**Lecture** **2**

Feature: input

Label: output

model = tf.keras.Sequential([tf.keras.lasyers.Dense(units= number of nodes of output, input\_shape=[ number of input values] )])

#l0 = tf.keras.layers.Dense(units= number of nodes of output, input\_shape=[ number of input values])

model = tf.keras.Sequential([l0])로 표현 가능

model.compile(loss= , optimizer=tf.keras.optimizers.Adam( ))

#optimizer안에는 0.001-0.1의 숫자

#optimizer는 최적화, loss는 손실함수

#loss에는 mean\_squared\_error(오차의 제곱의 평균)가 많이 쓰임

model.fit(input value, output value, epochs= , verbose= )

#training process

#epochs는 반복횟수, verbose는 logging이랑 같은거

model.predict([input])

#다음 값 예측

l0.get\_weight()

#weight, bias 구하기

**Lecture 3**

tf.keras.layers.Flatten (input\_shape=(width of feature, height of feature, downgrade 후 dimension))

#flattening 과정: 2D를 1D vector로 변환

#(width of feature, length of feature, downgrade 후 dimension)

tf.keras.layers.Dense( number of units in layer, activation=tf.nn.relu)

#activation function이 relu

tf.keras.layers.Dense(number of units in the label, activation=tf.nn.softmax)

#각 output일 확률을 구하는 함수

model.evaluate(test\_dataset, steps=math.ceil(num\_test\_examples/32))

#test data를 통해 accuracy 평가

Classification vs. Regression

-output: list of numbers that represent probabilities for each class / single number

-example: mnist / Celsius to fahrenheit

-loss function: sparse categorical crossentropy / mean squared error

-last layer activation function: softmax / none

Lecture 4

Filter(kernel)로 처리한 다음 pooling해서 output 생성

pooling에는 maxpooling, meanpooling 등이 있음

주로 overfitting을 막기 위해 사용

Batch size: weight와 bias를 update하는 example의 개수 주기

Lecture 5

Color image는 3D, RGB image는 depth =3

model.add(Conv2D(number of filter, (kernel’s row, kernel’s column), padding=’ ’, Input\_shape=(length, width, depth), activation=’ ‘))

#Conv2D는 주로 사용되는 convolution layer의 종류

#number of filter가 output의 depth 결정

#padding: 경계처리 방법/ ‘valid’: 유효한 영역만 출력🡪output size<input size / ‘same’: input size = output size

#input\_shape: 입력의 형태 정의(dimesion에 따라 값이 변화), 첫 layer에서만 정의

get\_file(‘file name’, origin=\_URL, extract= )

#this function download the dataset from url and unpack on disk

model.summary()

# 사용되는 layer의 종류와 개수, output, parameters 등 코드를 요약해줌

binary classification에서 softmax와 함께 sigmoid도 사용됨

#sigmoid를 사용할 때는 model.compile에서 loss 값을 binary\_crossentropy나 sparse\_categorical\_crossentropy로 넣어줘야 함

Image augmentation: 하나의 이미지를 회전, 확대 등으로 변화시킴으로써 제한된 training set에서 model을 더 general하게 만들어 줌

Dropout: 하나의 layer에서 특정한 node의 weight이 해당 layer의 과정을 독식하는 것을 막기 위해 training 중 랜덤하게 node를 turn off 시키는 과정

ImageDataGenerator( )

#horizontal\_flip=True: 뒤집기

#rotation\_range=45: 45도 회전

#zoom\_range=0.5: 확대

Early stopping: overfitting을 막기 위해 validation set의 loss를 확인해 training을 일찍 끝마치는 방법

Lecture 6

Transfer learning: reusing the model that has already been trained on a large datset.

# 마지막 layer를 우리가 원하는 class의 수에 맞게 바꿔야 함

Lecture 8

Moving average: trend와 season을 제거한 후 noise부터 잡고 적용

(dataset = tf.data.Dataset.range(10) 선언)

dataset.window(window size, shift=shift size, drop\_remainder=True or False)

# drop\_remainder가 True이면 windowsize에 맞는 dataset만 남기고 나머지는 없앰

dataset.flat\_map(lambda window: window.batch(batch size))

# batch size만큼의 tensor들을 set으로 묶기

dataset.map(lambda window: (window[:-1], window[-1:]))

# 크기가 5인 set을 4와 1 크기의 tuple로 나눔

dataset.batch(batch size).prefetch(number)

# 한 번에 돌리 batch size

# 앞의 batch가 돌아가는 동안 준비시킬 set의 개수

RNN

# 여러 개의 node를 하나의 later처럼 이용

# batch size, time, dimensionality per time step의 세가지 input이 들어가 3D input 요구

Lecture 9

NLP

Tokenization

-tokenizing input text

-create and padding sequences

-incorporating out of vocabulary words

-generalizing tokenization and sequence methods to real world datasets

Embedding

-transformed tokenized sequences into embeddings

-Developed a basic sentiment analysis model

-visualized the embeddings vector

-tweaked hyperparameters of the model to improve it

-and diagnosed potential issues with using pre-trained subword tokenizers when the network doesn’t have sequence contex

-size of tokenized vocabulary, number of embedding dimension to use, input length가 input

-output은 flatten or globalaveragepoolin1D layer=>dense나 fully connected layer가 아님

Lecture 10

Recurrent Neural Networks

LSTM

-hidden state을 통해 불필요한 정보를 지우고 기억해야 할 것을 정한다

Text Generation