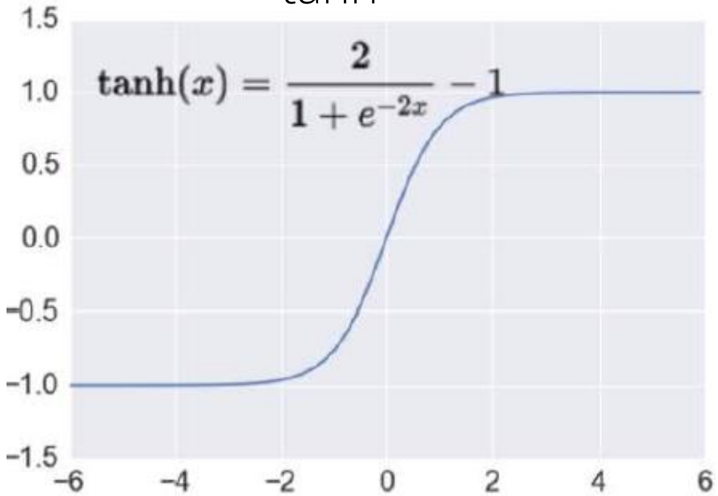
Euler



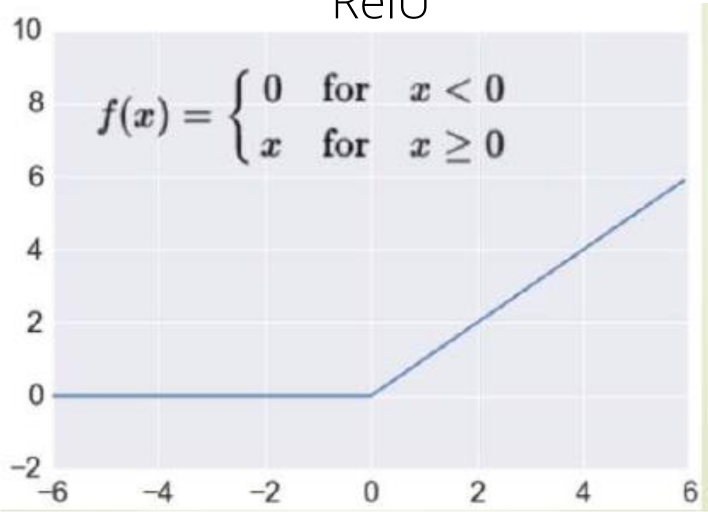
Sigmoid



tanh



RelU



$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

SoftMax

Activation function in Python

```
import numpy as np

Python sigmoid example:
z = 1/(1 + np.exp(-np.dot(W, x)))

Python tanh example:
z = np.tanh(np.dot(W,x));

Python ReLU example:
z = np.maximum(0, np.dot(W, x))
```

“Best” Activation Function

Initially: sigmoid was popular

Then: tanh became popular

Now: RELU is preferred (better results)

Softmax: for FC (fully connected) layers

sigmoid and tanh are used in LSTMs

Optimisers

- SGD
- rmsprop
- Adagrad
- Adam
- Others

Deep Learning Summary

- input layer, multiple hidden layers, and output layer
- nonlinear processing via activation functions
- perform transformation and feature extraction
- gradient descent algorithm with back propagation
- each layer receives the output from previous layer
- results are comparable/superior to human experts

Keras and Tensorflow

CNN in Keras

```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.optimizers import Adadelta
input_shape = (3, 32, 32)
nb_classes = 10
model = Sequential()
model.add(Conv2D(32,(3, 3),padding='same',
input_shape=input_shape))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

SWINBURNE UNIVERSITY OF TECHNOLOGY

Tensorflow

- An open source framework for ML and DL
- Created by Google (released 11/2015)
- Evolved from Google Brain
- Visualization via TensorBoard

- TF tensors are n-dimensional arrays
- TF tensors are very similar to numpy ndarrays

CNN: Training in Keras

```
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense, GlobalAveragePooling2D
from keras import backend as K
from keras.callbacks import CSVLogger


# dimensions of our images.
img_width, img_height = 330, 247

train_data_dir = '../images/data/train'
validation_data_dir = '../images/data/validation'
epochs = 100
batch_size = 64

if K.image_data_format() == 'channels_first':
   input_shape = (3, img_width, img_height)
else:
   input_shape = (img_width, img_height, 3)
```

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))

model.compile(loss='binary_crossentropy',
          optimizer='rmsprop',
          metrics=['accuracy'])
```

CNN: Training in Keras

```
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

validation_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
   train_data_dir,
   target_size=(img_width, img_height),
   batch_size=batch_size,
   shuffle=True,
   class_mode='binary')

validation_generator = validation_datagen.flow_from_directory(
   validation_data_dir,
   target_size=(img_width, img_height),
   batch_size=batch_size,
   shuffle=True,
   class_mode='binary')
```

```
history = model.fit_generator(
   train_generator,
   epochs=epochs,
   steps_per_epoch = len(train_generator.filenames) // batch_size,
   validation_steps = len(validation_generator.filenames) // batch_size,
   validation_data=validation_generator,
   callbacks=[ CSVLogger("training.log",
                      append=False,
                     separator=";")]
)

#saving model
model_json = model.to_json()
with open("model.json", "w") as json_file:
   json_file.write(model_json)
model.save_weights('model_weights.h5')
model.save("model.h5")

training_accuracy = history.history['acc']
validation_accuracy = history.history['val_acc']
```

Transfer Learning: Training

```
import keras
from keras.layers import Dense,GlobalAveragePooling2D
from keras.applications import ResNet50
from keras.utils import multi_gpu_model
from keras.applications.resnet50 import preprocess_input
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Model
from keras import backend as K
```

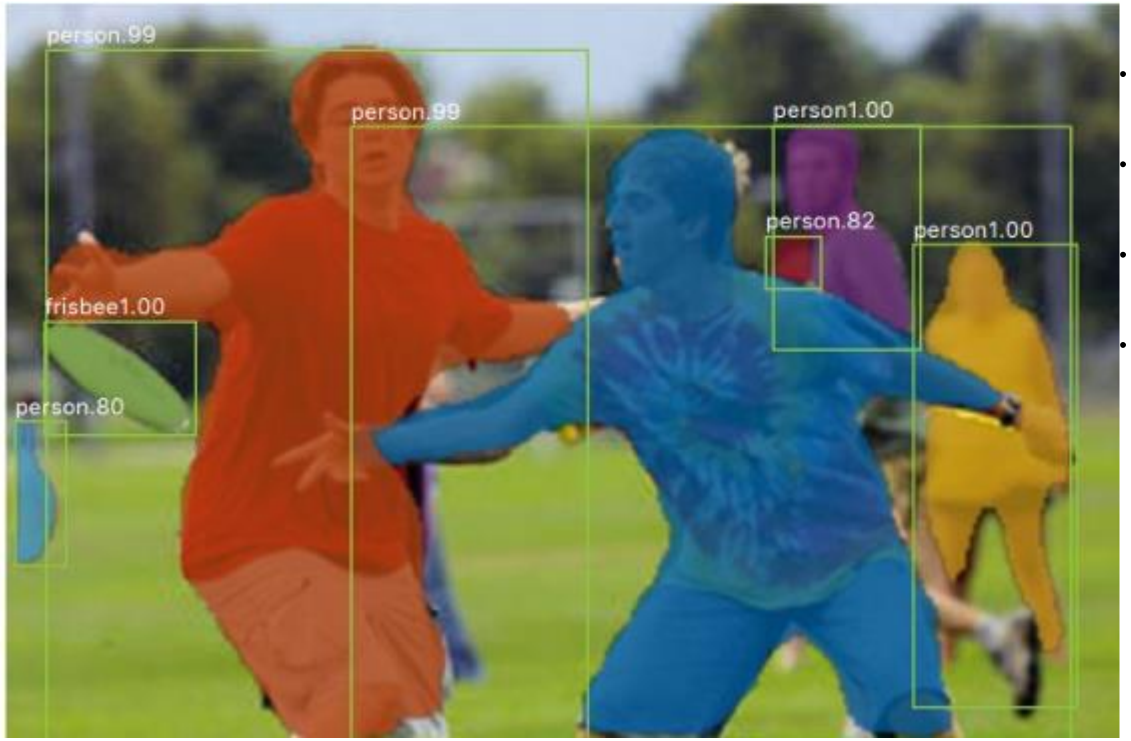
```
base_model = ResNet50(weights=None, include_top=False)
base_model.load_weights('resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5')
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(256, activation='relu')(x)
x = Dense(128, activation='relu')(x)
x = Dense(64, activation='relu')(x)
preds = Dense(1, activation='sigmoid')(x)
model = Model(inputs=base_model.input, outputs=preds)

for layer in base_model.layers:
    layer.trainable = False

parallel_model = multi_gpu_model(model, gpus=2)

parallel_model.compile(loss='binary_crossentropy',
                       optimizer='rmsprop',
                       metrics=['accuracy'])
```

Classification

```
test_generator = test_data_generator.flow_from_directory(
        dir_name,
        target_size=(img_width, img_height),
        batch_size=1,
        class_mode=None,
        shuffle=False)
probabilities = model.predict_generator(test_generator, (ImgWidth * IngHeight))
```

Mask RCNN

- Mask R-CNN is framework to solve the instance segmentation problem.
- Instance segmentation is a task of detecting and delineating each object in an image in a fine-grained pixel level
- Instance segmentation can estimate object position given an image, so tasks such as robot manipulation can perform grasp planning
- It is an extension of the Faster R-CNN framework.



Learn, Practice and Enjoy the AI journey