Date.
Page No.

Deep Learning Frameworks

- ~~Tensor~~

In Deeplearning we have two-main deeplearning frameworks.

1 → Tensor Flow
2 → Pytorch

① Tensor Flow.

↳ it is an open source machine learning library for preprocessing data, modeling data and serving models

↳ it contains many of most common machine learning functions you will want to use.

introduction to tensors

⇒ if you have ever used numpy, tensors are like ~~to~~ numpy ~~arrs~~ arrays.

" for now you can think of tensors as a multi-dimensional ~~matr~~ numerical representation of somethings, this some things can be.

- numbers
- images
- text or it could be some other form of ~~any~~ information you want to represent with numbers

Difference b/w tensors and numpy array is tensors can be used on (GPUs and TPU's)

Date.
Page No.

0 Creating tensors with tf.constant()

① Scalar = tf.constant(8)
↳ Here constant creates a scalar value. Scalar.ndim → 0 for here.
Scalar ⇒ also known as rank 0 tensor. Because it has no dimensions (it's just a number)

② ~~Scalar~~ Vector = tf.constant([3,3])

Here using constant we are creating a vector of 1-dim → (1×2)
~~of 3,3~~ ↳ creates a 2-D array consist [1,2]

③ 2D Vector = tf.constant([[1,2],
[3,4]])

Here we are creating a 2-D array.
# by default tf.constant() creates ~~one~~ tensors with either int32 or float32 dtype.
# but we can also specify dtype as
dtype = tf.float16

tf.constant([1,2,3], dtype = tf.int16)

NOTE → We can create n-dim of tensors and n can be any positive number

Date.
Page No.

(2) Creating tensors with tf.variable()

So let's understand why we are learning tf.variable to create tensors when we do have tf.constant.

→ So tensors created with tf.constant are immutable (can't be changed, can only be used to create new tensors)

whereas tensors created with ~~tensor~~ tf.Variable are mutable (can be changed)

```
Vector = tf.Variable([2,3,4])
```

Syntax is same as of Constant.

To change value we use .assign()

```
Vector[1].assign(11)
```

(3) Creating Random Vectors

we can create random tensors using

```
tf.random.Generator.from_seed(42)
```

(4) How to shuffle the order of tensor

why we need shuffle

→ let's say we are working with 15000 images first 8000 are of cat and last 7000 are of dog, if we train it on ~~but~~ that way only we might get overfitting model for the orders so we shuffle it.

→ tf.random.shuffle(vector)

Date.
Page No.

(5) Others ways to create tensors

to create tensor with value 1
tf.ones(shape = (3,2))

to create tensor with value 0
tf.zeros(shape = (3,2))

(6) You can ~~also~~ also convert/create to tensors from numpy array

Only diff b/w them is tensors can be run on GPU's.

tensor = tf.constant(arr, shape = (2,4,3))

#(element in arr should be equal to 2x4x3)

(7) tf.transpose(tensor) → gives transpose ~~&~~ row ⇌ col.

(8) tf.reshape(2,4) → reshapes ~~a~~ tensor

(9) To change D-type of tensor

B = tf.cast(B, dtype = tf.int32)

Date.
Page No.

- Getting information from tensors
  - shape → length (no. of element) of each dimension of tensor
  - rank → The number of tensor dimensions
  - Axis or dimension → A particular dimension of a tensor
  - Size → The total number of items in tensor

* Manipulating tensors with basic operation

(1) Addition

```
tensor = tf.constant([1,2,3])
tensor + 2
    ↳ [3,4,5]
```

↳ it is giving us a copy, original remains unchanged as we have used tf.constant

```
tensor
    ↳ [1,2,3]
```

Similarly we can do (*, -, /) etc

→ tensor * 10

→ tf.multiply(tensor, 10)
    ↳ (10, 20, 30)

* Manipulating with matrix multiplication

(1) tf.multiply(tensor1, tensor2)

(2) tensor1 @ tensor2

Date.
Page No.

→ Getting absolute value

tf.abs(tensor)

get min, max, sum, mean

→ tf.reduce_min(tensor)
→ tf.reduce_max(tensor)
→ tf.reduce_mean(tensor)
→ tf.reduce_sum(tensor)
→ tf.reduce_variance(tensor)
→ tf.reduce_std(tensor)

to get index of min, max

tf.argmax(Tensor)

tf.argmin(tensor)