AI IN DATA SCIENCE

INTRODUCTION TO KERAS

TASK TO BE SOLVED

1. Download the CIFAR10 dataset
2. Prepare the data
3. Implement Sequential neural network with Dense layers! (No CNN)
4. Implement both MSE loss and CE
5. Train the network

IMPORT MODULES

IMPORT MODULES

```
import numpy as np
import tensorflow as tf
import keras as k
from keras import layers
from keras import models
from keras import datasets
from keras import losses
from matplotlib import pyplot as plt
import sklearn
```

DOWNLOAD DATA

DOWNLOAD DATA

```
1 (x_train, y_train), (x_test, y_test) = //
2 datasets.cifar10.load_data()
3 print(x_train.shape)
4 print(y_train.shape)
```

DOWNLOAD DATA

```
1 (x_train, y_train), (x_test, y_test) = //
2 datasets.cifar10.load_data()
3 print(x_train.shape)
4 print(y_train.shape)
```

(50000, 32, 32, 3)

(50000, 1)

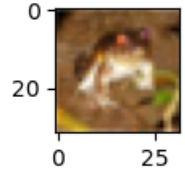
EXPLORING DATA

WHAT IS CIFAR10?

```
1 plt.figure(figsize=(1,1))
2 plt.imshow(x_train[0])
```

WHAT IS CIFAR10?

```
1 plt.figure(figsize=(1,1))
2 plt.imshow(x_train[0])
```



PROBLEMS

- RGB channel
- Unnormalized data
- Categorical target

PROBLEMS

- RGB channel
- Remove GB
- Unnormalized data
- /255
- Categorical target
- One-hot enecode it

PREPROCESS DATA

REMOVE GB

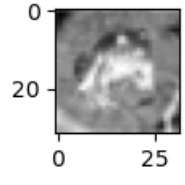
x_train.shape = (50000,32,32,3)

```
1 x_train = np.delete(x_train,1,axis=3)
```

```
x_train = np.delete(x_train,1,axis=3)
x_train = np.delete(x_train,1,axis=3)
x_test = np.delete(x_test,1,axis=3)
x_test = np.delete(x_test,1,axis=3)
print(x_train.shape)
plt.figure(figsize=(1,1))
plt.imshow(x_train[0])
```

```
x_train = np.delete(x_train,1,axis=3)
x_train = np.delete(x_train,1,axis=3)
x_test = np.delete(x_test,1,axis=3)
x_test = np.delete(x_test,1,axis=3)
print(x_train.shape)
plt.figure(figsize=(1,1))
plt.imshow(x_train[0])
```

(50000, 32, 32, 1)



NORMALIZE

```
1 x_train /= 255.0
2 x_test /= 255.0
```

PYTHON IS STRONGLY TYPED!

```
1 x_train = np.float32(x_train)
2 x_test = np.float32(x_test)
3 x_train /= 255.0
4 x_test /= 255.0
```

ONE-HOT ENCODING

```
1 categories_train_y = k.utils.to_categorical(y_train)
2 categories_test_y = k.utils.to_categorical(y_test)
3
4 print(categories_test_y)
5 number_of_labels = len(categories_test_y[0])
6 print(number_of_labels)
```

ONE-HOT ENCODING

```
1 categories_train_y = k.utils.to_categorical(y_train)
2 categories_test_y = k.utils.to_categorical(y_test)
3
4 print(categories_test_y)
5 number_of_labels = len(categories_test_y[0])
6 print(number_of_labels)
```

[[0. 0. 0. … 0. 0. 0.]
[0. 0. 0. … 0. 1. 0.]
[0. 0. 0. … 0. 1. 0.]
…]

ONE-HOT ENCODING

```
1 categories_train_y = k.utils.to_categorical(y_train)
2 categories_test_y = k.utils.to_categorical(y_test)
3
4 print(categories_test_y)
5 number_of_labels = len(categories_test_y[0])
6 print(number_of_labels)
```

[[0. 0. 0. ... 0. 0. 0.]
[0. 0. 0. ... 0. 1. 0.]
[0. 0. 0. ... 0. 1. 0.]
...]
10

CREATE THE MODEL

PURPOSE

- Predicting labels
- Using Sequential model ...
- ...and dense layers

SHAPES

INPUT SHAPE

(1,32,32,1)

INPUT SHAPE

(1,32x32)

INPUT SHAPE

```
1 np_reshaped_train = x_train.reshape(50000,32*32)
2 np_reshaped_test = x_test.reshape(10000,32*32)
3 print(np_reshaped_test.shape)
```

(10000, 1024)

LOSS

LOSS

- Regression vs category
- Defines the output
- Defines the metrics

LOSS

- Regression vs category
  - d("Frog", "Horse") < d("Frog", "Dog")
- Defines the output shape
  - CE: dim(output) = batch x #categories
- Defines the metrics
  - "accuracy" only if Confusion Matrix makes sense
  - Loss = universial metrics

LOSS

- CrossEntropyLoss()
- SparseCategoricalCrossentropy()
- MSELoss()

CONFUSION MATRIX

CONFUSION MATRIX

```
1 from sklearn.metrics import confusion_matrix
2 confusion_matrix(y_true, y_pred)
```

REGRESSION MODEL

DEFINING MODEL

```
my_MSE_model = models.Sequential()

Dense1 = layers.Dense(32,'relu',input_dim=1024)
Dense2 = layers.Dense(64,'relu')
DenseLast =layers.Dense(1)

my_MSE_model.add(Dense1)
my_MSE_model.add(Dense2)
my_MSE_model.add(DenseLast)

my_MSE_model.compile(optimizer="rmsprop", loss=losses.mean_squ
```

DEFINING MODEL

```
1 my_MSE_model.summary()
```

Layer(type)	Output shape	Param #
Dense	(None, 32)	32800
Dense	(None, 64)	2112
Dense	(None, 1)	65

DEFINING MODEL

```
my_Matrix_MSE_model = models.Sequential()

Dense1 = layers.Dense(32,'relu',input_shape= (32,32))
Dense2 = layers.Dense(64,'relu')
flat =layers.Flatten()
DenseLast = layers.Dense(1)

my_Matrix_MSE_model.add(Dense1)
my_Matrix_MSE_model.add(Dense2)
my_Matrix_MSE_model.add(flat)
my_Matrix_MSE_model.add(DenseLast)
my_Matrix_MSE_model.compile(optimizer="rmsprop", loss=losses.m
```

DEFINING MODEL

```
1 my_Matrix_MSE_model.summary()
```

Layer(type)	Output shape	Param #
Dense	(None, 32, 32)	1056
Dense	(None, 32, 64)	2112
Flatten	(None,2048)	0
Dense	(None, 1)	2049

WHAT IS THE DIFFERENCE?

- tf reshaping the object twice
- (32 x batch_size, last_input_dim)
- back to (batch_size, 32, num_neurons)
- When is it useful: time series!

PREDICT

```
1 pred = my_MSE_model.predict(np_reshaped_test)
2 pred = np.int16(pred)
3 print(pred)
```

[[3] [6] [4] ... [2] [3] [2]]

CATEGORICAL MODEL

CATEGORICAL MODEL

```
my_CE_model =models.Sequential([
layers.Flatten(input_shape = (32,32,1)),
layers.Dense(32,'relu'),
layers.Dense(64,'relu'),
layers.Dense(10,'softmax')
]
)
my_CE_model.compile(optimizer="rmsprop",
loss= losses.categorical_crossentropy,
metrics=['accuracy'])
```

CATEGORICAL MODEL

```
my_SCE_model = models.Sequential([
k.Input(shape=(32,32,1)),
layers.Flatten(),
layers.Dense(32,'relu'),
layers.Dense(64,'relu'),
layers.Dense(10,'softmax')
]
)
my_SCE_model.compile(optimizer="adam",
loss=losses.sparse_categorical_crossentropy,
metrics=['accuracy'])
```

WHAT IS THE DIFFERENCE?

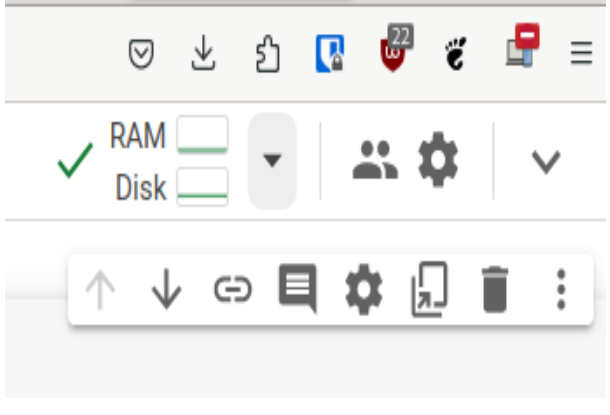
```
1 import tf.convert_to_tensor as cvt
2 x_test_tensor = cvt(x_test,dtype=tf.float32)
3 x_train_tensor = cvt(x_train,dtype=tf.float32)
4 y_train_tensor = cvt.convert_to_tensor(categories_train_y,dtype
5 y_test_tensor = cvt.convert_to_tensor(categories_test_y,dtype=t
6
7 my_CE_model.fit(x_train_tensor, y_train_tensor,
8 validation_data=(x_test_tensor,y_test_tensor), batch_size=3000,
9 epochs=10)
```

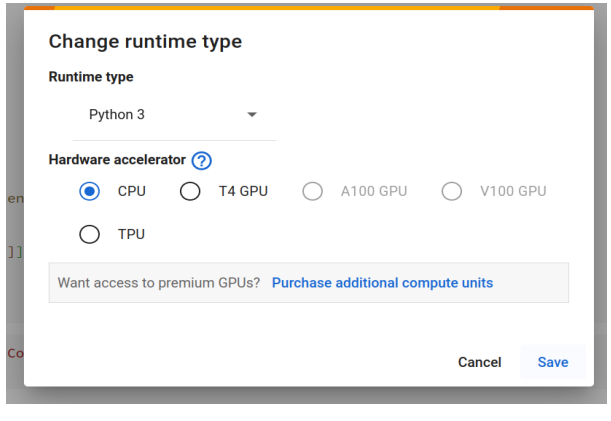
WHAT IS THE DIFFERENCE?

```
1 x_test_tensor = cvt(x_test,dtype=tf.float32)
2 x_train_tensor = cvt(x_train,dtype=tf.float32)
3 y_train_tensor = cvt(y_train,dtype=tf.float32)
4 y_test_tensor = cvt(y_test,dtype=tf.float32)
5
6 my_SCE_model.fit(x_train_tensor, y_train_tensor,
7 validation_data = (x_test_tensor,y_test_tensor),
8 batch_size=3000,epochs=10)
```

GPU TRAINING

- Only Nvidia is supported
- If you wish to use your own, install CUDA Toolkit
- You can use Colab, limited GPU access
  - Connect





CNN

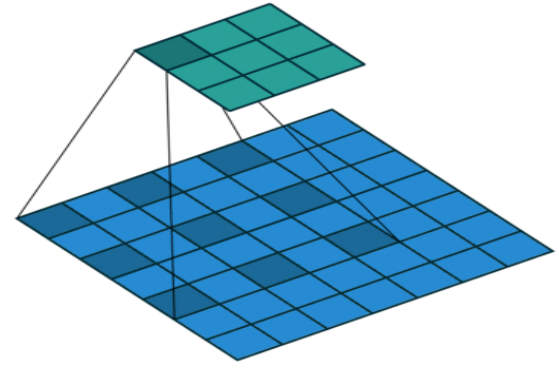
PARAMETERS FOR CNN

Kernel size

Padding

Stride

Dilation



Source:https://docs.huihoo.com/theano/0.9/tutorial/conv_arithmetic.html

2D CONVOLUTION

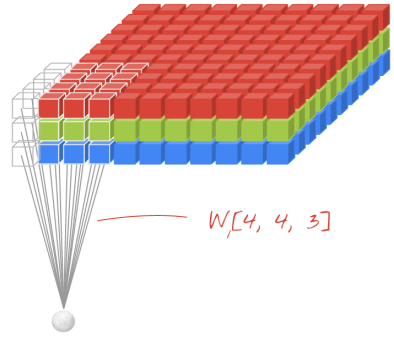
In reality 3D filter...

The dimension tells the direction

1D: X; 2D:(X,Y); 3D:(X,Y,Z)

Keras expects int or (int,int) as kernel size

But in reality it is 3D filter!


W[4, 4, 3]

BATCH NORMALIZATION, DROPOUT

Regularization techniques

Probability a neuron is ignored in the layer