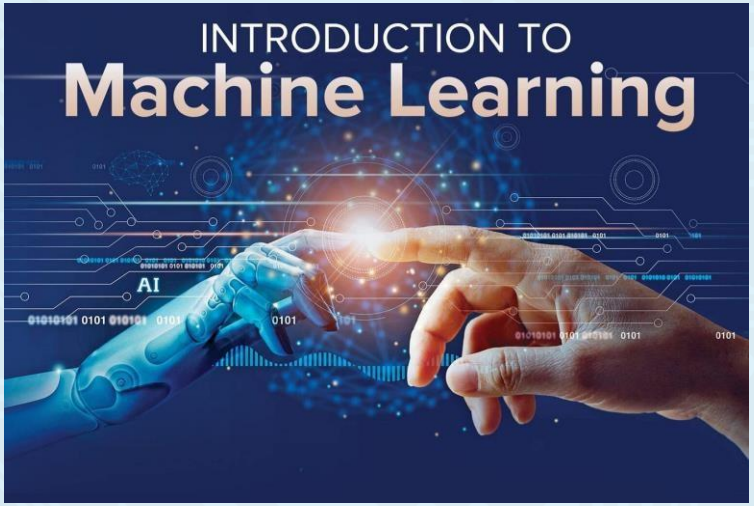
Introduction to machine learning (Workshop 1)



By/ Aly Maher Abdel Fattah

#SAVE PALESTINE

IEEE Galala University Student Branch

Lesson Outline

- *Medical diagnosis*
- *Image Classification*
- *Neural Network*
- *Convolutional Neural Network*
- *Dataset*
- *Libraries used*
- *Code*

Medical Diagnosis

- ***Medical diagnosis is the process of determining which disease or condition explains a person's symptoms and signs. It is most often referred to as diagnosis with the medical context being implicit.***
- ***The information required for diagnosis is typically collected from a history and physical examination of the person seeking medical care.***
- ***A medical diagnosis is a complex medical step involving patient history, personal exams, and testing. In many cases, there are multiple diagnoses completed by varying doctors. This process allows for a more comprehensive understanding of patient ailments.***

Image classification

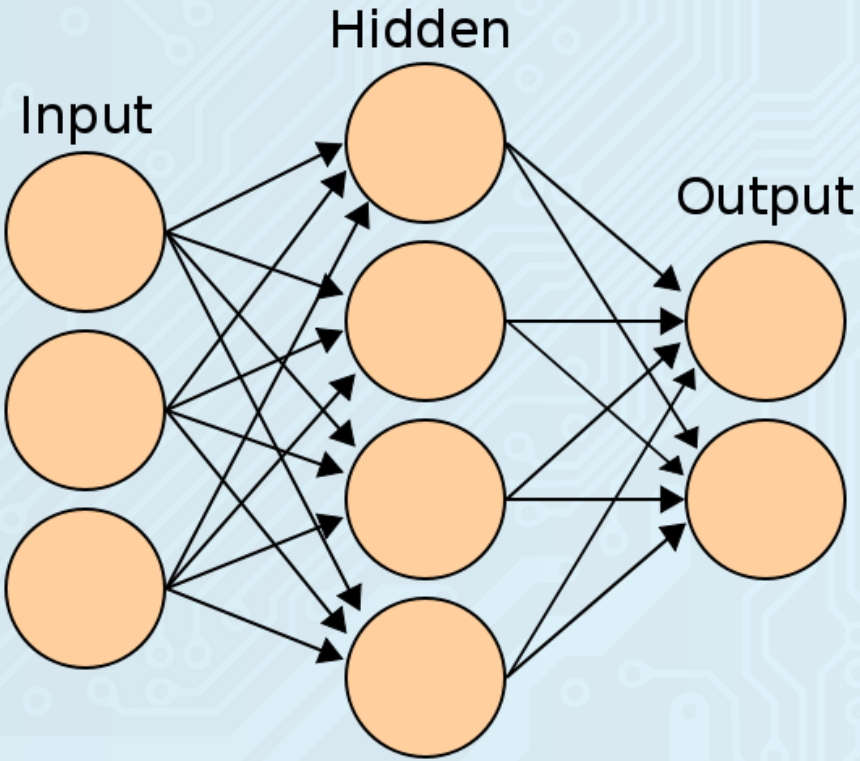
- ***The convolutional neural network (CNN) is a class of deep learning neural networks. CNNs represent a huge breakthrough in image recognition. They're most commonly used to analyze visual imagery and are frequently working behind the scenes in image classification. They can be found at the core of everything from Facebook's photo tagging to self-driving cars. They're working hard behind the scenes in everything from healthcare to security.***

- ***Image classification is the process of taking an input (like a picture) and outputting a class (like "cat") or a probability that the input is a particular class ("there's a 90% probability that this input is a cat").***

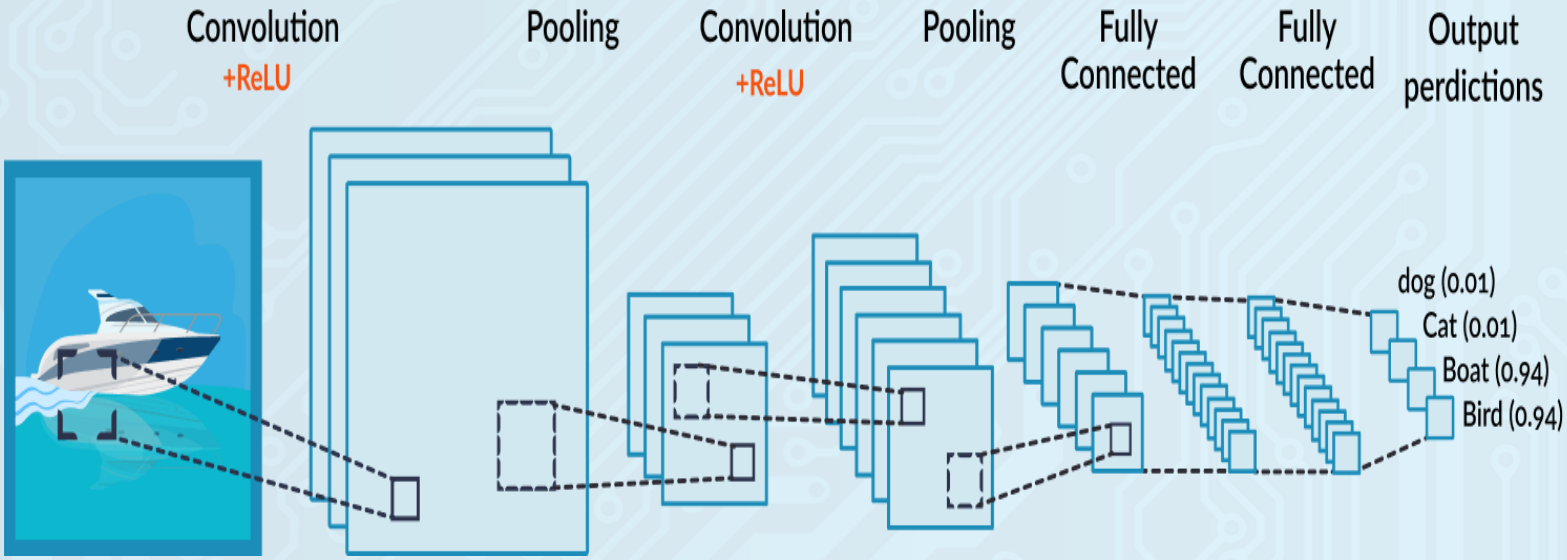
Image classification

- ***CNNs have an input layer, and output layer, and hidden layers. The hidden layers usually consist of convolutional layers, ReLU layers, pooling layers, and fully connected layers.***
- ***Convolutional layers apply a convolution operation to the input. This passes the information on to the next layer.***
- ***Pooling combines the outputs of clusters of neurons into a single neuron in the next layer.***
- ***Fully connected layers connect every neuron in one layer to every neuron in the next layer.***

Neural network



Convolutional neural network



Convolutional neural network

- ***A CNN***
  - ***starts with an input image***
  - ***applies many different filters to it to create a feature map***
  - ***applies a ReLU function to increase non-linearity***
  - ***applies a pooling layer to each feature map***
  - ***flattens the pooled images into one long vector.***
  - ***inputs the vector into a fully connected artificial neural network.***
  - ***processes the features through the network. The final fully connected layer provides the "voting" of the classes that we're after.***
  - ***trains through forward propagation and backpropagation for many, many epochs. This repeats until we have a well-defined neural network with trained weights and feature detectors.***

Dataset

DataSet:
https://github.com/Maher1410/Breast-Cancer-Classification/blob/main/Breast_Cancer_Event_Dataset.zip

This tutorial is better performed on a computer containing an Nvidia 1080ti GPU, dual-xeon E5-2670 Intel CPUs, and 64 gb RAM. However, you could perform this experiment on a typical laptop using the CPU only.

Libraries used

```
from keras import applications
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense, GlobalAveragePooling2D
from keras.models import Model
from keras.optimizers import Adam
```

Libraries used

- ***Keras**: Is A Neural Networks Library Written In Python That Is High-level In Nature – Which Makes It Extremely Simple And Intuitive To Use. It Works As A Wrapper To Low-level Libraries Like Tensorflow Or Theano High-level Neural Networks Library, Written In Python That Works As A Wrapper To Tensorflow Or Theano.*

- ***Imagedatagenerator**: Accepts An Input Batch Of Images, Randomly Transforms The Batch, And Then Returns Both The Original Batch And Modified Data.*

- ***Optimizers**: Update The Weight Parameters To Minimize The Loss Function. Loss Function Acts As Guides To The Terrain Telling Optimizer If It Is Moving In The Right Direction To Reach The Bottom Of The Valley, The Global Minimum.*

Libraries used

- ***Sequential**: The simplest model is defined in the Sequential class which is a linear stack of Layers.*
- ***Dropout**: is a technique where randomly selected neurons are ignored during training. Flatten: is used to flatten the input.*
- ***Dense**: is the only actual network layer in that model.A Dense layer feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer. It's the most basic layer in neural networks.*

Libraries used

- ***global average pooling:*** ***Most deep networks today use a single layer of global average pooling at or near the top of the network, rather than rely on multiple layers of fully connected networks for the highest levels of processing.***

- ***Model:*** ***As with the Sequential API, the model is the thing you can summarize, fit, evaluate, and use to make predictions. Keras provides a Model class that you can use to create a model from your created layers.***

- ***Adam:*** ***is a replacement optimization algorithm for stochastic gradient descent for training deep learning models. Adam combines the best properties of the AdaGrad and RMSProp algorithms to provide an optimization algorithm that can handle sparse gradients on noisy problems.***

Code

```
# dimensions of our images.
img_width, img_height = 299, 299

train_data_dir = 'C:\\Users\\DELL 5540\\OneDrive\Desktop\\Breast-Cancer\\Breast_Cancer_Event_Dataset\\TRAIN' #location of training data
validation_data_dir = 'C:\\Users\\DELL 5540\\OneDrive\\Desktop\\Breast-Cancer\\Breast_Cancer_Event_Dataset\\VAL' #location of validation data

# number of samples used for determining the samples_per_epoch
nb_train_samples = 65
nb_validation_samples = 10
epochs = 30
batch_size = 5
```

Model

- ***Build the pre-trained Inception V3 network, a popular CNN that achieved a top 5 accuracy of greater than 94% on the ILSVRC.***

```
base_model = applications.InceptionV3(weights='imagenet', include_top=False, input_shape=(img_width, img_height, 3))
```

Code

```
model_top = Sequential()
model_top.add(GlobalAveragePooling2D(input_shape=base_model.output_shape[1:], data_format=None)),
model_top.add(Dense(256, activation='relu'))
model_top.add(Dropout(0.5))
model_top.add(Dense(1, activation='sigmoid'))

model = Model(inputs=base_model.input, outputs=model_top(base_model.output))

model.compile(optimizer=Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-08),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Code

```
model_top = Sequential()
model_top.add(GlobalAveragePooling2D(input_shape=base_model.output_shape[1:], data_format=None)),
model_top.add(Dense(256, activation='relu'))
model_top.add(Dropout(0.5))
model_top.add(Dense(1, activation='sigmoid'))

model = Model(inputs=base_model.input, outputs=model_top(base_model.output))

model.compile(optimizer=Adam(learning_rate=0.0001, beta_1=0.9, beta_2=0.999, epsilon=1e-08),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Training data

```
history = model.fit(
            train_generator,
            steps_per_epoch=nb_train_samples // batch_size,
            epochs=epochs,
            validation_data=validation_generator,
            validation_steps=nb_validation_samples // batch_size)
```

Output of training

```
Epoch 1/30
13/13 [==============================] - 107s 4s/step - loss: 0.7401 - accuracy: 0.4769 - val_loss: 0.8065 - val_accuracy: 0.5000
Epoch 2/30
13/13 [==============================] - 40s 3s/step - loss: 0.6490 - accuracy: 0.5846 - val_loss: 0.7887 - val_accuracy: 0.3000
Epoch 3/30
13/13 [==============================] - 43s 3s/step - loss: 0.6631 - accuracy: 0.5692 - val_loss: 0.7803 - val_accuracy: 0.3000
Epoch 4/30
13/13 [==============================] - 53s 4s/step - loss: 0.7149 - accuracy: 0.5846 - val_loss: 0.7679 - val_accuracy: 0.6000
Epoch 5/30
13/13 [==============================] - 55s 4s/step - loss: 0.6266 - accuracy: 0.6000 - val_loss: 0.9217 - val_accuracy: 0.5000
Epoch 6/30
13/13 [==============================] - 52s 4s/step - loss: 0.6262 - accuracy: 0.6923 - val_loss: 0.7386 - val_accuracy: 0.7000
Epoch 7/30
13/13 [==============================] - 41s 3s/step - loss: 0.6038 - accuracy: 0.7231 - val_loss: 0.6653 - val_accuracy: 0.7000
Epoch 8/30
13/13 [==============================] - 41s 3s/step - loss: 0.4862 - accuracy: 0.7692 - val_loss: 0.7655 - val_accuracy: 0.4000
Epoch 9/30
13/13 [==============================] - 41s 3s/step - loss: 0.4936 - accuracy: 0.7385 - val_loss: 0.8384 - val_accuracy: 0.6000
Epoch 10/30
13/13 [==============================] - 37s 3s/step - loss: 0.5376 - accuracy: 0.7846 - val_loss: 0.8621 - val_accuracy: 0.7000
Epoch 11/30
13/13 [==============================] - 23s 2s/step - loss: 0.4669 - accuracy: 0.7692 - val_loss: 0.8442 - val_accuracy: 0.3000
Epoch 12/30
13/13 [==============================] - 23s 2s/step - loss: 0.3908 - accuracy: 0.8308 - val_loss: 1.2142 - val_accuracy: 0.3000
Epoch 13/30
...
Epoch 29/30
13/13 [==============================] - 31s 2s/step - loss: 0.1918 - accuracy: 0.9385 - val_loss: 1.0803 - val_accuracy: 0.5000
Epoch 30/30
13/13 [==============================] - 30s 2s/step - loss: 0.2458 - accuracy: 0.9538 - val_loss: 1.0443 - val_accuracy: 0.6000
```

Plot data

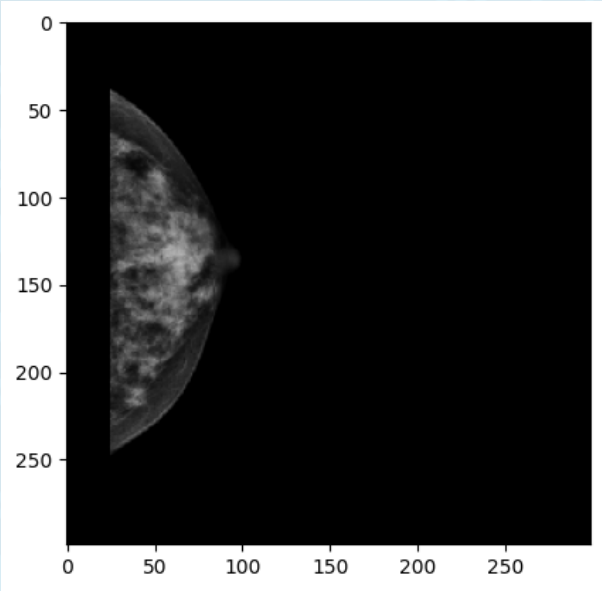
```
import matplotlib.pyplot as plt

print(history.history.keys())

plt.figure()
plt.plot(history.history['accuracy'], 'orange', label='Training accuracy')
plt.plot(history.history['val_accuracy'], 'blue', label='Validation accuracy')
plt.plot(history.history['loss'], 'red', label='Training loss')
plt.plot(history.history['val_loss'], 'green', label='Validation loss')
plt.legend()
plt.show()
```

Output of plotting

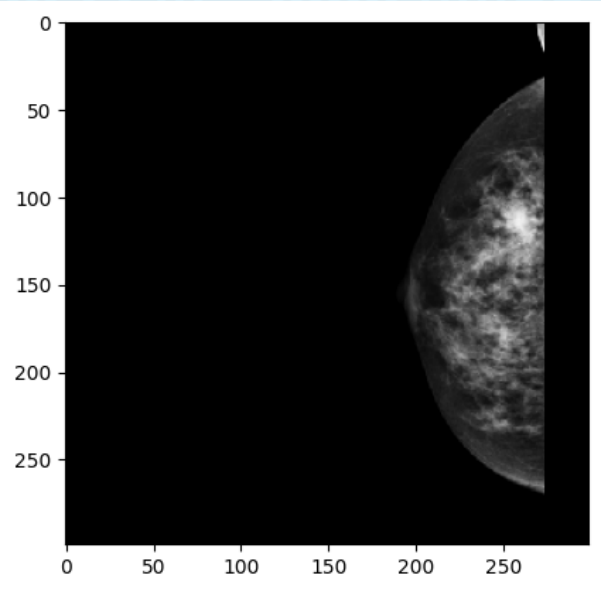
Testing

```
import numpy as np
from keras.preprocessing import image

img_path='C:\\Users\\DELL 5540\\OneDrive\\Desktop\\Breast-Cancer\\Breast_Cancer_Event_Dataset\\TEST\\benign.jpg'
img_path2='C:\\Users\\DELL 5540\\OneDrive\\Desktop\\Breast-Cancer\\Breast_Cancer_Event_Dataset\TEST\\malignant.jpg'
img = image.load_img(img_path, target_size=(img_width, img_height))
img2 = image.load_img(img_path2, target_size=(img_width, img_height))
plt.imshow(img)
plt.show()

img = image.img_to_array(img)
x = np.expand_dims(img, axis=0) * 1./255
score = model.predict(x)
print('Predicted:', score, 'benign' if score < 0.5 else 'malignant')

plt.imshow(img2)
plt.show()

img2 = image.img_to_array(img2)
x = np.expand_dims(img2, axis=0) * 1./255
score2 = model.predict(x)
print('Predicted:', score2, 'benign' if score2 < 0.5 else 'malignant')
```

Output of testing




```
1/1 [==============================] - 0s
97ms/step Predicted: [[0.013973]] benign
```




```
1/1 [==============================] - 0s
124ms/step Predicted: [[0.9800183]] malignant
```

Thank you

Any Question?