Ambient AI Bootcamp:
Practice 1

SNU Graduate School of Data Science

Contents

- Tensorflow 이용하여 간단한 모델 만들어보기 실습 (Sequential API)
- Tensorflow Functional API
- Tensorflow Data API 실습

Tensorflow 활용한 간단한 모델 학습

Tensorflow 소개

- Python 기반으로 만들어진 Machine Learning / Deep Learning 개발 라이브러리
- Google Brain 팀에 의해 개발/유지/관리되고 있음.
- Keras: Tensorflow API 중 하나로, Deep learning model을 간단히 개발할 수 있게 도와준다.
- TF version 1과 version 2는 상당히 다름! 우리는 TF 2 기준으로 진행할 예정.
- 자매품: PyTorch(Meta)

모델 학습 Workflow

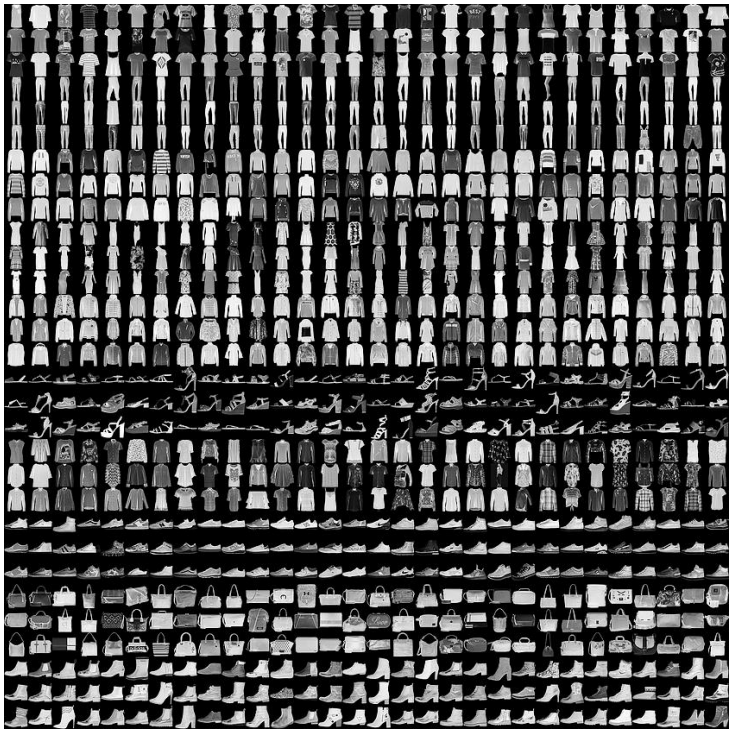
- 간단한 모델이든 복잡한 모델이든 학습과정은 아래의 기본 흐름에서 크게 벗어나지 않음.
- 우리는 Toy dataset과 간단한 모델로 아래의 Workflow를 처음부터 끝까지 실습해 볼 것임.

Data 소개: Fashion MNIST

- 이미지 분류를 위한 Dataset으로 70,000장의 28x28 흑백 이미지가 제공되며 총 10개의 클래스로 구분된다.

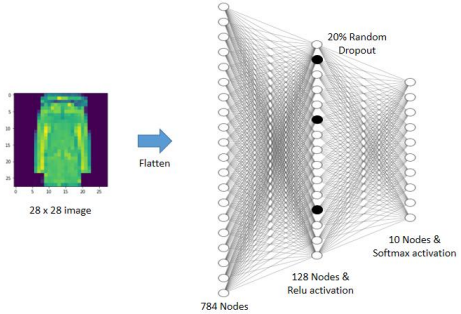
  (T-shirt, Trouser, Dress,...)

- 60,000장은 Training, 10,000장은 Test용으로 사용한다.



Model: Multi Layer Perceptron (MLP)

- 가장 간단한 Neural Network인 Multi Layer Perceptron (MLP) 를 이용하여 모델을 구성해 볼 것이다.
  - Fully Connected (FC), Dense(Keras). Linear(PyTorch) 라고도 한다.



- Activation: Non-linearity를 추가해주기 위해 Layer를 통과한 후 계산을 추가해준다.
  - ReLU(Rectified Linear Unit): $f(x) = x^+ = \max(0, x)$
  - Softmax: $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$
- Dropout: Overfitting을 막기위해 Training시 일부 Node를 임의로 제거한다.
  - Inference를 할 때에는 비활성화된다.

Loss and Optimizer

- Cross Entropy Loss: 분류 문제에 가장 많이 사용되는 Loss이다.
  - $H(P^*|P) = -\sum_i \underbrace{P^*(i)}_{\text{True class distribution}} \underbrace{logP(i)}_{\text{Predicted class distribution}}$

- Adam optimizer: Adaptive Moment를 활용한 Optimizer
  - 기본적으로 Stochastic gradient 방식이지만, update의 크기를 파라미터마다 적절히 가중평균하여 적용한다.

Training and Analysis

- Training이 잘 되는지 확인하기 위해 Loss, Accuracy 등을 확인한다.
  - Training epoch이 증가함에 따라 Loss는 감소해야 하고, Accuracy는 증가해야 한다.
  - 다만, 일시적인 등락은 발생할 수 있다.
- Overfitting이 발생하는지 확인하기 위해 Validation loss, accuracy를 같이 확인하면 좋다.

```
Epoch 1/10
1500/1500 [==============================] - 11s 7ms/step - loss: 9.3376 - accuracy: 0.5425 - val_loss: 0.9771 - val_accuracy: 0.6796
Epoch 2/10
1500/1500 [==============================] - 7s 5ms/step - loss: 1.0013 - accuracy: 0.6276 - val_loss: 0.7594 - val_accuracy: 0.7190
Epoch 3/10
1500/1500 [==============================] - 10s 7ms/step - loss: 0.8553 - accuracy: 0.6631 - val_loss: 0.6760 - val_accuracy: 0.7212
Epoch 4/10
1500/1500 [==============================] - 7s 5ms/step - loss: 0.7747 - accuracy: 0.6850 - val_loss: 0.7358 - val_accuracy: 0.7204
Epoch 5/10
1500/1500 [==============================] - 8s 5ms/step - loss: 0.7775 - accuracy: 0.6903 - val_loss: 0.6752 - val_accuracy: 0.7430:
0s - loss: 0.7775 - accuracy: 0.69
Epoch 6/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.7813 - accuracy: 0.6913 - val_loss: 0.6408 - val_accuracy: 0.7812
Epoch 7/10
1500/1500 [==============================] - 5s 3ms/step - loss: 0.7552 - accuracy: 0.7174 - val_loss: 0.6987 - val_accuracy: 0.7602
Epoch 8/10
1500/1500 [==============================] - 5s 4ms/step - loss: 0.7606 - accuracy: 0.7328 - val_loss: 0.8769 - val_accuracy: 0.7753
Epoch 9/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.7131 - accuracy: 0.7539 - val_loss: 0.5809 - val_accuracy: 0.8248
Epoch 10/10
1500/1500 [==============================] - 4s 3ms/step - loss: 0.7173 - accuracy: 0.7500 - val_loss: 0.6042 - val_accuracy: 0.8152
```

Loss는 감소하고,

Accuracy는 증가한다.

Validation loss, accuracy 모두 Training loss, accurac와 크게 차이나지 않는다.

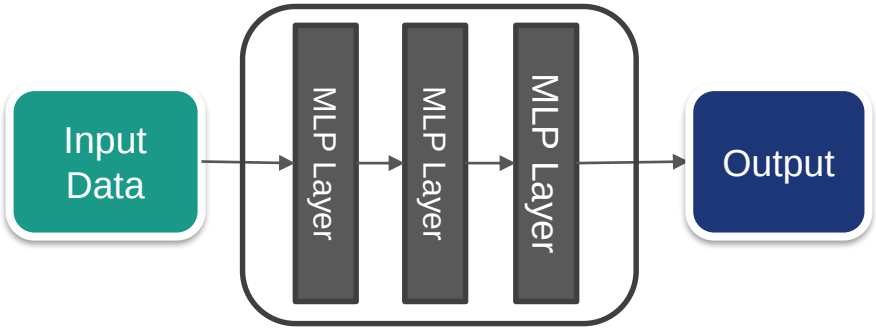
Tensorflow functional APIs

Sequential vs Functional API

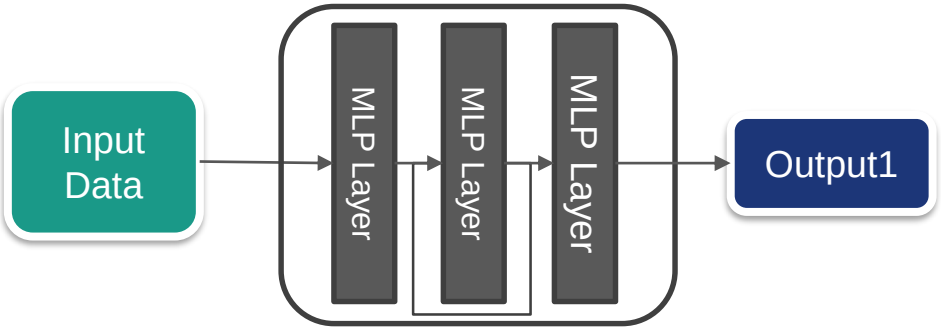
- Tensorflow에서 모델을 구성하는 여러 방법이 있음
- Sequential API: Layer를 하나씩 쌓아나가는 개념으로 이해하면 됨. 정확히 하나의 입력 텐서와 하나의 출력 텐서가 있는 경우에 적합
- Functional API: 자유도가 높아 모델의 구조가 복잡한 경우에 Sequential API대신 사용할 수 있음.

Sequential Model: 1개의 Input 1개의 Output, Layer가 순차적으로 쌓이는 모델

Model = Seqeuntial([Layer1, Layer2, Layer3])

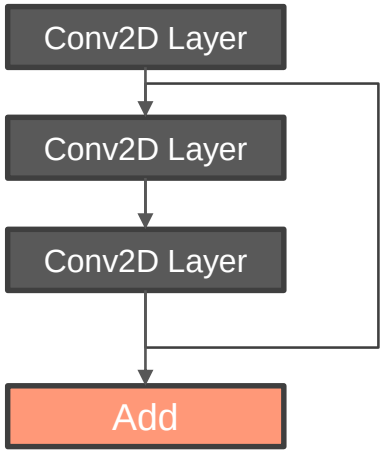


Functional Model: Layer가 순차적으로 쌓이지 않거나, 여러개의 input 또는 output이 있는 경우
x1 = Layer1(x); x2 = Layer2(x1); x3 = x1 + x2
Out = Layer3(x3)
functional_model = Model(Input=x, Output=Out)

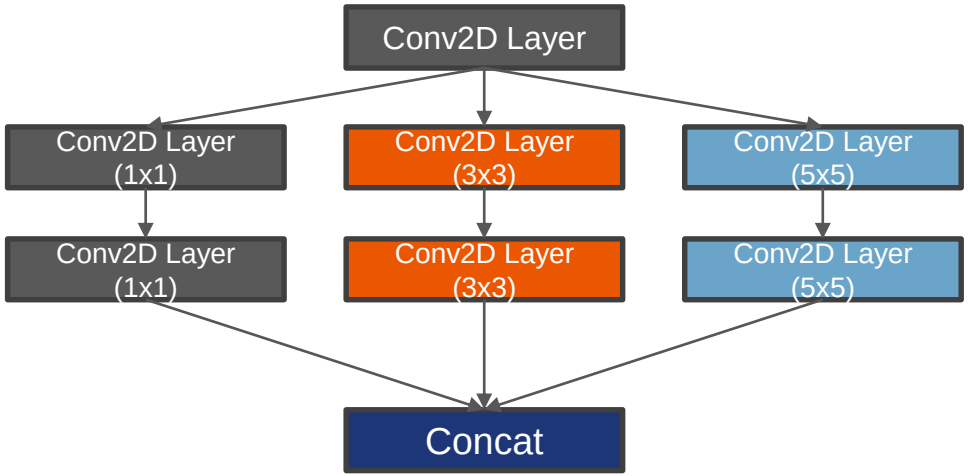


Functional API 활용할 수 있는 모델 구조

- 실습에서 ResNet의 Skip Connection, Inception network의 Inception module 등을 구현할 수 있다



ResNet skip connection



Inception module

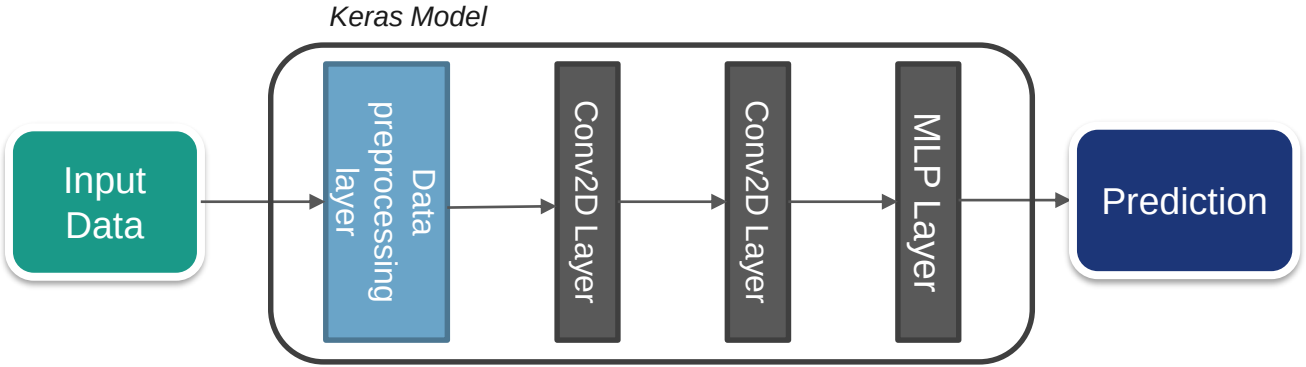
Tensorflow Data API

Importance of data preprocessing

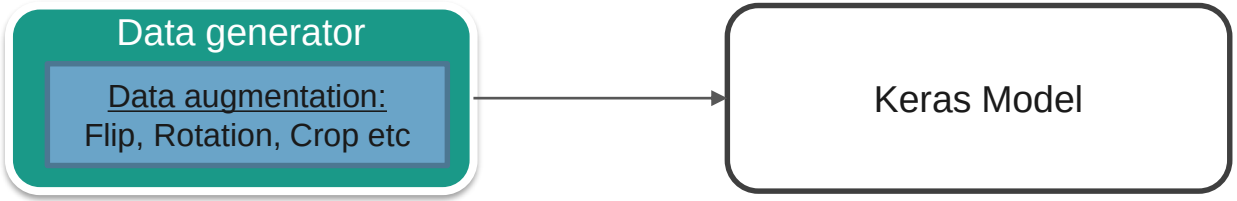
- 좋은 모델 못지 않게 Data preprocessing이 중요함.
- Toy dataset에서 벗어나 대규모 데이터를 활용하거나, 실제 환경에서 수집된 데이터를 활용해야 하는 경우 더욱 중요성이 부각됨.
- Data preprocessing을 효율적으로 수행하기 위해 Tensorflow에서 여러 API를 제공하고 있음.
- 꼭 Tensorflow가 아니더라도, OpenCV, Numpy, Pillow, Scipy 등의 라이브러리를 활용 가능.

Keras data API

- Keras datasets: 자주 사용하는 데이터셋들을 편리한 형태로 제공
- Data preprocessing/augmentation using Keras layers: Data preprocessing 과정을 모델의 레이어와 비슷한 형태로 활용할 수 있도록 제공



- Keras ImageDataGenerator: Data generator안에 data augmentation을 손쉽게 추가할 수 있다.



Tf.data API

- Keras에서 제공하는 기능에 비해 더 최적화된 Pipeline을 구성할 수 있어, 대용량 데이터를 다룰 때 활용하면 좋다. 다만 최적화에 초점이 맞춰져 있어 사용성이 조금 떨어지는 단점이 있다.
- `tf.data.Datasets`: tf.data API의 기본이 되는 class이다.
  - `Tensorflow_datasets`: Tensorflow에서 제공하는 데이터셋 모듈(`keras.datasets`와 비슷하다고 보면 된다.)
- `tf.data.Datasets.map`: Tensor 연산을 적용하기 위해 활용
- `tf.py_function`: 일반 Python function을 이용하여 Data preprocessing을 하고자 할 때 활용