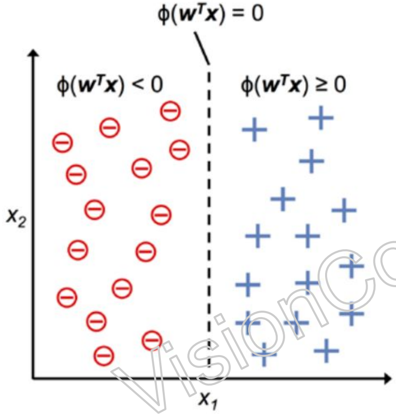
Deep Neural Networks (DNN)

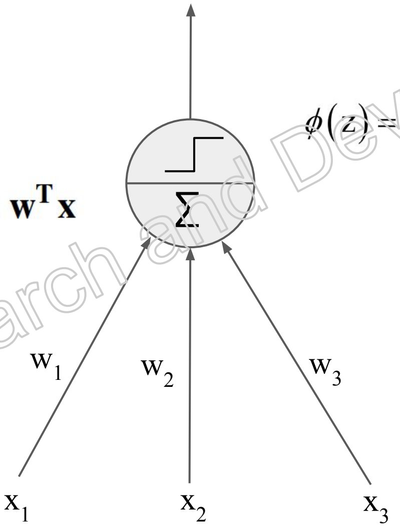
Dr. Ram Prasad K
VisionCog R&D

ram.krish@visioncog.com
https://www.visioncog.com

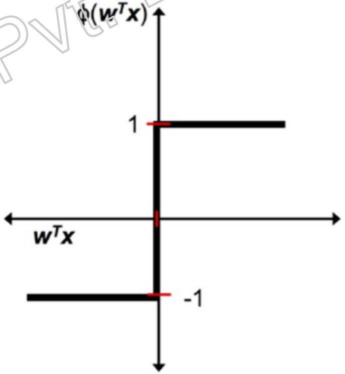
Linear Threshold Unit (LTU)



$$\sum_{j=1}^{n} w_j x_i = \mathbf{w}^{\mathbf{T}}\mathbf{x}$$



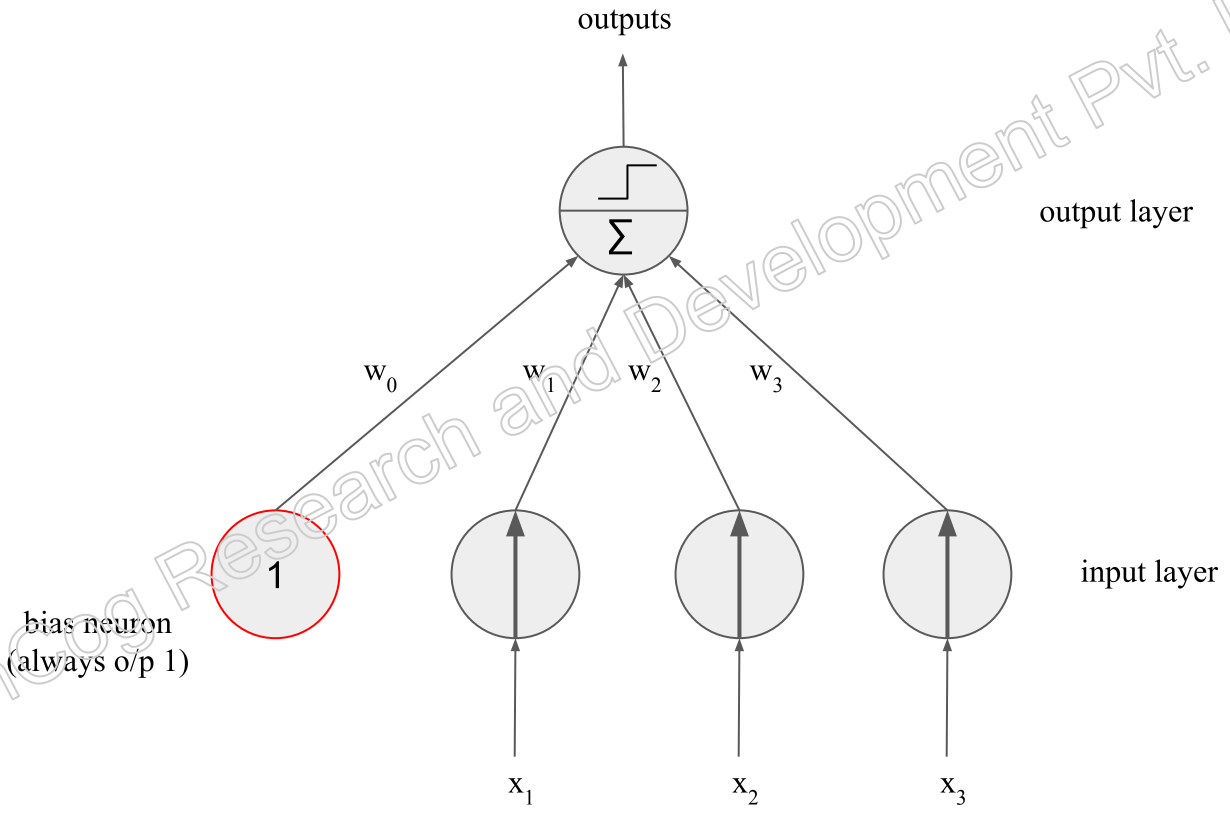
$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{otherwise} \end{cases}$$



VisionCog Research and Development Pvt. Ltd.

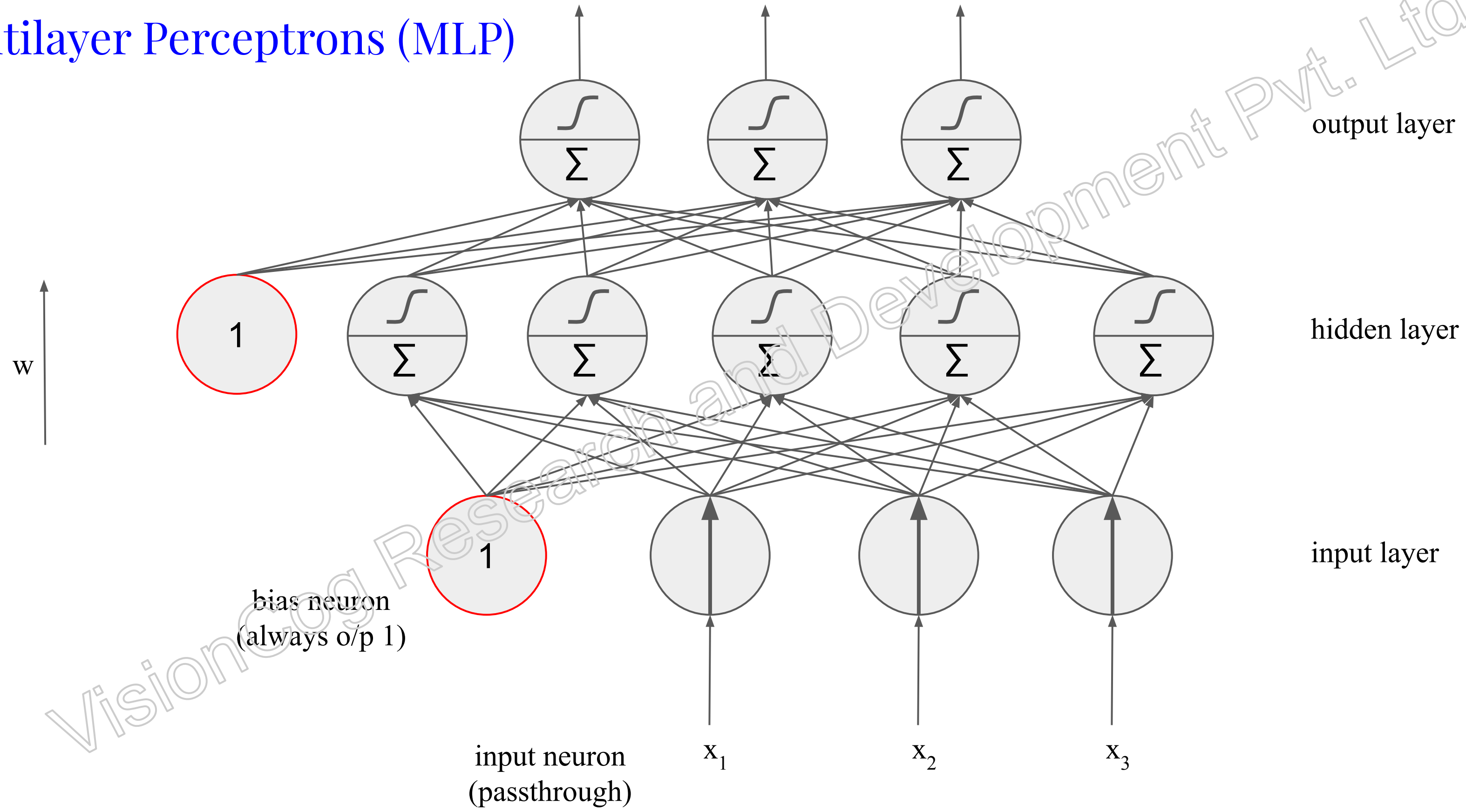
DEEPNETS

Perceptron



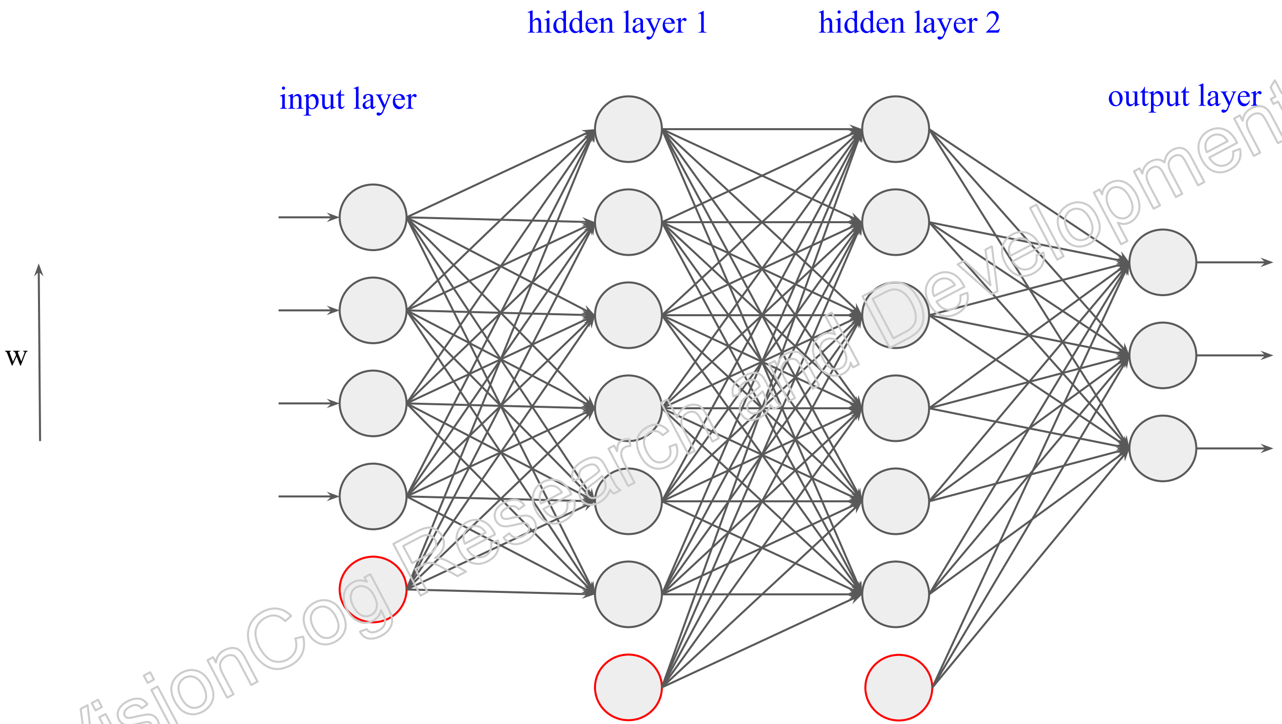
VisionCog Research and Development Pvt. Ltd.

Multilayer Perceptrons (MLP)



VisionCog Research and Development Pvt. Ltd.

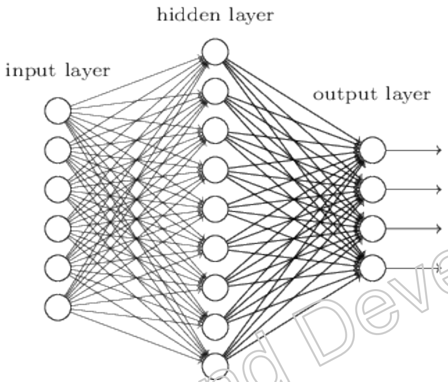
DeepNets



VisionCog Research and Development Pvt. Ltd.

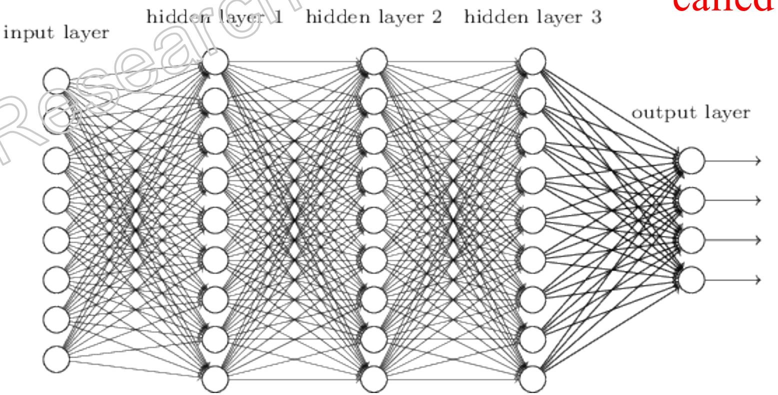
DeepNets

Multilayer Perceptrons (MLP)



When MLP contains ***two or more hidden*** layers, then such an MLP is called ***Deep Neural Network*** (DNN)

Deep Neural Network (DNN)



VisionCog Research and Development Pvt. Ltd.

FLOWER CLASSIFICATION

DeepNets

Flower Classification

daisy

VisionCog Research and Development Pvt. Ltd.

DeepNets

Flower Classification

dandelion

VisionCog Research and Development Pvt. Ltd.

Flower Classification

rose



VisionCog Research and Development Pvt. Ltd.

Flower Classification

sunflower

VisionCog Research and Development Pvt. Ltd.

Flower Classification

tulip

VisionCog Research and Development Pvt. Ltd.

DEEPNETS

http://download.tensorflow.org/example_images/flower_photos.tgz (3,670)

Tiny version

https://www.visioncog.com/rpk/tiny_FR.zip (500)

Flower Classification
(100 each, size of image varies)

- daisy
- dandelion
- rose
- sunflower
- tulip

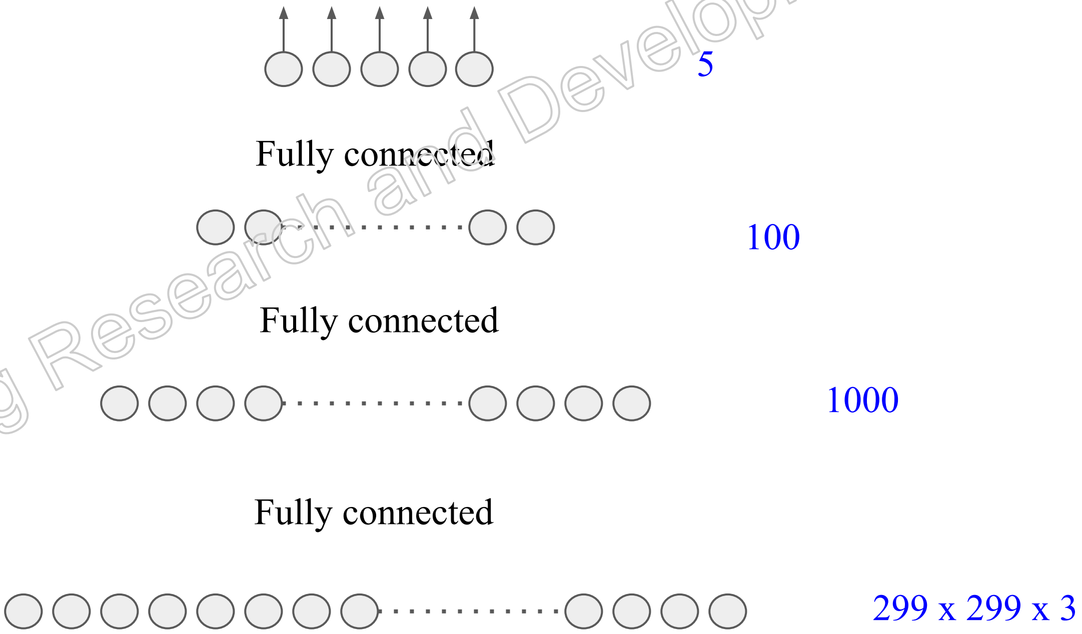




VisionCog Research and Development Pvt. Ltd.

DeepNets

Build a DNN for Flower classification as shown below:



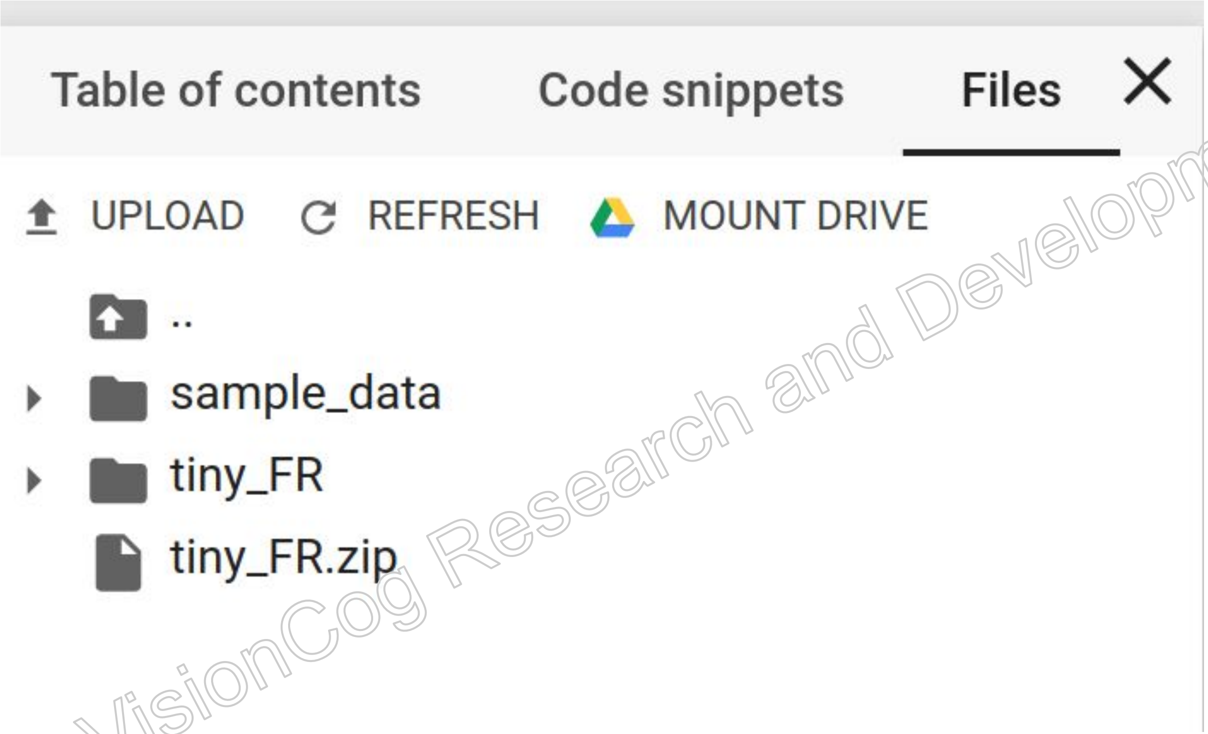
VisionCog Research and Development Pvt. Ltd.

DeepNets

```
# Original dataset
# http://download.tensorflow.org/example_images/flower_photos.tgz

# Download tiny version of the dataset from VisionCog website
# After download and unzip, remember to comment the following two lines.
```

```
!wget https://www.visioncog.com/rpk/tiny_FR.zip
!unzip tiny_FR.zip
```

VisionCog Research and Development Pvt. Ltd.



VisionCog Research and Development Pvt. Ltd.

```
# Original dataset
# http://download.tensorflow.org/example_images/flower_photos.tgz

# Download tiny version of the dataset from VisionCog website
# After download and unzip, remember to comment the following two lines.

#!wget https://www.visioncog.com/rpk/tiny_FR.zip
#!unzip tiny_FR.zip
```

VisionCog Research and Development Pvt. Ltd.

DEEPNETS

```
# Install TensorFlow
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass

import tensorflow as tf
print(tf.__version__)


# TensorFlow 2.x selected.
# 2.0.0-rc1
```

VisionCog Research and Development Pvt. Ltd.

```
from tensorflow import keras
tf.random.set_seed(42)

import numpy as np
np.random.seed(42)

import matplotlib.pyplot as plt
%matplotlib inline

import glob
import PIL
from PIL import Image
```

VisionCog Research and Development Pvt. Ltd.

```
imgFiles = glob.glob("tiny_FR/*/*.jpg")

for items in imgFiles[:5]:
  print(items)

# tiny_FR/sunflower/1715303025_e7065327e2.jpg
# tiny_FR/sunflower/2442985637_8748180f69.jpg
# tiny_FR/sunflower/27466794_57e4fe5656.jpg
# tiny_FR/sunflower/40411019_526f3fc8d9_m.jpg
# tiny_FR/sunflower/253586685_ee5b5f5232.jpg
```

VisionCog Research and Development Pvt. Ltd.

```
X = []
y = []

for fName in imgFiles:

  X_i = Image.open(fName) # tiny_FR/sunflower/1715303025_e7065327e2.jpg (500, 333)

  X_i = X_i.resize((299,299)) # To make them approriate to Xception model when using Transfer Learning

  X_i = np.array(X_i) / 255.0 # Normalize to range 0.0 to 1.0 (not stretching, only scaling)


  X.append(X_i)


  label = fName.split('/') # ['tiny_FR', 'sunflower', '1715303025_e7065327e2.jpg']

  y_i = label[1] # 'sunflower'


  y.append(y_i)
```

VisionCog Research and Development Pvt. Ltd.

```
print(set(y))
# {'daisy', 'sunflower', 'dandelion', 'rose', 'tulip'}


from sklearn.preprocessing import LabelEncoder

lEncoder = LabelEncoder()
y = lEncoder.fit_transform(y)

print(set(y))
# {0, 1, 2, 3, 4}

print(lEncoder.classes_)
# ['daisy' 'dandelion' 'rose' 'sunflower' 'tulip']
```

VisionCog Research and Development Pvt. Ltd.

```
X = np.array(X)
y = np.array(y)

print(X.shape)
# (500, 299, 299, 3)

print(y.shape)
# (500,)

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    stratify=y, random_state=42)
```

VisionCog Research and Development Pvt. Ltd.

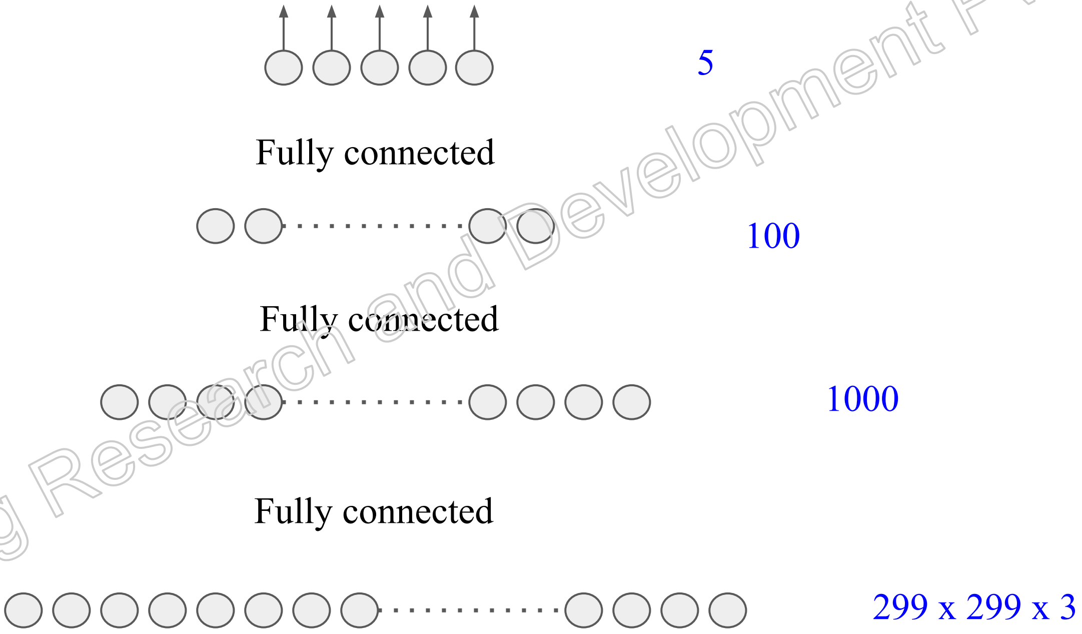
```
print("X_train_shape: {}".format(X_train.shape))
# X_train_shape: (400, 299, 299, 3)

print("X_test_shape: {}".format(X_test.shape))
# X_test_shape: (100, 299, 299, 3)


# Standard scaling
mu = X_train.mean()
std = X_train.std()

X_train_std = (X_train-mu)/std
X_test_std = (X_test-mu)/std
```

VisionCog Research and Development Pvt. Ltd.

```python
# Create the network using Functional API method

input_ = keras.layers.Input(shape = X_train.shape[1:])

x = keras.layers.Flatten()(input_)
x = keras.layers.Dense(units=1000, activation='relu')(x)
x = keras.layers.Dense(units=100, activation='relu')(x)

output_ = keras.layers.Dense(units=5, activation='softmax')(x)

model_DNN = keras.models.Model(inputs=[input_], outputs=[output_])
```

VisionCog Research and Development Pvt. Ltd.

DEEPNETS



VisionCog Research and Development Pvt. Ltd.

DEEPNETS

```
model_DNN.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
input_1 (InputLayer)        [(None, 299, 299, 3)]     0
_________________________________________________________________
flatten (Flatten)           (None, 268203)            0
_________________________________________________________________
dense (Dense)               (None, 1000)              268204000
_________________________________________________________________
dense_1 (Dense)             (None, 100)               100100
_________________________________________________________________
dense_2 (Dense)             (None, 5)                 505
=================================================================
Total params: 268,304,605
Trainable params: 268,304,605
Non-trainable params: 0
_________________________________________________________________
```

VisionCog Research and Development Pvt. Ltd.

DeepNets

```
model_DNN.compile(loss='sparse_categorical_crossentropy',
                  optimizer='adam', metrics=['accuracy'])
```

```
history_DNN = model_DNN.fit(X_train, y_train, epochs=25,
                            validation_split=0.1, batch_size=16)
```

```
Train on 360 samples, validate on 40 samples

Epoch 1/25
360/360 [===]-3s 9ms/sample - loss: 155.1205 - accuracy: 0.2278 - val_loss: 61.1286 - val_accuracy: 0.2750
Epoch 2/25
360/360 [===]-2s 5ms/sample - loss: 30.8947 - accuracy: 0.3750 - val_loss: 23.5717 - val_accuracy: 0.2750
Epoch 3/25
360/360 [===]-2s 5ms/sample - loss: 12.8762 - accuracy: 0.4889 - val_loss: 16.7457 - val_accuracy: 0.4000

...

Epoch 24/25
360/360 [===] - 2s 5ms/sample - loss: 0.4303 - accuracy: 0.9333 - val_loss: 11.7746 - val_accuracy: 0.3750
Epoch 25/25
360/360 [===] - 2s 5ms/sample - loss: 0.1650 - accuracy: 0.9611 - val_loss: 10.3161 - val_accuracy: 0.3250
```

VisionCog Research and Development Pvt. Ltd.

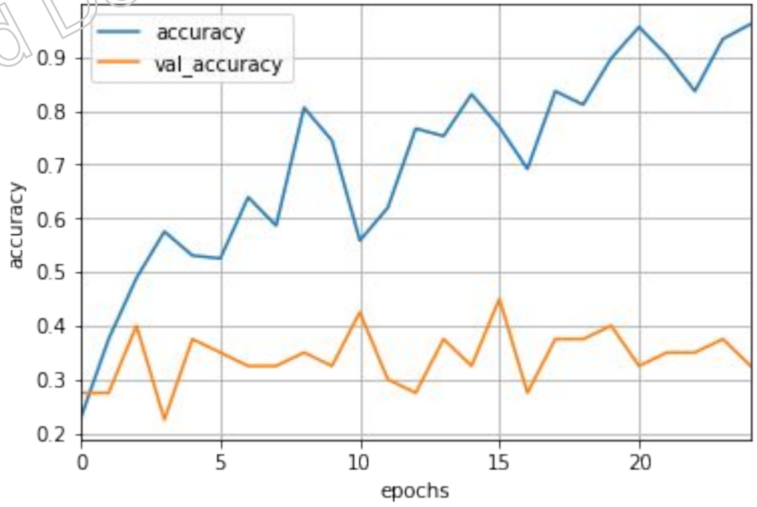
DeepNets

```
Train on 360 samples, validate on 40 samples

Epoch 1/25
360/360 [===]-3s 9ms/sample - loss: 155.1205 - accuracy: 0.2278 - val_loss: 61.1286 - val_accuracy: 0.2750
Epoch 2/25
360/360 [===]-2s 5ms/sample - loss: 30.8947 - accuracy: 0.3750 - val_loss: 23.5717 - val_accuracy: 0.2750
Epoch 3/25
360/360 [===]-2s 5ms/sample - loss: 12.8762 - accuracy: 0.4889 - val_loss: 16.7457 - val_accuracy: 0.4000

...

Epoch 24/25
360/360 [===] - 2s 5ms/sample - loss: 0.4303 - accuracy: 0.9333 - val_loss: 11.7746 - val_accuracy: 0.3750
Epoch 25/25
360/360 [===] - 2s 5ms/sample - loss: 0.1650 - accuracy: 0.9611 - val_loss: 10.3161 - val_accuracy: 0.3250
```

VisionCog Research and Development Pvt. Ltd.

DeepNets

```
keys = ['accuracy', 'val_accuracy']
progress = {k:v for k,v in history_DNN.history.items() if k in keys}

import pandas as pd
pd.DataFrame(progress).plot()

plt.xlabel("epochs")
plt.ylabel("accuracy")

plt.grid(True)
plt.show()
```



VisionCog Research and Development Pvt. Ltd.

```
test_loss, test_accuracy = model_DNN.evaluate(X_test, y_test)
# 100/1 [===] - 0s 1ms/sample - loss: 13.9873 - accuracy: 0.3800

print("Test-loss: %f, Test-accuracy: %f" % (test_loss, test_accuracy))
# Test-loss: 12.608617, Test-accuracy: 0.380000
```

VisionCog Research and Development Pvt. Ltd.