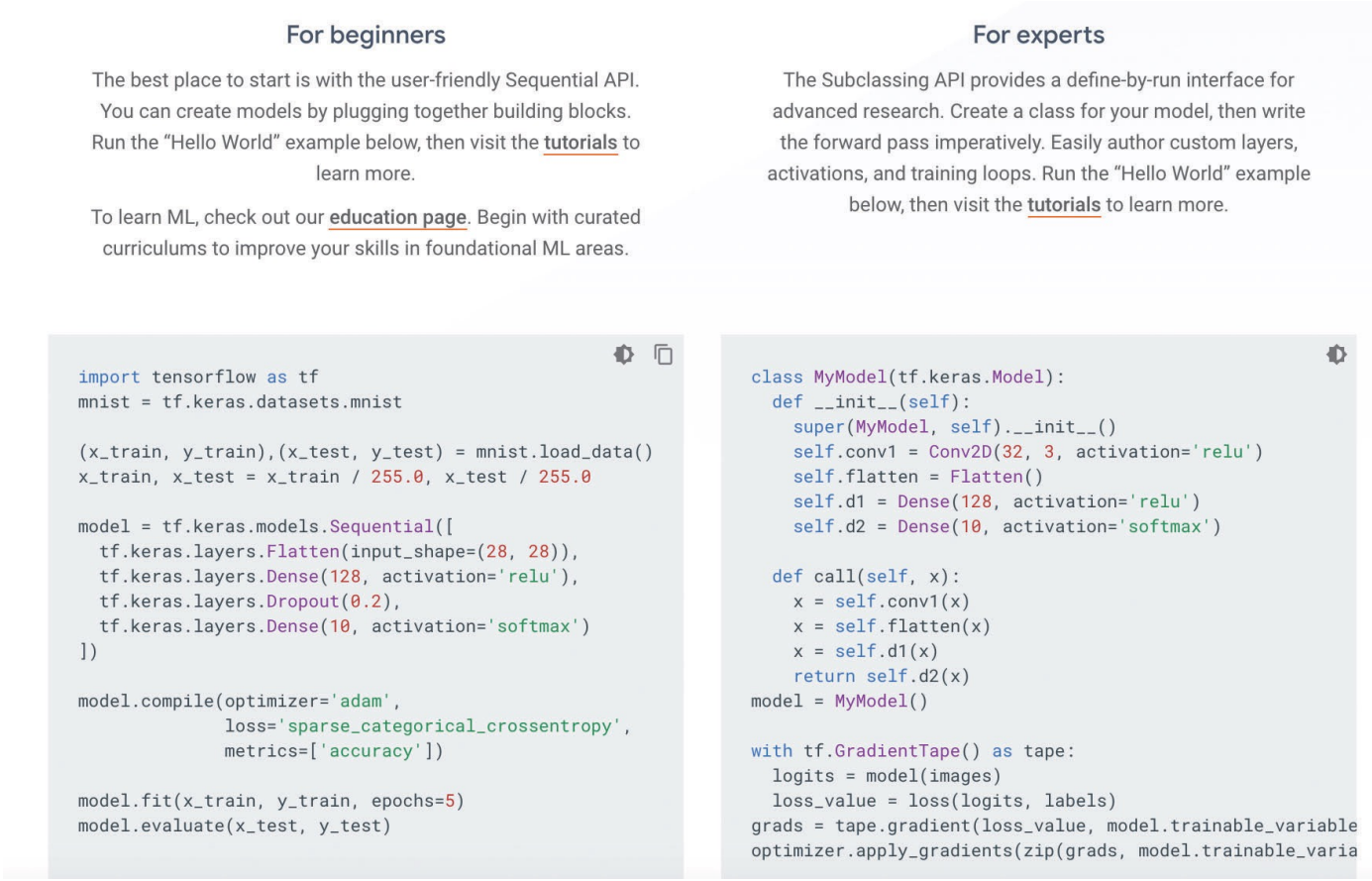
TensorFlow 2.0을 이용한 알고리즘 구현의 2가지 방식 – beginner style, expert style

- https://www.tensorflow.org/overview?hl=ko
- TensorFlow 2.0을 이용해서 딥러닝 알고리즘을 구현하는 방법은 크게 2가지 방식으로 나뉠 수 있습니다.



For beginners

The best place to start is with the user-friendly Sequential API. You can create models by plugging together building blocks. Run the "Hello World" example below, then visit the tutorials to learn more.

To learn ML, check out our education page. Begin with curated curriculums to improve your skills in foundational ML areas.

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

For experts

The Subclassing API provides a define-by-run interface for advanced research. Create a class for your model, then write the forward pass imperatively. Easily author custom layers, activations, and training loops. Run the "Hello World" example below, then visit the tutorials to learn more.

```
class MyModel(tf.keras.Model):
  def __init__(self):
    super(MyModel, self).__init__()
    self.conv1 = Conv2D(32, 3, activation='relu')
    self.flatten = Flatten()
    self.d1 = Dense(128, activation='relu')
    self.d2 = Dense(10, activation='softmax')

  def call(self, x):
    x = self.conv1(x)
    x = self.flatten(x)
    x = self.d1(x)
    return self.d2(x)
model = MyModel()

with tf.GradientTape() as tape:
  logits = model(images)
  loss_value = loss(logits, labels)
grads = tape.gradient(loss_value, model.trainable_variable
optimizer.apply_gradients(zip(grads, model.trainable_varia
```

TensorFlow 2.0을 이용한 알고리즘 구현의 2가지 방식 – beginner style, expert style

- 한 가지 방식은 beginner style로 초심자를 위한 구현 형태입니다. 이는 케라스에서 제 공하는 compile과 fit API를 이용한 하이레벨High-level 방식의 구현으로 손쉽게 딥러닝 알고리즘을 구현할 수 있다는 장점이 있습니다. 하지만 알고리즘의 디테일한 부분을 직접 컨트롤할 수 없어서 자유도가 떨어진다는 단점이 있습니다.
- 또다른 방식은 expert style로 전문가를 위한 구현 형태입니다. 이는 위에 언급한 서브클래싱을 이용해서 모델을 구현하고, 직접 경사하강법으로 파라미터를 갱신하는 코드를 작 성합니다. 일종의 로우레벨Low-level 방식의 구현으로 이렇게 할 경우, 코드를 조금더 많 이 작성해야한다는 단점이 있지만, 알고리즘의 디테일한 부분을 직접 컨트롤 할 수 있다는 장점이 있습니다.
- 두 방식 모두 각각의 장단점이 있기 때문에 필요에 따라 적합한 방식으로 딥러닝 알고리즘을 구현하면 됩니다.
- 본 강의에서는 주로 expert style의 코드구현을 설명드릴 예정입니다.

Keras Sublcassing을 이용한 모델 구현

- 기존 TensorFlow 1.0에서도 케라스Keras를 이용하여 하이레벨High-level API를 이용한 형태로 모델을 구현할 수 있었습니다. 케라스를 이용할 경우 더욱 간결한 형태로 모델을 구현할 수 있다는 장점이 있습니다. 따라서 TensorFlow 2.0에서는 되도록 케라스를 사용해서 모델을 구현하는 것을 장려하고 있습니다.

- TensorFlow 2.0에서 추천하는 모델 구현 방법은 아래와 같습니다.

① tf.keras.Model을 상속받는 class를 정의합니다.

② class의 생성자(__init__)에 모델 구조 정의를 위한 연산들(예를 들어, convolution layer, pooling layer, fully connected layer 등)을 tf.keras.layers API를 이용해서 정의합니다.

③ class의 호출부(call)에서 인자값argument으로 인풋 데이터를 받고, 생성자 부분에서 정의한 연산들을 통해서 모델의 아웃풋을 계산한 다음 반환합니다.

TensorFlow 2.0을 이용한 Softmax Regression 구현

- Softmax Regression 알고리즘을 TensorFlow 2.0 코드로 구현해봅시다.
- https://github.com/solaris33/deep-learning-tensorflow-book-code/blob/master/Ch04-Machine_Learning_Basic/mnist_classification_using_softmax_regression_v2_keras.py

Chapter 3 - 텐서플로우 기초와 텐서보드

- 텐서플로우 기초 – 그래프 생성과 그래프 실행 (Code) (TF v2 Code)
- 플레이스홀더 (Code) (TF v2 Code)
- 선형 회귀(Linear Regression) 알고리즘 (Code) (TF v2 Code)
- 선형 회귀(Linear Regression) 알고리즘 + 텐서보드(TensorBoard) (Code) (TF v2 Code) (TF v2 Keras Code)

Chapter 4 - 머신러닝 기초 이론들

- 소프트맥스 회귀(Softmax Regression)를 이용한 MNIST 숫자분류기 (Code) (TF v2 Code) (TF v2 Keras Code)
- tf.nn.sparse_softmax_cross_entropy_with_logits API를 사용한 소프트맥스 회귀(Softmax Regression)를 이용한 MNIST 숫자분류기 (Code) (TF v2 Code)

Thank you!