강의 1

Tensorflow API : Build and train models

Tensorflow Lite(for mobile), Tensorflow.js(for web) : deploy the model for use

사용할 것들

Tensorflow API

Python programming language

Colab : interactive python notebook hosted on google cloud (web)

//참고: enumerate()으로 list의 (index, value) 튜플 만들기

>> a = np.array([34, 56, 89])

For I, v in enumerate(a, start = 1):

Print(I, v)

Print(“Index is {} and corresponding value is {}” .format(l, v))

>> (1, 34), (2, 56), (3, 89)가 출력됨

이때, python의 for 문은 for (item) in (iterator) 형태로 구성되고, 반복 가능한 iterator(list, dict, str, tuple 등등)을 (iterator) 자리에 쓰면 됨.

그러면 next(iterator 이름) 함수를 연속해서 실행해 내부에 있는 값을 반환함.

이때, enumerator는, next를 실행했을 때 반환하는 iterator 객체를 애초에 (index, item)형태의 tuple로 바꿔주는 역할을 함!

#Colab 사용법

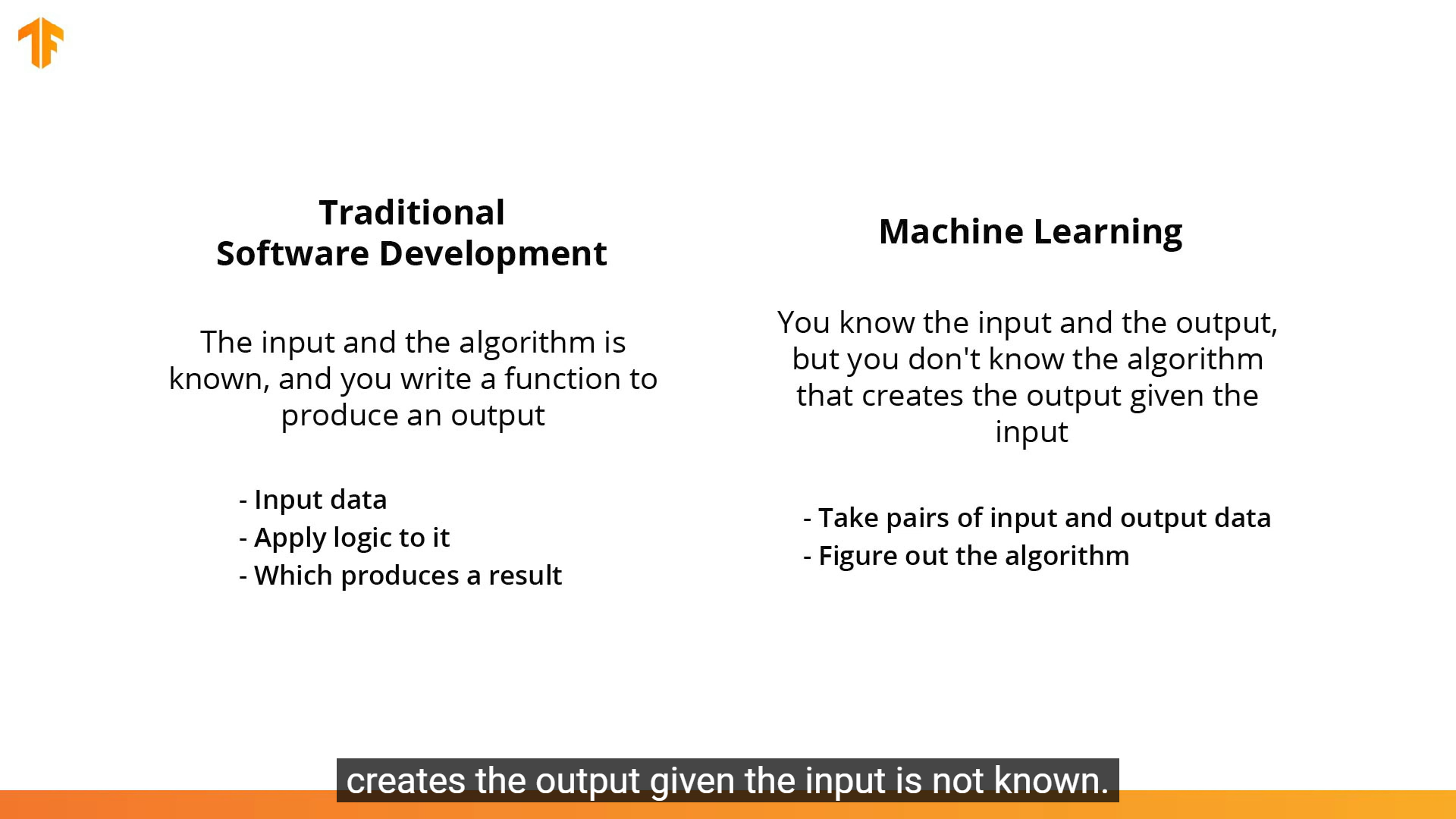
!쓴 후 linux command 쓸 수 있음

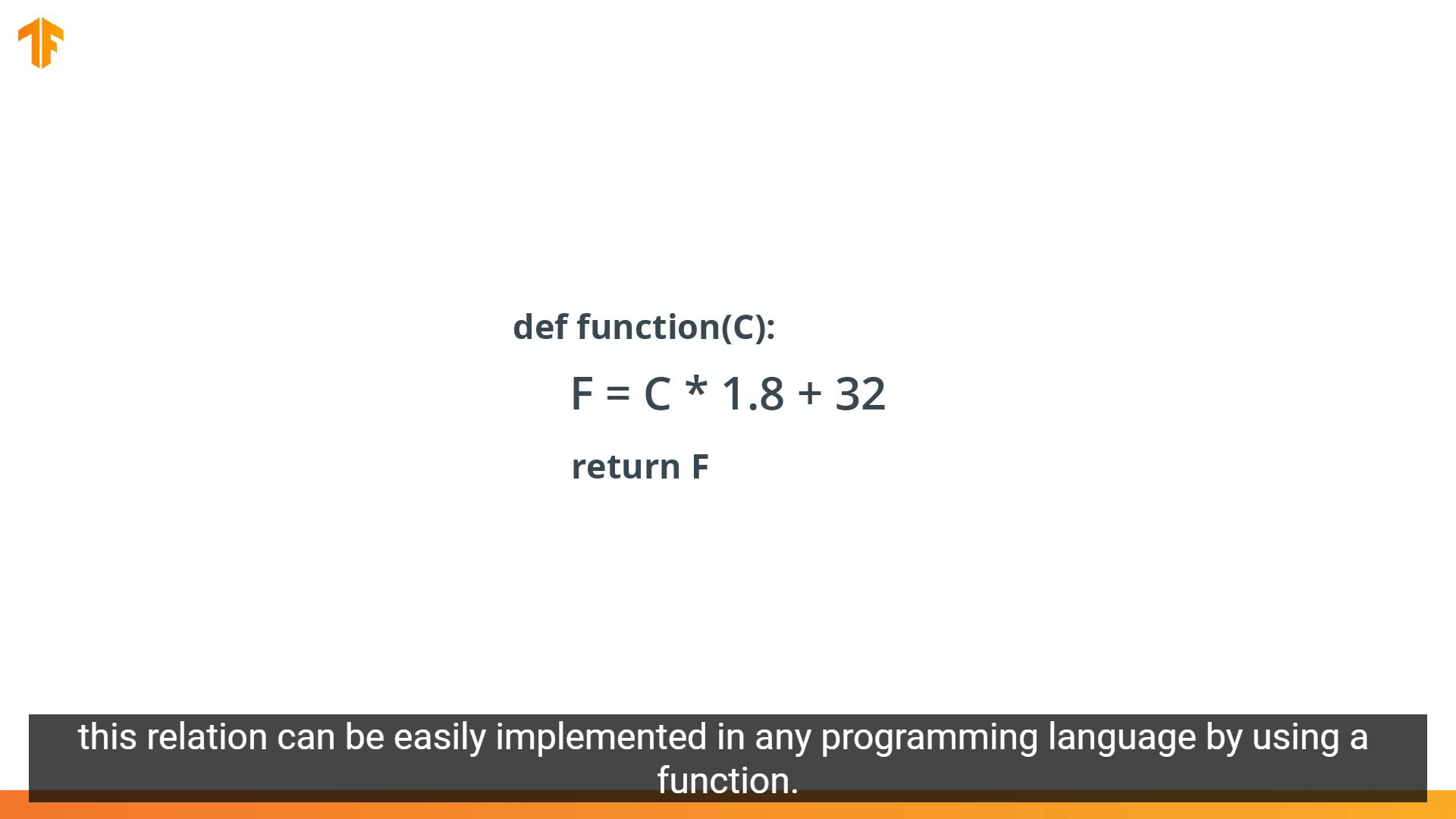
Runtime – clear all runtime : 여태까지 실행한 모든 거 초기화

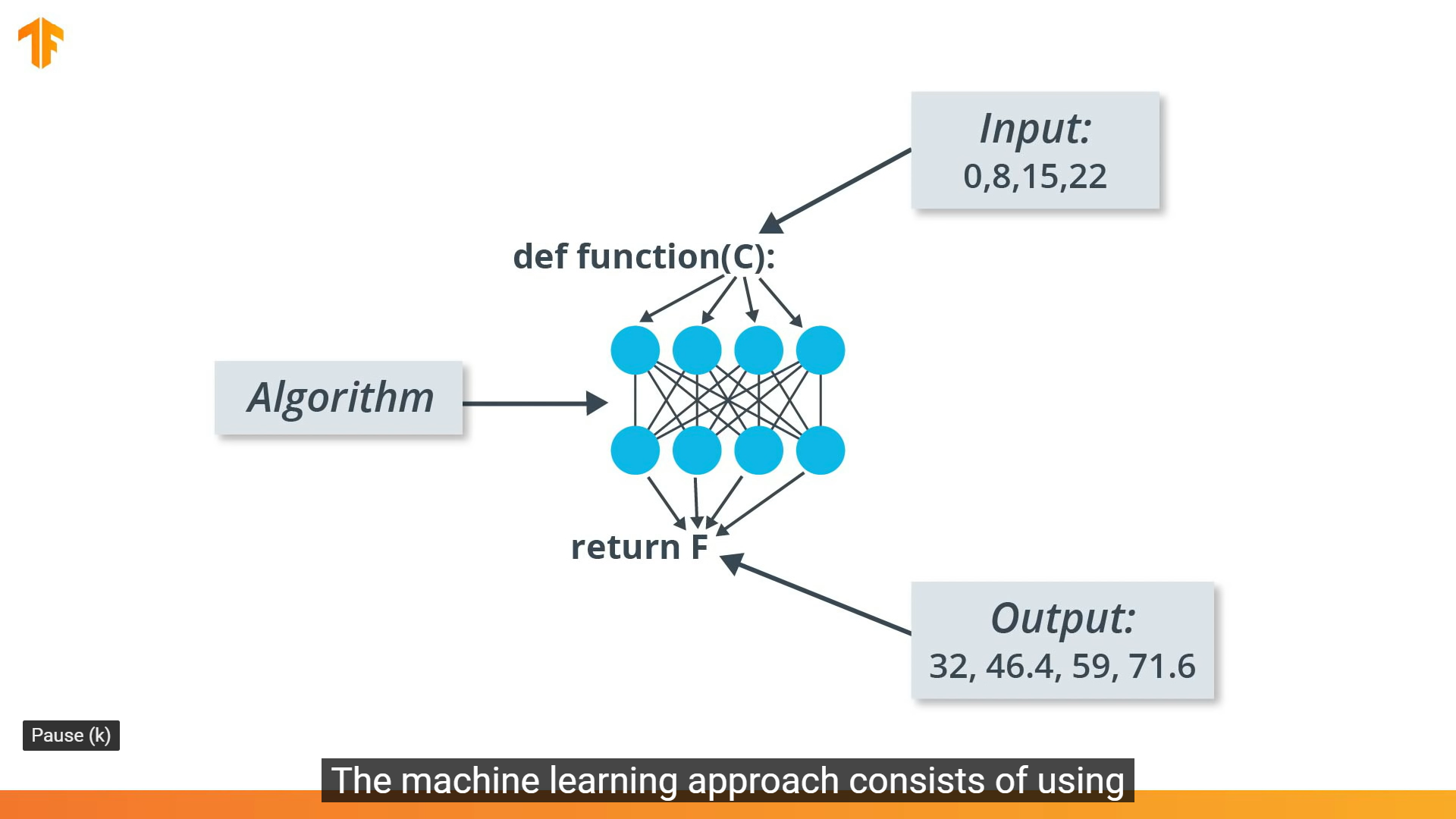
Edit – clear output: output 모두 지움

강의 2

Machine Learning: Input과 Output 세트를 알고 있을 때, 둘 사이의 관계를 도출해 내는 과정!







Neural networks = multiple layers with internal variables + calculations

>> Training: repeatedly letting the computer try to find a correct relation between input and outputs. Tweaking internal variables in the process -> 이거를 어떻게 하는지 알려주는 것

Terminology

Feature: inputs to our model

Labels: output that our model predicts

Example: a pair of input/outputs that is used in training!

Loss function: how far off the predictions are from desired outcomes – we used mean squared error here

Optimizer function: the function that adjusts internal variables to reduce loss

Weights: internal variables를 부르는 말 – 초기에는 랜덤하게 생성됨

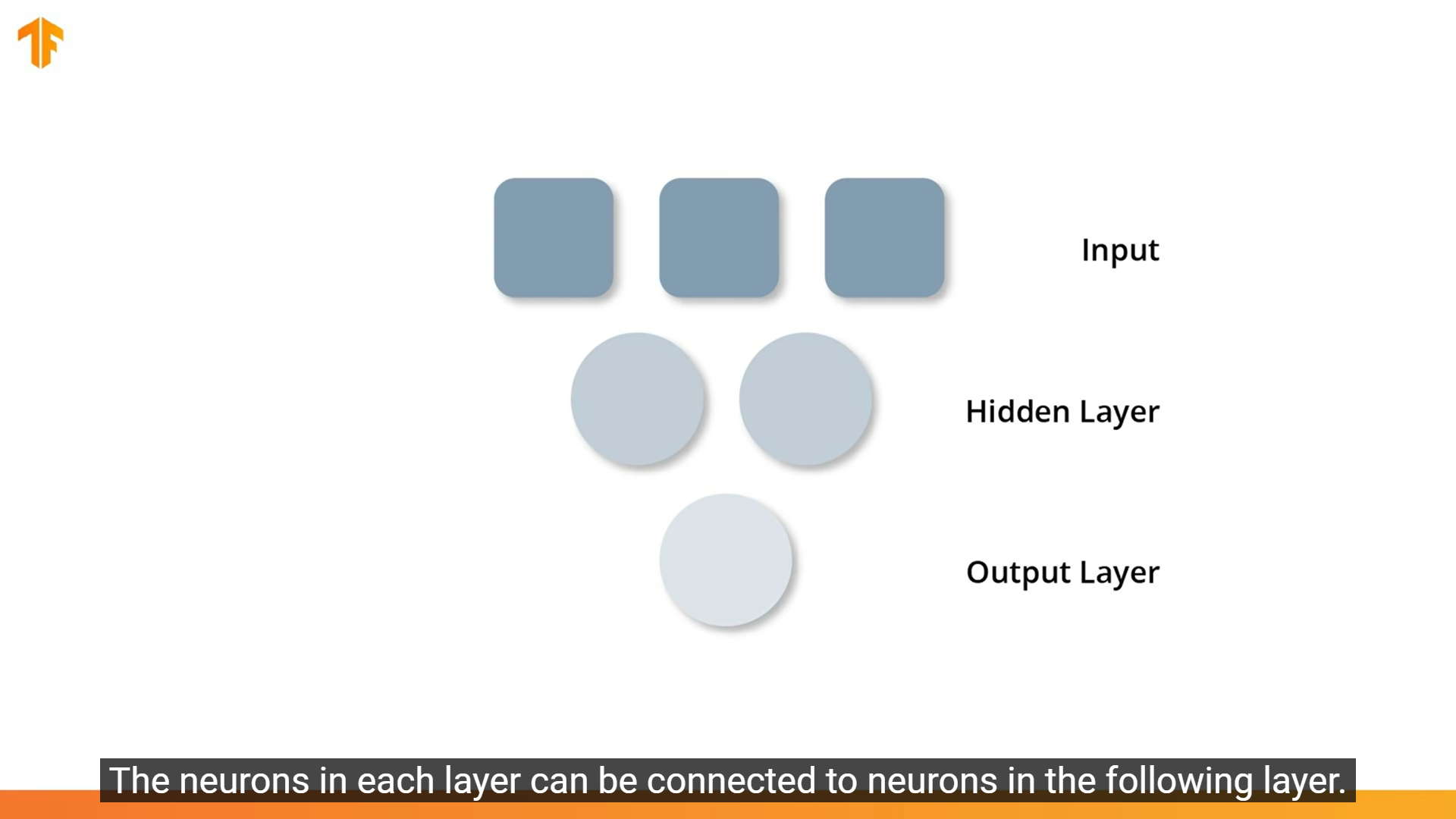
Creating a model

1. Import inputs and outputs : e.g. using one dimensional array
2. Build a layer & Build a model
3. Compile the model
4. Train the model
5. See model statistics – layer weights(internal variables), epoch to loss graph
6. Predict values with the model

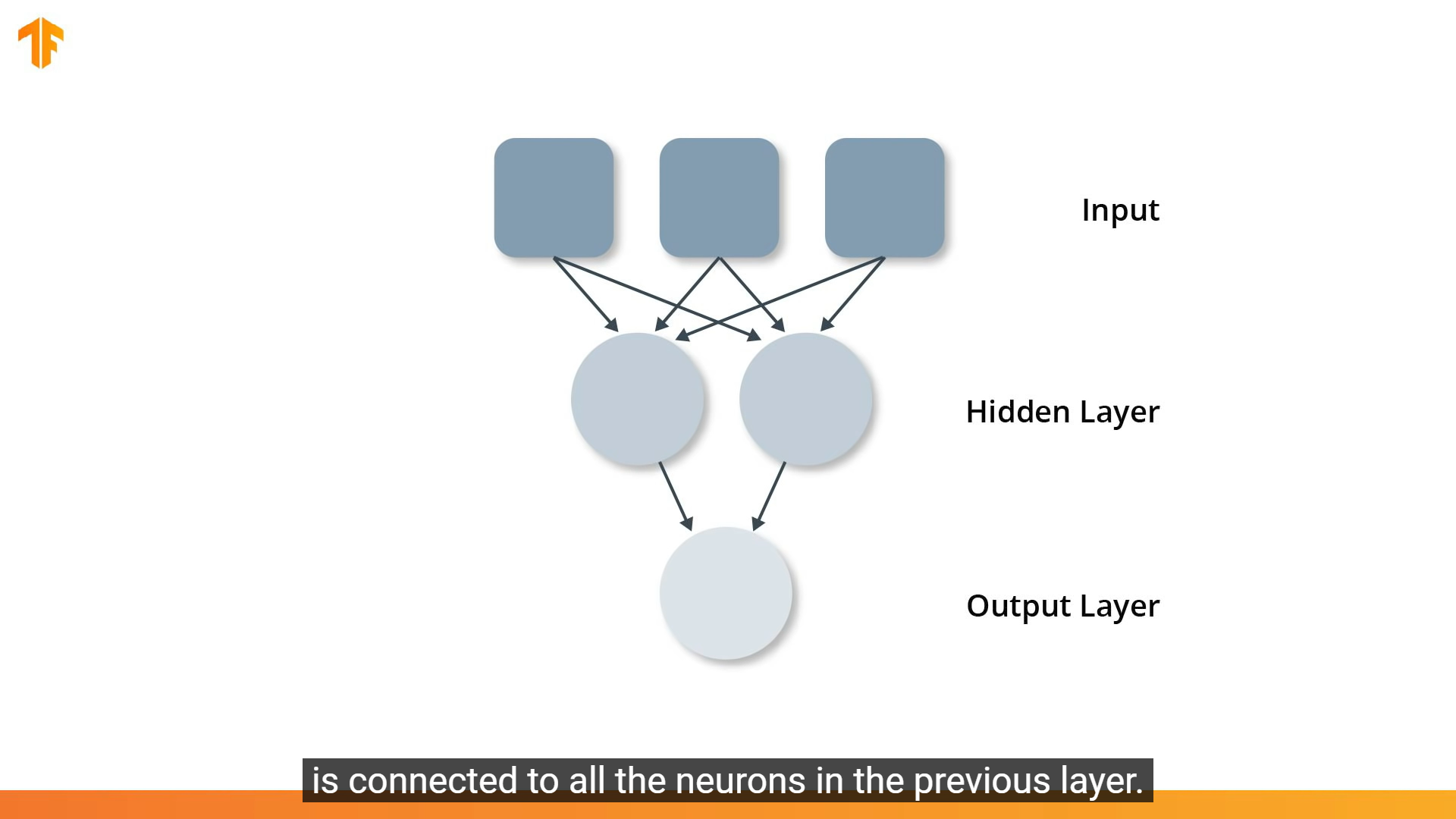
Training

1. Forward pass (acquires input)
2. Computes output with initial random internal var
3. Back propagation: 3번과 4번 - Calculates loss function
4. Tweaks internal variables(weights, biases) using gradient descent, with optimizer

What is Dense layer?

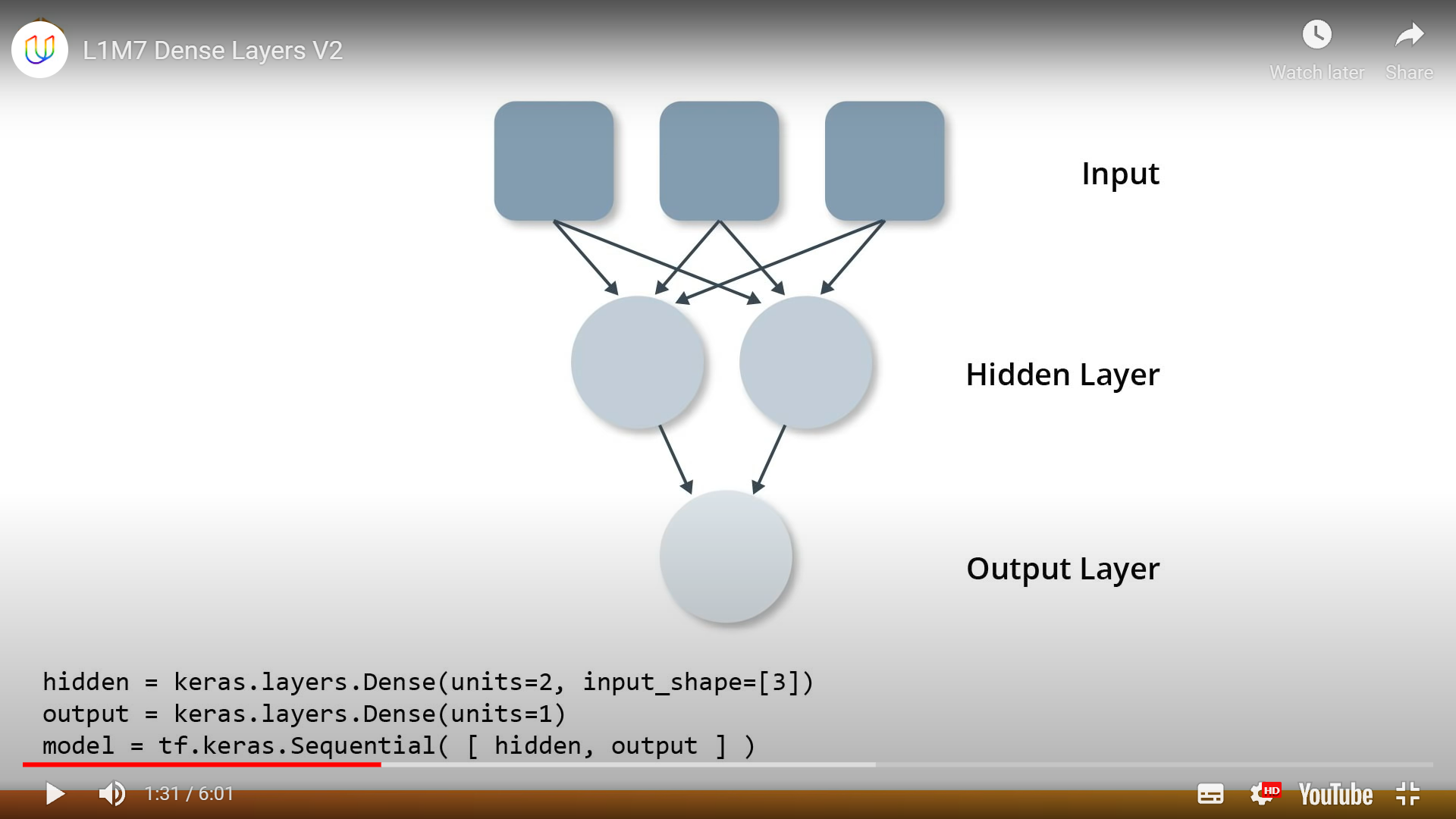


Model에 3개의 layer를 이렇게 만듦



Fully connected layer(Dense layer) = 각 레이어의 unit이 그 다음 layer의 unit과 모두 연결되어 있는 경우

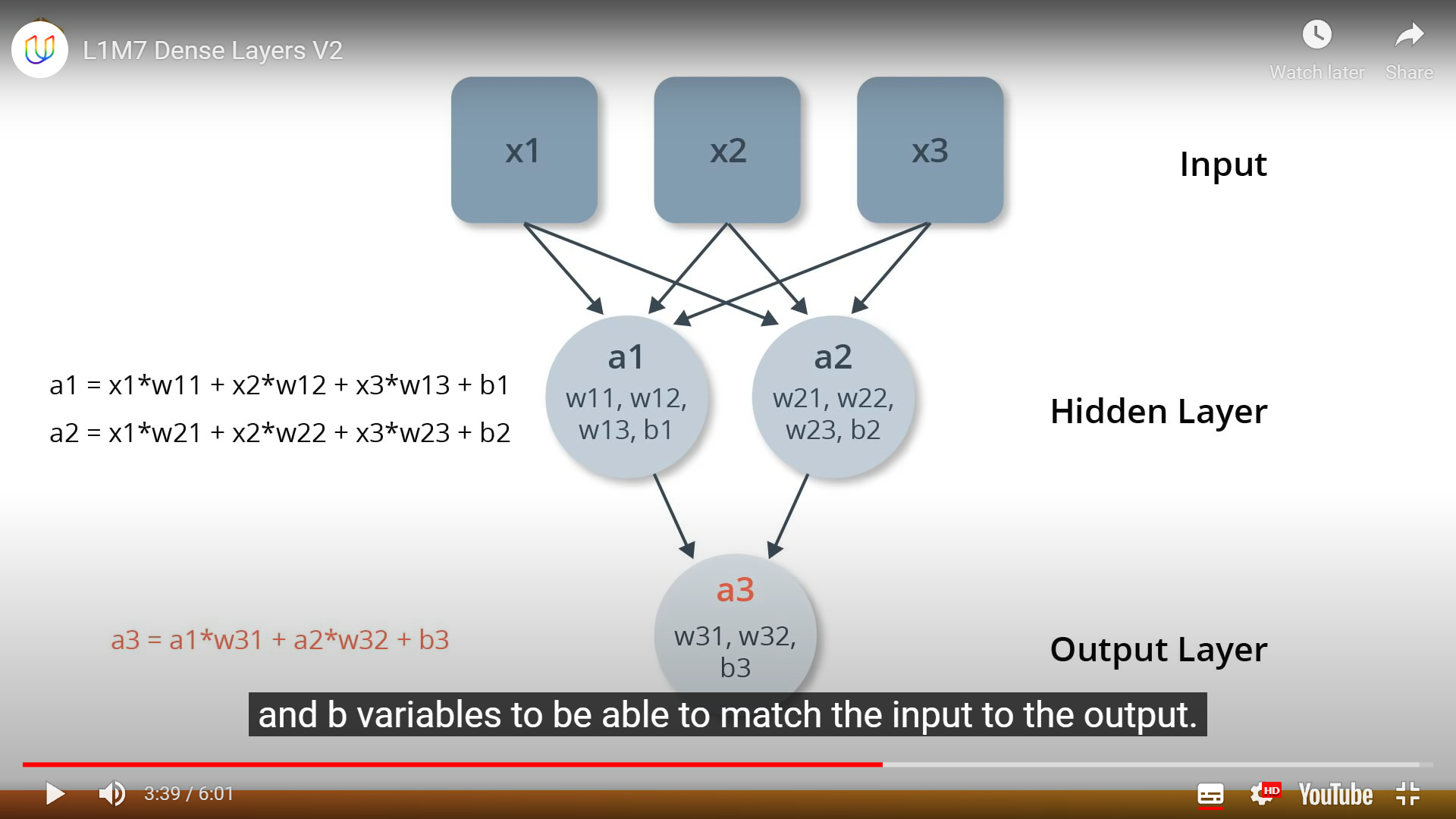
>> dense layer = 해당 레이어의 모든 유닛은 그 이전 레이어의 모든 유닛과 연결되어 있을 때!



이렇게 쓰면 됨!

Hidden은 input 각각 3개씩 1차원 배열로 받는 유닛 두 개

Output은 유닛 한 개



각 유닛은 변하지 않는 공식을 갖고 있음

A1의 값 = input1 \* weight1 + input2 \* weight2 + input3 \* weight3 + bias1

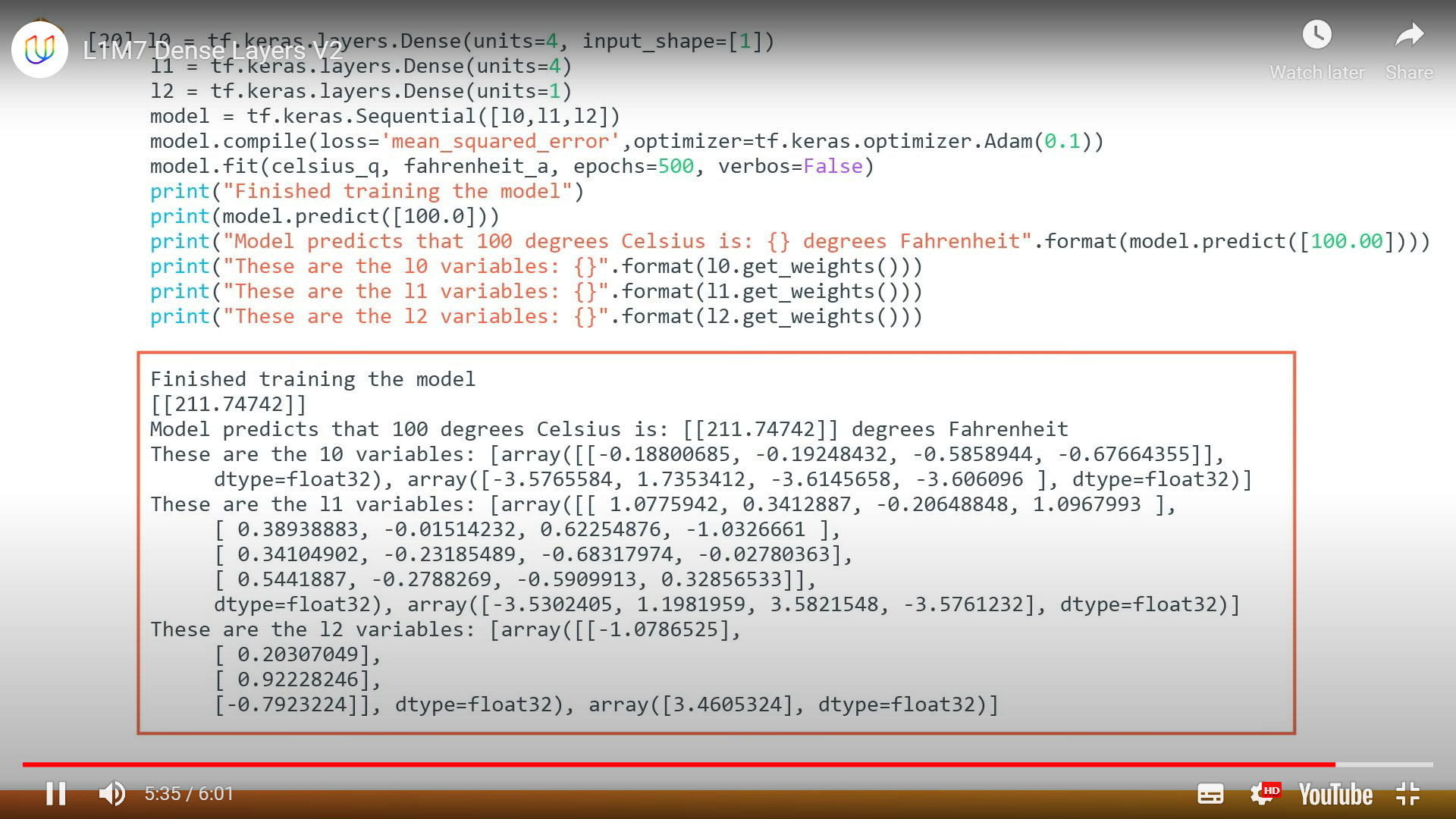
이렇게 계산하고, 결과값이 나온 후 loss – optimizer – w와 b의 값을 바꿈!



Weight 1개, bias 1개

>> 선형 관계를 도출함!

실제로는, 이렇게 정확히 1.8, 32 맞추는 건 불가능함(애초에 모름)

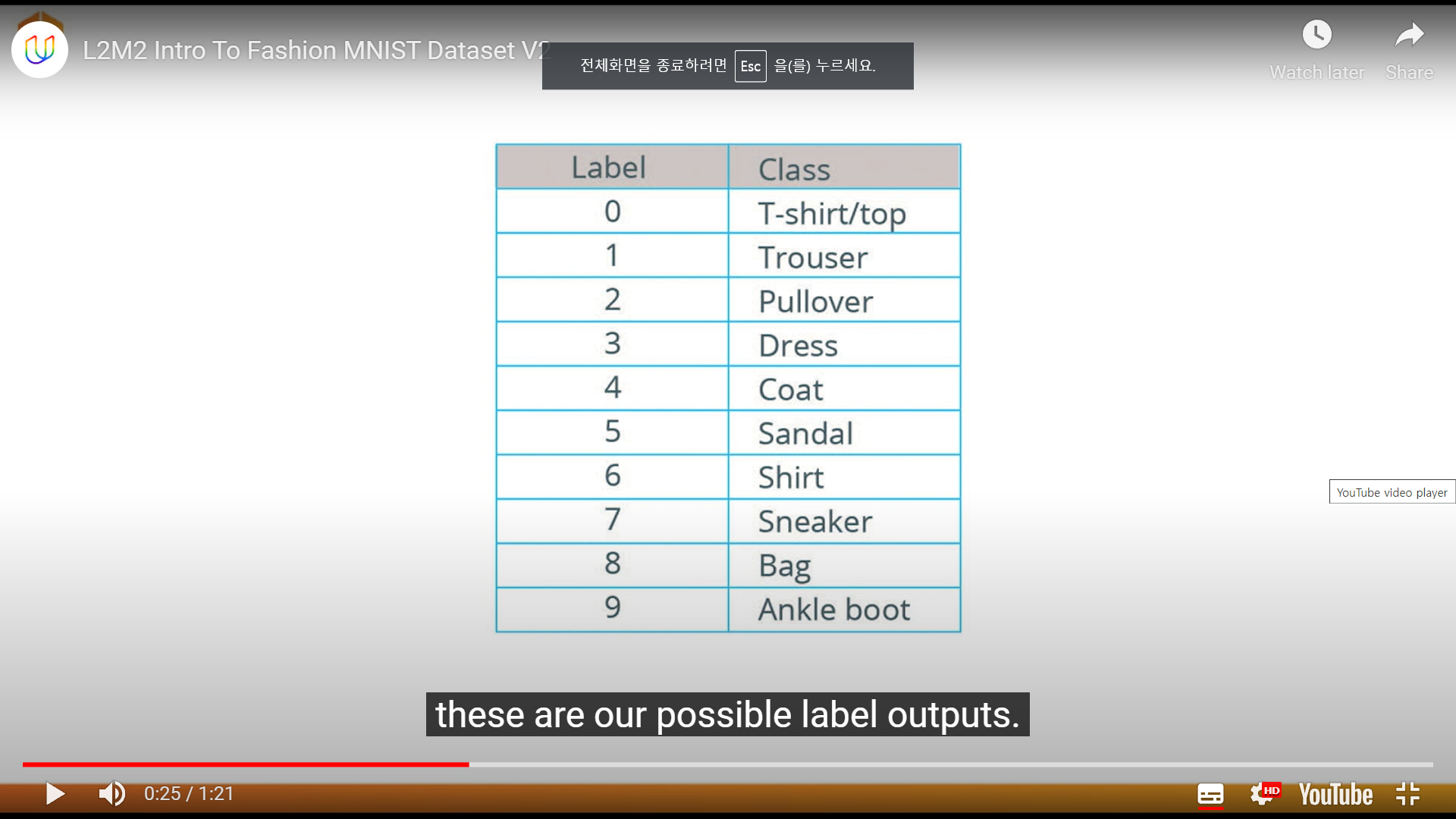


Input 1개, unit 4개, unit 4개, output unit 1개로 구성된 위 모델도 정확히 값 도출했지만,

1.8, 32 값은 어디에도 없음

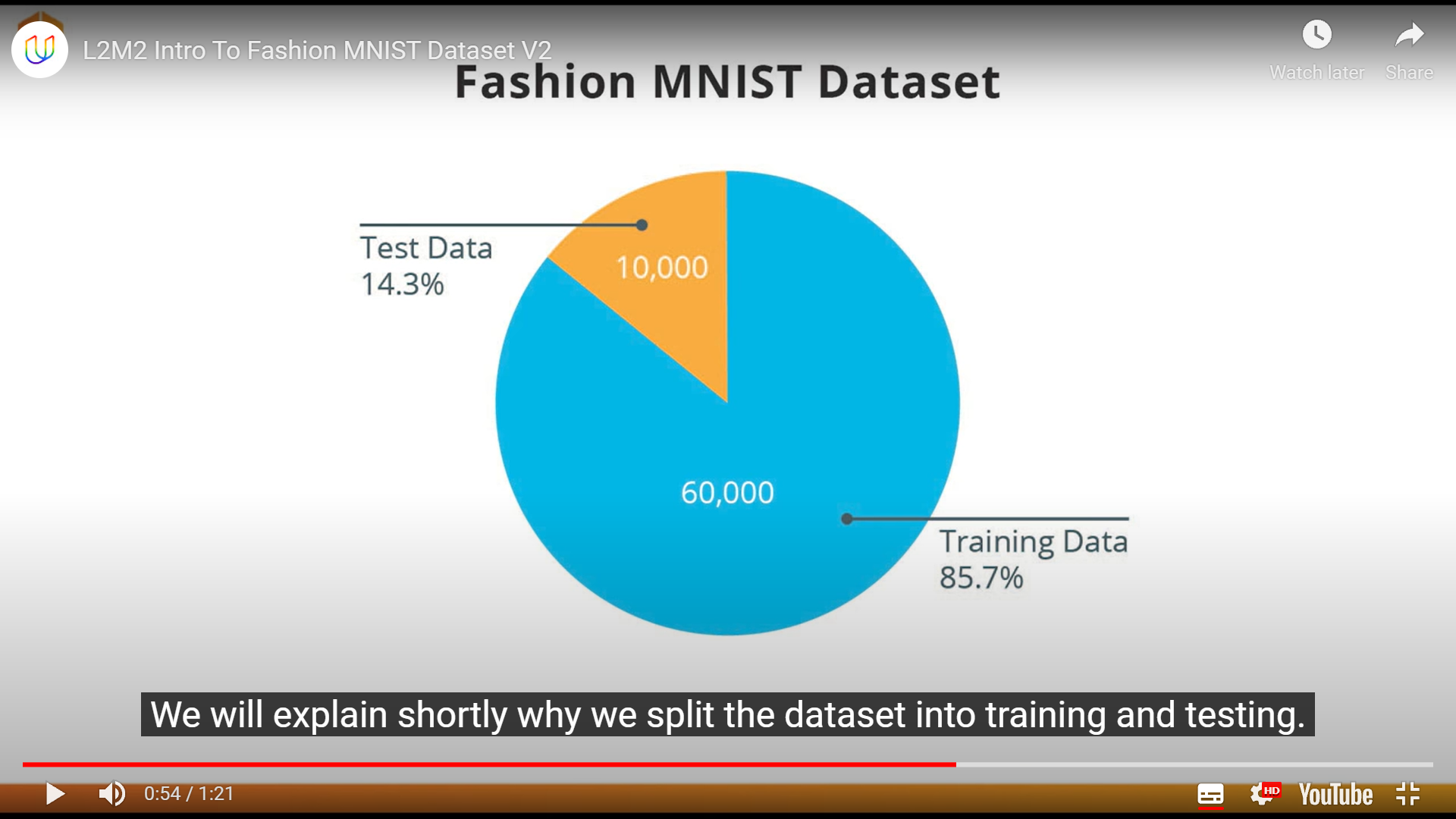
>> 레이어 개수, 유닛 개수를 그냥 계속 바꿔가면서 정확히 맞추는지 확인하는 과정을 거침

강의 3 – Fashion MNIST dataset



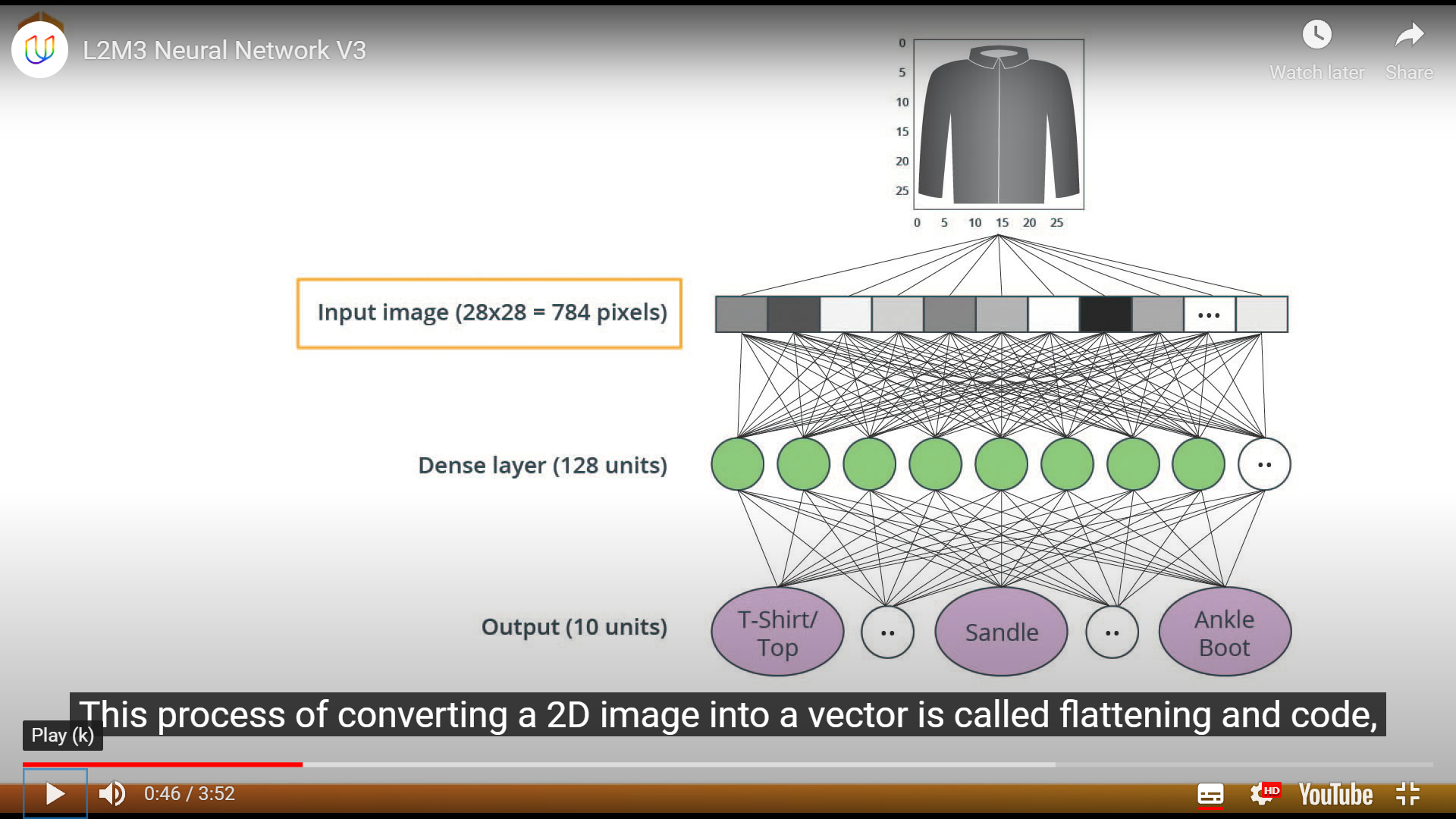
Feature = 28\*28픽셀, 그레이스케일 이미지 = 784바이트

Label = 위 사진에 있는 결과



Training data = 60000장

Testing data = 10000장(잘 예측하는지 테스트할 때 쓰는 거)



우리의 모델

Input image = 784개의 item을 가진 one-dimensional array로 변환해 들어감.

>> Flattening : 2D image (2D array 28 \* 28) -> 1D array of 784 (1D vector)

Dense layer 1개, 128units

>> ReLu : 우리 Dense layer에 넣는 함수; 분류를 도와줌

\*\* Activation function; 우리 모델이 비선형 관계도 찾을 수 있도록 도와줌

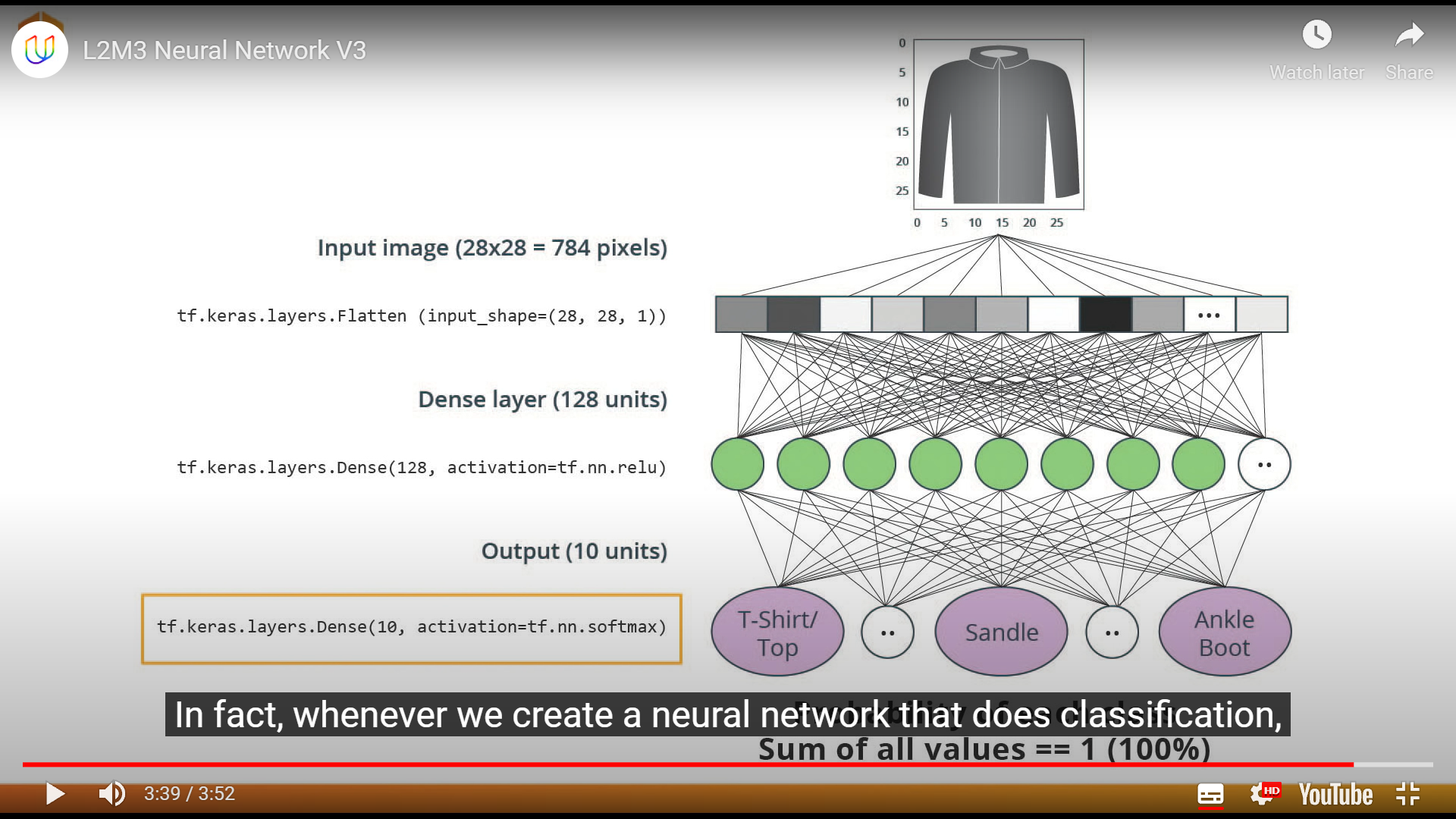
Output layer 1개, 10 units >> 10개의 label이 있기 때문!

>> 10개 label 각각이 맞을 확률을 반환함 (confidence that this input is corresponding label)



합 = 1(확률)

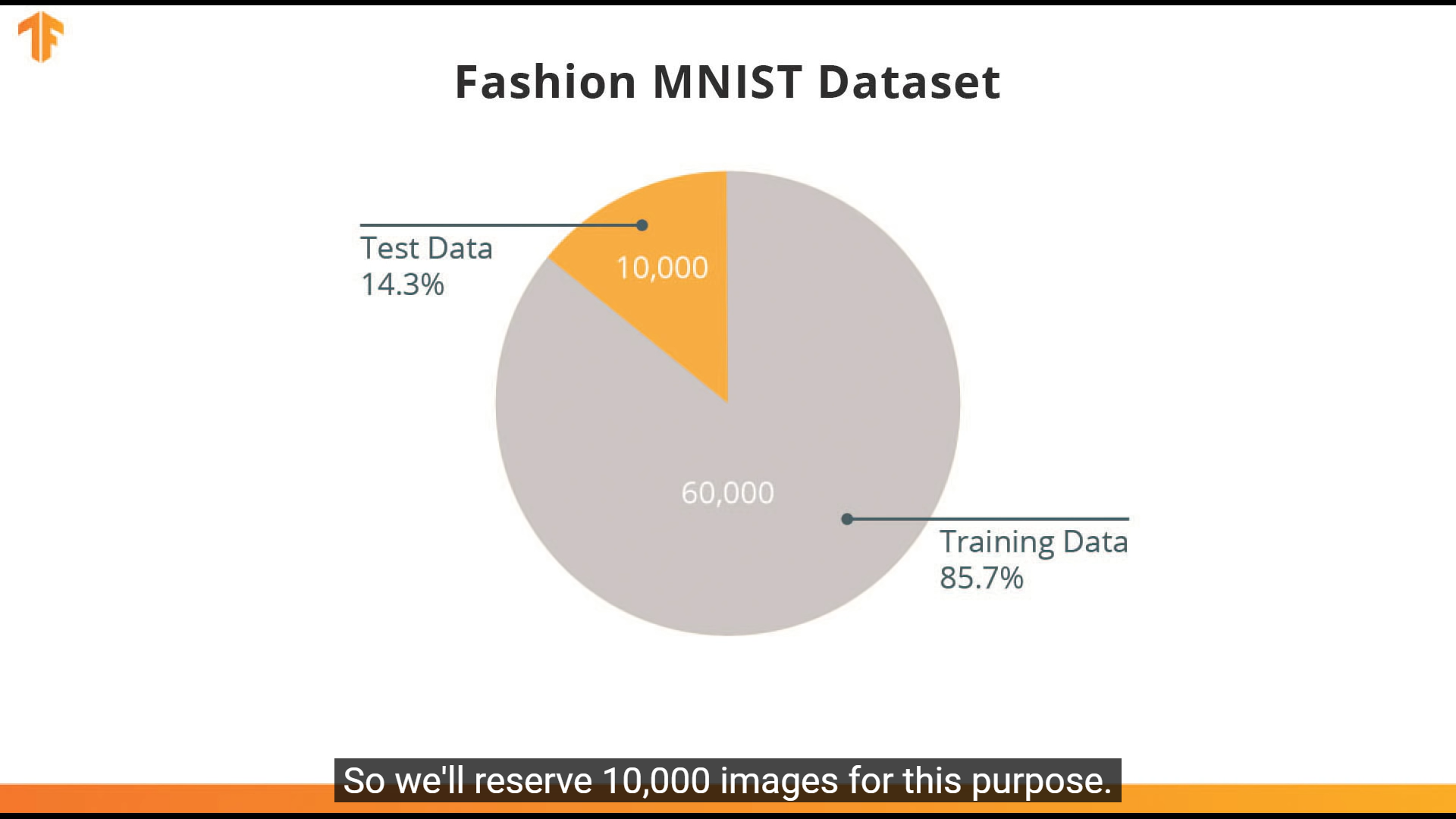
(Distribution of probability()



Output layer(이것도 Dense layer에 해당)

>> softmax 함수(activation function의 일종)를 이용해 각 labe에 대한 확률 반환하게 해줌

\*\*Classification model: 항상 우리가 분류하고자 하는 label 개수에 맞는 Dense output layer를 쓰고, 항상 softmax함수를 사용함



모델 훈련할 때는, 데이터를 적어도 두 개로 분류해야 함

>> Training set, Test set

: Training set으로 훈련시키고, Test set으로 잘 분류/예측하는지 확인

\*\*Validation set

>> 훈련 도중에 모델에게 투입하는 데이터! 훈련이 잘 진행되고 있는지 확인하는 역할

>> loss는 줄어들고 있는데, validation set을 잘 예측하지 못한다면 잘못 훈련되고 있거나, 또는 반복하는 과정에서 training set만을 외우고 있다는 뜻.

Classifying Model: 학습 과정

1 Divide dataset into Train, Test, (validation).

2 Normalize the datasets; every item in the dataset should have a value between 0 and 1

3 Create a model;

Requires ‘Flatten’ layer first to transform 2D image -> 1D array

Requires output dense layer which has the same number of units as output labels.

//

사용된 Activation Function

ReLu : Hidden Dense layer에 사용됨 – 모델이 비선형 관계를 찾을 수 있게 도와줌

Softmax : Output Dense layer에 사용됨 – 입력받은 사진이 신발일 확률, 옷일 확률 등 각 label에 해당하는 확률을 0에서 1 사이로 반환해줌

4 Set number of units, layers, and epochs

Unit, layer, epoch개수가 늘어나면 (주로) 정확도도 늘어났었음