Experiment 2: multilayer perceptron

```
//mlp//
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Activation
import matplotlib.pyplot as plt
(x_train,y_train), (x_test,y_test)=tf.keras.datasets.mnist.load_data()
x_train = x_train.astype('float32')
x_test=x_test.astype('float32')
gray_scale=255
x_train/=gray_scale
x_test/=gray_scale
print("Feature matrix: ",x_train.shape)
print("Target matrix: ",x_test.shape)
print("Feature matrix: ",y_train.shape)
print("Target matrix: ",y_test.shape)
fig, ax = plt.subplots(10,10)
k=0
for i in range(10):
    for j in range(10):
        ax[i] [j].imshow(x_train[k].reshape(28,28),aspect='auto')
        k+=1
plt.show()
model=Sequential([
    Flatten(input_shape=(28,28)),
    Dense(256, activation='sigmoid'),
    Dense(128, activation='sigmoid'),
    Dense(20, activation='sigmoid'),
])
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['a
ccuracy'])
model.fit(x_train,y_train,epochs=10,batch_size=2000,validation_split=0.2)
results=model.evaluate(x_test,y_test,verbose=0)
print('test los,test acc:',results)
```

experiment :3 cnn

```
//cnn//
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten(input_shape=(32,32,1)))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10,activation='softmax'))
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=2,
                    validation_data=(test_images, test_labels))
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print(test_acc)
```

experiment:4 rnn

```
//rnn//
import numpy as np
import tensorflow_datasets as tfds
import tensorflow as tf
dataset, info = tfds.load('imdb_reviews',with_info =True,as_supervised=True)
train_dataset, test_dataset = dataset['train'], dataset['test']
train_dataset.element_spec
for example, label in train_dataset.take(1):
  print('text: ',example.numpy())
  print('label: ',label.numpy())
  BUFFER_SIZE =10000
  BATCH_SIZE = 64
  train_dataset = train_dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE).prefetch
(tf.data.AUTOTUNE)
  test_dataset = test_dataset.batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
  VOCAB_SIZE = 1000
  encoder = tf.keras.layers.TextVectorization(max_tokens=VOCAB_SIZE)
  encoder.adapt(train_dataset.map(lambda text, label: text))
  model = tf.keras.Sequential([encoder,
                           tf.keras.layers.Embedding(
                              input_dim=len(encoder.get_vocabulary()),
                              output_dim=64,mask_zero=True),
                           tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
                           tf.keras.layers.Dense(64, activation='relu'),
                           tf.keras.layers.Dense(1)
                           ])
  sample_text = ('The movie was cool. The animation and the graphics were out of th
e world. I would rcommend this movie')
  prediction = model.predict(np.array([sample_text]))
  print(prediction[0])
  print("$$$$$$$$$$$")
  model.compile(loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              optimizer=tf.keras.optimizers.Adam(1e-4),
              metrics=['accuracy'])
  history = model.fit(train_dataset, epochs=2,
                    validation_data=test_dataset,
                    validation_steps=30)
  test_loss, test_acc = model.evaluate(test_dataset)
  print('Test Loss:',test_loss)
  print('Test Accuaracy:', test_acc)
```

experiment :5 visualisaing DL model

```
//visual dl model//
import tensorflow as tf
import visualkeras
from tensorflow import keras
from keras.models import Sequential
from tensorflow.keras.layers import Input , Conv2D, Dense,Flatten,Dropout
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.models import Model
model=Sequential()
model.add(Conv2D(64,(4,4),input_shape=(32,32,3),activation ='relu', padding='same
'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(128,(4,4),input_shape=(32,32,3),activation ='relu', padding='sam
e'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.summary()
visualkeras.layered_view(model)
```

experiment:6 Loading and saving

```
//loading saving//
import tensorflow as tf
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Activation
import matplotlib.pyplot as plt
(x_train, y_train), (x_test,y_test) = tf.keras.datasets.mnist.load_data()
x_train = x_train.astype("float32")
x_test = x_test.astype("float32")
gray_scale = 255
x_train /= gray_scale
x_test /= gray_scale
model = Sequential([
Flatten(input_shape =(28,28)),
Dense(256, activation ="sigmoid"),
Dense(256, activation="sigmoid"),
Dense(10, activation="sigmoid"),
])
model.summary()
model.save_weights("MLPWeights.h5")
print("Model saved!")
savedModel = model.load_weights("MLPWeights.h5")
print("Model loaded!")
```

experiment 7: plot a curve

//plot a curve//

```
import matplotlib.pyplot as plt
import tensorflow as tf
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten,Dense,Activation
from sklearn import datasets
from sklearn.model_selection import train_test_split
a=datasets.load_breast_cancer()
x=a.data
y=a.target
model=Sequential([Dense(32,activation='relu'),
                  Dense(32,activation='relu'),
                  Dense(1,activation='sigmoid')])
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.35)
model.compile(optimizer=tf.keras.optimizers.RMSprop(lr=0.01),loss='binary_crossentropy',m
etrics=['accuracy'])
history=model.fit(x_train,y_train,epochs=10,batch_size=1,validation_data=(x_test,y_test))
his=history.history
loss=his['loss']
vloss=his['val_loss']
acc=his['accuracy']
vacc=his['val_accuracy']
epochs=range(1,len(loss)+1)
fig,ax=plt.subplots(1,2)
ax[0].plot(epochs,acc,'bo',label="Tacc")
ax[0].plot(epochs,vacc,'b',label="vacc")
ax[0].set_title("acc")
ax[0].set_xlabel("accu")
ax[0].set_ylabel("ep")
ax[0].legend()
ax[1].plot(epochs,loss,'bo',label="Tloss")
ax[1].plot(epochs,vloss,'b',label="vloss")
ax[1].set_title("los")
ax[1].set_xlabel("loss")
ax[1].set_ylabel("ep")
ax[1].legend()
```

9.drop out

```
//drop out//
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop
from keras.callbacks import EarlyStopping
from keras.datasets import mnist
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats
(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train.reshape(60000, 784).astype('float32') / 255
X_test = X_test.reshape(10000, 784).astype('float32') / 255
y_train = keras.utils.to_categorical(y_train, num_classes=10)
y_test = keras.utils.to_categorical(y_test, num_classes=10)
model = Sequential([
   Dense(512, activation='relu', input_shape=(784,)),
   Dropout(0.2),
   Dense(512, activation='relu'),
   Dropout(0.2),
   Dense(10, activation='softmax')
])
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
early_stopping_monitor = EarlyStopping(patience=3)
history = model.fit(X_train, y_train, batch_size=128, epochs=20,
                    callbacks=[early_stopping_monitor], verbose=1, validation_data=(X_test, y_test))
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

example 10: batch normalisation

```
//batch normalisation//
import tensorflow as tf
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten,Activation,Dense,BatchNormalization
(x_train,y_train),(x_test,y_test)=tf.keras.datasets.fashion_mnist.load_data()
x_train=x_train.astype('float32')
x_test=x_test.astype('float32')
x_train=x_train/255.0
x_test=x_test/255.0
a=x_train[:5000]
b=y_train[:5000]
model=Sequential([Flatten(input_shape=(28,28)),
                  Dense(300,activation='relu'),
                  BatchNormalization(),
                  Dense(200,activation='relu'),
                  BatchNormalization(),
                  Dense(100,activation='relu'),
                  BatchNormalization(),
                  Dense(10,activation='softmax')
                  ])
for i in model.layers[2].variables:
  print(i.name)
model.compile(optimizer='SGD',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
model.fit(x_train,y_train,epochs=10,validation_data=(a,b))
res=model.evaluate(x_test,y_test,verbose=0)
print(res)
```

experiment 11:resnet vgg inception

```
//resnet vgg inception//
from keras.utils.image_utils import img_to_array
from tensorflow.keras.applications.resnet50 import ResNet50,preprocess_input,decode_predictions
from tensorflow.keras.preprocessing import image
import numpy as np
model=ResNet50(weights='imagenet')
a='orange.jpeg'
img=image.load_img(a,target_size=(224,224))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
x=preprocess_input(x)
pred=model.predict(x)
print(decode_predictions(pred,top=3)[0])
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input,decode_predictions
from tensorflow.keras.preprocessing import image
model=VGG16(weights='imagenet')
b='orange.jpeg'
img=image.load_img(b,target_size=(224,224))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
x=preprocess_input(x)
pred=model.predict(x)
print(decode_predictions(pred,top=3)[0])
from tensorflow.keras.applications.inception_v3 import InceptionV3,preprocess_input,decode_predi
ctions
from tensorflow.keras.preprocessing import image
model=InceptionV3(weights='imagenet')
c='orange.jpeg'
img=image.load_img(c,target_size=(299,299))
x=image.img_to_array(img)
x=np.expand_dims(x,axis=0)
x=preprocess_input(x)
pred=model.predict(x)
print(decode_predictions(pred,top=3)[0])
```