بنام خدا

آموزش خط به خط شبکه mlp

نحوه import کردن تنسورفلو

قدم اول در کدنویسی، فراخوانی کتابخانه‌های مورد نیاز است. ابتدا نیاز است که ما تنسورفلو و کراس را در کولب فراخوانی کنیم. برای این کار کافی است بنویسیم:

```
import tensorflow as tf
from tensorflow import keras
```

در خط اول تنسورفلو را فراخوانی کردیم و مشخص کردیم که در کد به جای عبارت tensorflow از مخفف tf استفاده خواهیم کرد. در خط دوم، از کتابخانه تنسورفلو، کراس را هم فراخوانی کردیم. بیایید ورژن این دو کتابخانه را نیز مشخص کنیم. این کار را با دستور .__version__ انجام می‌دهیم:

```
print("tensorflow is version: ", tf.__version__)
print("Keras is version: ", keras.__version__)
```

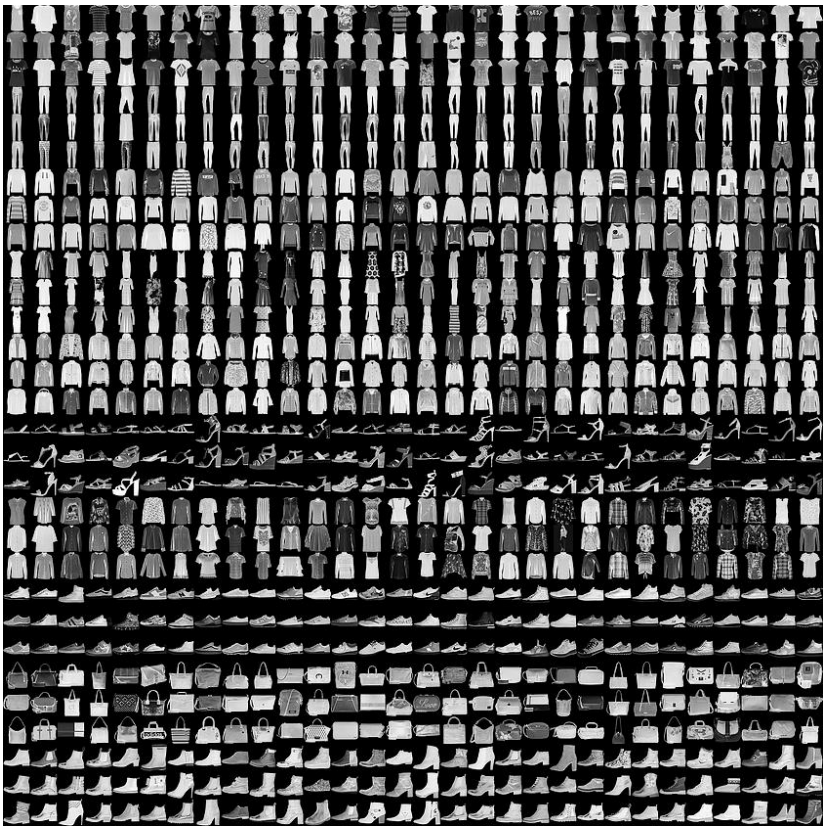
با اجرای کد بالا خواهیم داشت:

```
tensorflow is version: 2.2.0
Keras is version: 2.3.0-tf
```

مشاهده می‌کنید که تنسورفلو نسخه ۲.۲.۰ و کراس نسخه ۲.۳.۰ است. کنار ورژن کراس عبارت tf نوشته شده است. کراس یک فریمورک خالص نیست! یعنی اینکه روی فریمورک‌های دیگری نوشته شده است. چرا؟ چون آن فریمورک‌های دیگر (از جمله تنسورفلو) پیچیده بودند. کراس کدنویسی را ساده‌تر می‌کند و پیچیدگی فریمورک‌هایی مثل تنسورفلو را نیز ندارد. بعد از تغییر اساسی تنسورفلو، کراس و تنسورفلو در هم ادغام شدند به نوعی! واژه tf به معنای این است که کراس روی فریمورک تنسورفلو نوشته شده است.

۱،۱ پایگاه داده fashion mnist

احتمالا همه شما نام دیتاست mnist را شنیده باشید. این دیتاست معروف شامل مجموعه‌ای از اعداد دست‌نویس است که به عنوان بنچمارک در یادگیری ماشین استفاده می‌شود. همه تصاویر در mnist سیاه و سفید بوده و ابعادشان نیز 28×28است. پایگاه داده fashion-mnistپایگاه داده جدیدی است که توسط شرکت Zalando ارائه شده است. این پایگاه داده شامل ۷۰هزار تصویر در حوزه فشن است .همه این تصاویر سیاه و سفید بوده و ابعادشان نیز ۲۸×۲۸ است، دقیقا مانند .Mnist تصاویر نمونه این پایگاه داده در شکل زیر آورده شده است:



این پایگاه داده با هدف جایگزین شدن با mnist ارائه شده است. یک دلیل این است که الگوریتم‌های یادگیری ماشین امروزی (شبکه‌های CNN) توانسته‌اند به دقت 99.7درصد برسند .پایگاه دادهfashion-mnist ، ده کلاس دارد .این کلاس‌ها به همراه برچسب‌های متناظرشان در جدول زیر نشان داده شده‌اند.

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

از ۷۰ هزار تصویری که در پایگاه داده fashion-mnist وجود دارد، 60هزار تصویر برای آموزش مدل و 10هزار تصویر برای تست مدل در نظر گرفته شده است. این سه اصطلاح را در اغلب پایگاه‌های داده می‌شنویم:

معمولا پایگاه داده را به سه بخش تقسیم می کنند . validation dataset، training dataset و test dataset بخش اول آن را فقط برای آموزش شبکه استفاده می کنند .(training dataset) بخش دوم را برای اعتبارسنجی شبکه استفاده می کنند .(validation dataset) اعتبارسنجی یعنی اینکه در حین آموزش شبکه، یک درکی از عملکرد شبکه داشته باشیم .ممکن است یک پایگاه داده validation نداشته باشد .در این صورت خودمان بخشی از داده‌های آموزش را جدا کرده و به عنوان validation استفاده می‌کنیم. مثلا ۸۰ درصد از داده‌های آموزشی برای آموزش شبکه و ۲۰ درصد آن را برای اعتبارسنجی در نظر می‌گیریم. در نهایت بخش سوم پایگاه داده برای ارزیابی شبکه استفاده می‌شود (test dataset).

۱/۲ فراخوانی پایگاه دادهfashion-mnist

برخی فریمورک‌ها توابعی برای خواندن پایگاه داده‌های معروف نوشته‌اند تا کار را برای محقق آسان‌تر کنند .در کراس هم توابعی برای فراخوانی تعدادی از دیتاست های معروف وجود دارد. برای فراخوانی این دیتاست باید بنویسید:

```
fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_data()
```

این دو خط کد، داده‌های fashion mnist را دانلود کرده و فراخوانی می کند. به خط دوم دقت کنید .خروجی دستور fashion_mnist.load_data()دو tuple است. tuple اول مربوط به داده های آموزش و tupleدوم مربوط به داده های تست است .هر tuple هم دو المان در خود دارد. که یکی تصاویر و دومی برچسب آن‌ها است .بنابراین بهتر است همین اول داده‌ها را از هم جدا کنیم. با اجرای کد بالا داده‌ها دانلود و در متغیرهای مخصوص به خودشان ریخته می‌شوند:

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4423680/4422102 [==============================] - 0s 0us/step
```

به همین راحتی توانستیم دیتاست fashion-mnist را دانلود کرده و فراخوانی کنیم! حالا بیایید کمی این داده‌ها را زیر و رو کنیم و ببینیم چه ابعادی دارند و از چه نوعی هستند:

```
print('x_train_full shape: ', x_train_full.shape, 'x_train_full type: ', x_train_full.dtype)
print('y_train_full shape: ', y_train_full.shape, 'y_train_full type: ', y_train_full.dtype)
print('x_test shape: ', x_test.shape, 'x_test type: ', x_test.dtype)
print('y_test shape:', y_test.shape, 'y_test type', y_test.dtype)
```

کد بالا را اجرا می کنیم:

```
x_train_full shape: (60000, 28, 28) x_train_full type: uint8
y_train_full shape: (60000,) y_train_full type: uint8
x_test shape: (10000, 28, 28) x_test type: uint8
```

```
y_test shape: (10000,) y_test type uint8
```

مشاهده می‌کنید همان‌طور که انتظار داشتیم داده‌های آموزش ۶۰هزار نمونه و داده‌های تست ۱۰هزار نمونه بوده و همه آن‌ها از نوع uint8 هستند .سعی کنید حتما بعد از فراخوانی پایگاه داده، ابعاد و نوع آن را چک کنید. تا هرگونه خطای احتمالی در فراخوانی داده ها همین اول کار برطرف شود.

اگر دقت کرده باشیدfashion-mnist ، داده اعتبارسنجی ندارد .پس بیایید یکی بسازیم! تعداد ۱۰هزار تا از نمونه‌های trainرا انتخاب کرده و به اعتبارسنجی اختصاص می‌دهیم .دقت کنید که این ۱۰هزار نمونه باید از داده‌های آموزشی حذف شوند .علاوه بر این تصاویر uint8 هستند. یعنی هر پیکسل در تصاویر، مقداری بین ۰ تا ۲۵۵ دارند. چون از گرادیان کاهشی می‌خواهیم استفاده کنیم، باید این مقادیر را بین ۰ تا ۱ بیاوریم .برای این کار کافی است تصاویر را بر ۲۵۵ تقسیم کنیم.

```
x_valid, x_train = x_train_full[50000:] / 255.0, x_train_full[:50000] / 255.0
y_valid, y_train = y_train_full[50000:], y_train_full[:50000]
```

خب تا اینجا ما توانستیم داده‌ها را فراخوانی کنیم. مقادیر را بین ۰ و ۱ آوردیم. همچنین ۱۰هزارتا از داده‌های آموزش را به اعتبارسنجی اختصاص دادیم.

۱/۳ ساختن شبکه در تنسورفلو

در این بخش می‌خواهیم یک شبکه دسته بندی با mlp و تنسورفلو ۲ را پیاده‌سازی کنیم. شبکه mlp که باید پیاده‌سازی کنیم ساختاری به شکل زیر دارد:

۱) لایه اول: لایه ورودی

۲) لایه دوم: ۳۰۰ نورون تابع فعالسازیrelu

۳) لایه سوم: ۱۰۰ نورون تابع فعالسازیrelu

۴) لایه سوم: لایه خروجی شامل ۱۰ نورون تابع فعالسازیsoftmax

مدل‌های sequential، ساده‌ترین نوع شبکه‌ها هستند که از پشت هم قرار گرفتن چند لایه ساخته می‌شوند. و پیچیدگی اضافه‌ای ندارند. برای تعریف یک مدل sequentialدر کراس از دستور keras.models.Sequential استفاده می‌کنیم. دو راه برای استفاده از این دستور وجود دارد .راه اول این است که ابتدا با کمک keras.models.Sequentialیک مدل خالی تعریف کنیم .سپس لایه‌ها را یکی یکی اضافه کنیم. برای این کار می‌نویسیم:

```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
```

```
model.add(keras.layers.Dense(10, activation="softmax"))
```

چون ورودی یک تصویر ۲۸×۲۸ است، ابتدا باید آن را به یک بردار تبدیل کنیم .به این کار flatکردن گفته می‌شود. این کار در کراس با استفاده از دستور keras.layers.Flatten انجام می‌شود. مشاهده می‌کنید که ورودی این دستور، ابعاد ورودی است که به صورت input_shape=[28, 28] وارد شده است. برای تعریف لایه‌های دیگر از دستور keras.layers.Denseاستفاده می‌کنیم .این دستور یک لایه fully connected یا dense تولید می‌کند .اولین ورودی این دستور تعداد نورون‌ها را نشان می‌دهد .سپس نوع تابع فعالسازی را مشخص می‌کنیم .که در اینجا برای لایه‌های میانی از ReLU و برای لایه خروجی از softmax استفاده کردیم.

روش دوم برای تعریف این شبکه این است که تمام لایه‌ها را داخل همان keras.models.Sequential بنویسیم .به شکل زیر:

```
model = keras.models.Sequential([ keras.layers.Flatten(input_shape=[28, 28]),
keras.layers.Dense(300, activation="relu"),
keras.layers.Dense(100, activation="relu"),
keras.layers.Dense(10, activation="softmax") ])
```

خب ما یک شبکه mlp طراحی کردیم. حالا از کجا بفهمیم که شبکه را لایه به لایه درست طراحی کرده‌ایم یا نه؟ در کراس با کمک دستور model.summary می‌توانیم مدل را لایه به لایه چک کنیم.

```
model.summary()
```

با اجرای کد بالا خواهیم داشت:

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_3 (Dense)              (None, 300)               235500
_________________________________________________________________
dense_4 (Dense)              (None, 100)               30100
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1010
=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```

مشاهده می‌کنید هر لایه از شبکه نشان داده شده است. حتی تعداد پارامترهای هر لایه نیز آورده شده است. در پایان هم تعداد کل پارامترها، پارامترهای قابل آموزش و پارامترهای غیرقابل آموزش هم آورده شده است. می‌دانید منظور از پارامتر چیست دیگر؟ منظور همان وزن‌ها و بایاس‌هایی است که شبکه باید آن‌ها را یاد بگیرد.

بعد از ساختن مدل، نوبت به آموزش آن می‌رسد .برای آموزش مدل، ابتدا باید یکسری چیزها را برای مدل روشن کنیم.

۱) تابع اتلاف چیست

۲) الگوریتم بهینه‌سازی چیست

۳) معیار ارزیابی چیست

در این مسئله ما می‌خواهیم از تابع اتلاف sparse_categorical_crossentropy استفاده کنیم. الگوریتم بهینه‌سازی گرادیان کاهشی باشد. همچنین معیار ارزیابی accuracy یا دقت باشد .برای این کار کافی است بنویسیم:

```
model.compile(loss="sparse_categorical_crossentropy", optimizer="sgd", metrics=["accuracy"])
```

مشاهده می‌کنید که هرکدام از توابع اتلاف، متریک‌ها و بهینه‌سازها نماد مخصوص به خود را دارند. شما می‌توانید این نمادها را در سایت کراس مشاهده کنید

بعد از compile کردن مدل حالا دیگر بالاخره می‌توانیم با دستور model.fit مدلی که تعریف کردیم (model) را آموزش دهیم. برای این کار باید بنویسیم:

```
history = model.fit(X_train, y_train, epochs=30, validation_data=(X_valid, y_valid))
```

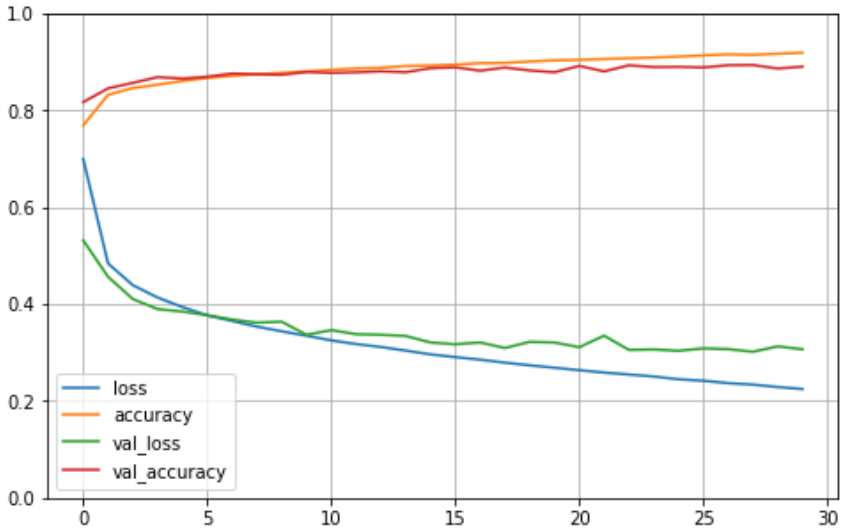
مشاهده می‌کنید که اول داده‌های آموزش و سپس برچسب آن‌ها وارد شده است. در ادامه عبارت epochs = 30 آورده شده است. این عبارت یعنی اینکه تعداد تکرارِ پروسه آموزشِ شبکه، ۳۰ بار باشد .در نهایت هم داده‌های validationرا وارد می‌کنیم. با اجرای کد بالا خواهیم داشت:

```
Epoch 1/30
1719/1719 [==============================] - 5s 3ms/step - loss: 0.6995 - accuracy: 0.7692 - val_loss: 0.5317 -
val_accuracy: 0.8172
Epoch 2/30
1719/1719 [==============================] - 5s 3ms/step - loss: 0.4840 - accuracy: 0.8323 - val_loss: 0.4565 -
val_accuracy: 0.8456
Epoch 3/30
1719/1719 [==============================] - 5s 3ms/step - loss: 0.4392 - accuracy: 0.8461 - val_loss: 0.4106 -
val_accuracy: 0.8570
.
.
.
Epoch 27/30
1719/1719 [==============================] - 5s 3ms/step - loss: 0.2369 - accuracy: 0.9158 - val_loss: 0.3071 -
val_accuracy: 0.8934
Epoch 28/30
1719/1719 [==============================] - 5s 3ms/step - loss: 0.2339 - accuracy: 0.9148 - val_loss: 0.3017 -
val_accuracy: 0.8938
Epoch 29/30
1719/1719 [==============================] - 5s 3ms/step - loss: 0.2289 - accuracy: 0.9171 - val_loss: 0.3128 -
val_accuracy: 0.8866
Epoch 30/30
1719/1719 [==============================] - 5s 3ms/step - loss: 0.2249 - accuracy: 0.9193 - val_loss: 0.3069 -
val_accuracy: 0.8904
```

مشاهده می‌کنید که در هر تکرار مقدار اتلاف و accuracy برای داده‌های آموزش و اعتبارسنجی نوشته شده است. به طور کلی دیده می‌شود که مقدار اتلاف داده‌های اعتبارسنجی و آموزش کم شده و accuracy آن‌ها افزایش یافته است. اما شاید بخواهید این نمودارها را رسم کنید. برای این کار کد زیر را بنویسید:

```
import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1) # set the vertical range to [0-1]
plt.show()
```

با اجرای کد بالا خواهیم داشت:



در نمودارهای بالا به وضوح مشخص است که اتلاف سیر نزولی داشته و مقدار accuracy سیر صعودی دارد. اما یک مرحله مهم از کار مانده است. و آن هم این است که عملکرد شبکه را روی داده‌های تست بررسی کنیم .برای این کار باید بنویسم:

```
model.evaluate(x_test/255, y_test)
```

دقت کنید هر بلایی که سر داده‌های آموزش آوردید، باید روی داده‌های تست هم بیاورید .چون شبکه روی داده‌های آموزش، یادگرفته است. بنابراین اگر داده‌های تست با داده‌های آموزش متفاوت باشد، شبکه نمی‌تواند درست تشخیص بدهد!

مقدار x_test را بر ۲۵۵ تقسیم می‌کنیم تا مقادیر را به بازه ۰ و ۱ بیاوریم .با اجرای کد بالا خواهیم داشت:

```
313/313 [==============================] - 1s 2ms/step - loss: 0.4659 - accuracy: 0.8346
[0.4658662676811218, 0.8345999717712402]
```

عدد اولی که در خروجی می‌بینید، مقدار اتلاف را نشان می‌دهد .عدد دوم accuracy را نشان می‌شود. یعنی ما به دقت ۸۳.۴۵ درصد روی پایگاه داده fashion-mnist رسیدیم! —