PHASE 3 PROJECT

DEVOLOPMENT 1:

TENSORFLOW AND KERAS:

Tensorflow is an open-sourced end-o-end platform,a library for multiple machine learning tasks, while keras is a high level neural network library that runs on top of tensor flow.

Every tensorflow user should use the keras APIs by default. whether you are an engineer, researcher, an MLA practitioner, you should start with keras. there are a few use cases(for example, building tools on top of tensor flow or developing your own high performance platform).that require the low level tensorflow core APIs. But if you use case doesn’t fall into one of the core API application ,you should prefer keras.

KERAS API COMPONENTS:

The core data structures of keras are layers and models .a layer is a simple input/output transformation, and a model is a directed acyclic graph(DAG) of layers.

LAYERS: The tf.keras.layers.Layer class is the fundamental abstraction in keras. A layer encapsulates a state (weights) and some computation(defined in the tf.keras.layers.Layer.call method).

you can also use layers to handle data preprocessing tasks like into a model, either during or after training, which makes the model portable.

Models: A model is an object that groups layers together and then can be trained on data. this simplest type of model is the sequential model, which is a linear stack of layers. for more complex architectures, you can either use the keras functional APIs which lets you built arbitrary graph of layers, or use subclassing to write models for scratch.

The tf.keras .model class features built-in training and revolution methods:

* tf.keras.model.fit:trains the model for a fixed number of epchos.
* tf.keras.model.predict:generate output predictions for input samples.
* Tf.keras.model.evaluate;returns the loss and metric values for the model:configure via the tf.keras.model.compile method.

These methods give u access to the following builtin training features;

* Callbacks.you can leverage builtin callbacks for early stopping,model check pointing,and tensor board monitoring.you can also implement custom callbacks.
* Distributed training.you can easily scaleup your training to multiple GPUs,TPUs,or other devices.
* Step fusing.with the steps\_per\_execution argument in

single tf.function call,which greatly improves device utilization on TPUs.

For a detailed overview of how to use fit,see the training and evaluation guide.to learn how to customize the builtin training and evaluation loops,see customizing what happens in fit().

TO GET STARTED USING KERAS WITH TENSOR FLOW,CHECK OUT THE

FOLLOWING TOPICS:

* The sequential model
* The functional API
* The training and evaluation with builtin methods.
* Making new layers and models
* Serialization and saving
* Working with preprocessing layers
* Customizing what happens in fit()
* Writing a training loop for scratch
* Working with RNNs
* Understanding masking and padding
* Write your own callbacks
* Tranfer learning and fine-tuning
* Multi-GPU and distributed training.

CONVOLUTION NEURAL NETWORK:

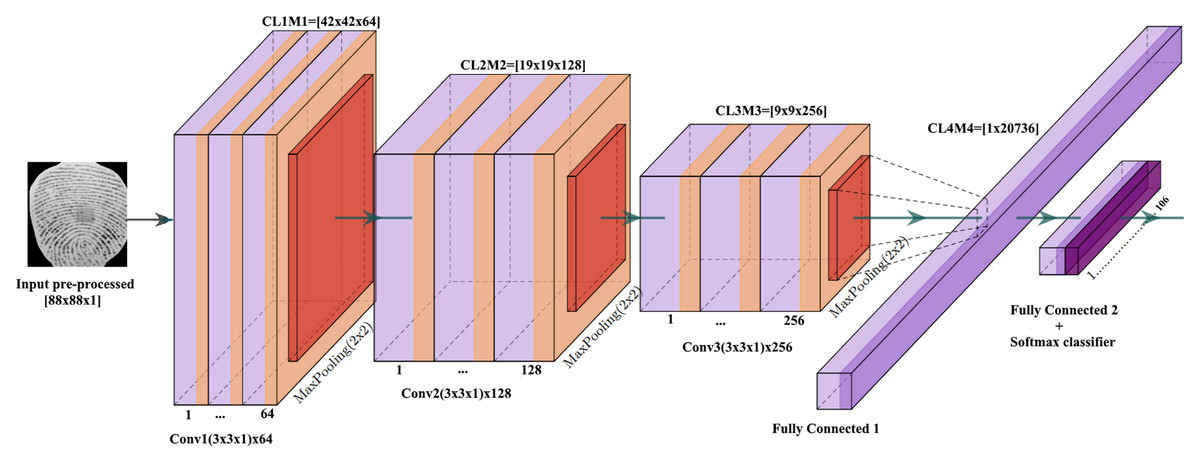
A convolutional neural network(CNN) is a type of deep learning neural network architecture commonly used in computer vision.computer vision is a field of artificial intelligence that enables a computer to understand and interpret the image of visual data.

When it comes to machine learning,artificial neural networks perform really well.neural networks are used in various data sets like images,audio and text.different types of neural networks are used for different purposes,for example for predicting the sequence og words we use recurrent neural networks more precisely and LSTM, similarily for image classification we use convolution neural networks.in this blog,we are going to build a basic building block for CNN.

In a regular neural network there are 3 types of layers:

1. INPUT LAYERS:it’s the layer in which we give input to our model.the number of neurons in this layer is equal to the total number of features in our data(no: of pixels in the case of an image).
2. HIDDEN LAYER:the input from the input layer is then feed into the hidden layer.there can be many hidden layers depending upon our model and data size.each hidden layer can have different numbers of neurons which are generally graeterr than the number of features. The output from each layer can have different numbers of neurons which are generally greater than the number of features.the output from each layer is computed by matrix multiplication of output of the previous layer with learnerableweights of that layer and then by the addition of learnerable biasis fllowed by activation function which makes the network non linear.
3. OUTPUT LAYER:the output from the hidden layer is the fed into a logistic function like sigmoid or soft max which converts the output of each class into the probability score of each class.

The data is fed into the model and output from each layer is obtained from the above step is called feedforward, we then calculate the erroe using an error function,some common eerrrroe functions are cross-entropy, squareloss error,etc.the error function measures how well the network is performing.after that ,we back propogate into the model by calculating the derivatives. This step is called backpropogation which basically used to minimize the lose.



[This Photo](https://peerj.com/articles/cs-248/) by Unknown Author is licensed under [CC BY](https://creativecommons.org/licenses/by/3.0/)

OPENCV:

OpenCV is a library of programming functions mainly for ral time computer vision.orginally developed by intel,it was later supported by willow garage, then itseez. The library is cross platform and licensed as free and open source software under apache licence2.

PROGRAM:

#import necessary libraries

Import tensorflow as tf

From tensorflow.keras.preprocessing import image

From tensorflow.keras.applications.mobilenet\_v2 import mobilenetv2, preprocess\_input,decode\_predictions

Import numpy as np

#load pretrained mobile netv2 model

Model=mobilenetv2(weights=’image net’)

#function to perform image classification

Def classify\_image(image\_path):

Img=image.load\_img(image\_path,target\_size=(224,224))

Img\_array=image.img\_to\_array(img)

Img\_array=np.expand\_dims(img\_array,axis=0)

Img=array=preprocess\_input(img\_array)

Predictions=model.predict(img\_array)

Decoded\_predictions=decode\_predictions(predictions,top=3)[0]

Print(“predictions:”)

For i,(imagenet\_id,label,score)in enumerate(decoder\_predictions):

Print(f”{i+1}: {label}({score:.2f})”)

#example usage

Image\_path=’path/to/your/image.jpg’

Classify\_image(image\_path)