# Artificial intelligence

## Phase:3

## Tensorflow and keras-ANN

TensorFlow and Keras are powerful tools for building Artificial Neural Networks (ANNs) in Python. Keras is actually an API that is now integrated into TensorFlow (as of TensorFlow 2.0), making it easier to build and train neural networks.

You can create an ANN using Keras and TensorFlow by following these general steps:

Import Libraries: Import TensorFlow and Keras libraries.

Data Preprocessing: Prepare your data, including scaling and splitting it into training and testing sets.

Build the Model: Create a Sequential model in Keras. You can add layers using Dense for fully connected layers and specify activation functions.

Compile the Model: Define the loss function, optimizer, and metrics for your model using the compile method.

Train the Model: Use the fit method to train your model on the training data.

Evaluate and Fine-Tune: Evaluate the model's performance on the test data, and fine-tune hyperparameters as needed.

Make Predictions: Once your model is trained, you can use it to make predictions on new data.

Here's a simple example of creating a feedforward neural network using Keras and TensorFlow:

pythonCopy code

```
import tensorflow as tf from tensorflow import keras # Load your data and preprocess it model = keras.Sequential([ keras.layers.Dense(64, activation='relu', input_shape=(input_dim,)), keras.layers.Dense(32, activation='relu'), keras.layers.Dense(1, activation='sigmoid') ]) model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy']) model.fit(X_train, y_train, epochs=10, batch_size=32) test_loss, test_accuracy = model.evaluate(X_test, y_test) print("Test accuracy:", test_accuracy)
```

This is a basic overview of building ANNs with TensorFlow and Keras. ANNs can be much more complex with different layer types, architectures, and techniques depending on your specific task.

## Convolunational Neural network

Convolutional Neural Networks (CNNs) are a specialized type of artificial neural network primarily used for tasks involving image and spatial data. CNNs are highly effective in computer vision, image recognition, and various other applications where the spatial relationships in the data are crucial. Here's an overview of how CNNs work:

Convolution Layers: CNNs start with one or more convolutional layers. Convolution is a mathematical operation that scans a small filter (also called a kernel) over the input data. The filter is designed to detect specific patterns or features in the data. Multiple filters are used to capture different features.

Pooling Layers: After each convolution layer, pooling layers are often used. Pooling reduces the

spatial dimensions of the feature maps produced by the convolution layers, making the network more robust and efficient. Max pooling and average pooling are common pooling techniques.

Fully Connected Layers: After several convolution and pooling layers, there are one or more fully connected layers. These layers are like traditional neural network layers and are used to make final decisions based on the features learned by the convolutional layers.

Activation Functions: Activation functions like ReLU (Rectified Linear Unit) are applied after convolution and fully connected layers to introduce non-linearity into the model.

Flatten Layer: Before transitioning from convolutional layers to fully connected layers, a flatten layer is used to convert the 2D feature maps into a 1D vector.

Output Layer: The final layer usually has as many neurons as there are classes in a classification task. In regression tasks, it may have a single neuron.

Training: CNNs are trained using labeled data and optimization techniques like gradient descent. The network learns to adjust its parameters to minimize a loss function, which measures the error between predicted and actual values.

Testing and Predictions: Once trained, CNNs can make predictions on new, unseen data. The output of the network corresponds to the class probabilities in classification tasks or the predicted values in regression tasks.

CNNs are widely used in image classification, object detection, facial recognition, and many other computer vision applications. They have proven to be highly effective in capturing hierarchical and complex patterns in images, making them a cornerstone of modern AI and machine learning.

## Open cv

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a wide range of tools and functions for various computer vision

tasks, including image and video processing, object detection, feature extraction, image stitching, machine learning, and more. OpenCV is written in C++ and has Python bindings, making it accessible to developers in both languages. Here are some key features and use cases for OpenCV:

Image Processing: OpenCV offers a comprehensive set of functions for image processing tasks such as filtering, thresholding, resizing, and color space transformations.

Video Processing: You can use OpenCV to work with video streams, process frames in real-time, and perform tasks like video capture, recording, and analysis.

Object Detection and Tracking: OpenCV provides pre-trained models and tools for object detection, face recognition, and object tracking. Popular techniques like Haar cascades and deep learningbased methods are supported.

Feature Detection and Matching: OpenCV includes algorithms for detecting and matching features in images, making it useful for applications like

image stitching and feature-based object recognition.

Machine Learning: OpenCV can be integrated with machine learning libraries like scikit-learn and TensorFlow for training and using machine learning models for computer vision tasks.

Camera Calibration: It offers tools for camera calibration, which is essential for 3D reconstruction from multiple images and augmented reality applications.

Image Segmentation: OpenCV provides methods for segmenting images into regions based on characteristics such as color or texture.

Graphical User Interfaces: OpenCV includes graphical user interface (GUI) functions for displaying images and interacting with the user.

Cross-Platform: OpenCV is available on various platforms, including Windows, Linux, macOS, Android, and iOS.

Open Source: OpenCV is open source and has a large community of developers and users, making

it a widely adopted library for computer vision applications.

To use OpenCV in Python, you can install it using pip and then import it in your code. Here's a simple example of loading an image and displaying it using OpenCV in Python:

pythonCopy code

```
import cv2 # Load an image from file image = cv2.imread('image.jpg') # Display the image in a window cv2.imshow('My Image', image) # Wait for a key press and then close the window cv2.waitKey(0) cv2.destroyAllWindows()
```

OpenCV is a versatile library that can be used for a wide range of computer vision and image processing tasks, making it an essential tool for many computer vision applications.

THANK YOU

Submitted by: V.ADARSH

adarshvarma056@gmail.com

au723921243056