Classification

In [ ]:
```
import tensorflow as tf
print(tf.__version__)
```

```
2.12.0
```

In [ ]:
```
from tensorflow import keras as kr
print(kr.__version__)
```

```
2.12.0
```

In [ ]:
```
fashion_mnist = kr.datasets.fashion_mnist
(x_train,y_train), (x_test, y_test) = fashion_mnist.load_data()
```

In [ ]:
```
print(x_train.shape)
```

```
(60000, 28, 28)
```

In [ ]:
```
x_valid, x_train = x_train[:5000]/255, x_train[5000:]/255
y_valid, y_train = y_train[:5000], y_train[5000:]
```

In [ ]:
```
class_names = ["T-shirt/top", "Trouser","pullover","Dress","Coat","Sendal
```

In [ ]:
```
class_names[y_train[6]]
```

Out[ ]:
```
'Coat'
```

In [ ]:
```
model = kr.models.Sequential()
model.add(kr.layers.Flatten(input_shape =[28,28]))
model.add(kr.layers.Dense(300, name= "Hidden1", activation="relu"))
model.add(kr.layers.Dense(100, name= "Hidden2", activation="relu"))
model.add(kr.layers.Dense(10,name= "Output", activation="softmax"))
```

```
Metal device set to: Apple M1

systemMemory: 8.00 GB
maxCacheSize: 2.67 GB
```

In [ ]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 Hidden1 (Dense)             (None, 300)               235500

 Hidden2 (Dense)             (None, 100)               30100

 Output (Dense)              (None, 10)                1010

=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 Hidden1 (Dense)             (None, 300)               235500

 Hidden2 (Dense)             (None, 100)               30100

 Output (Dense)              (None, 10)                1010

=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```

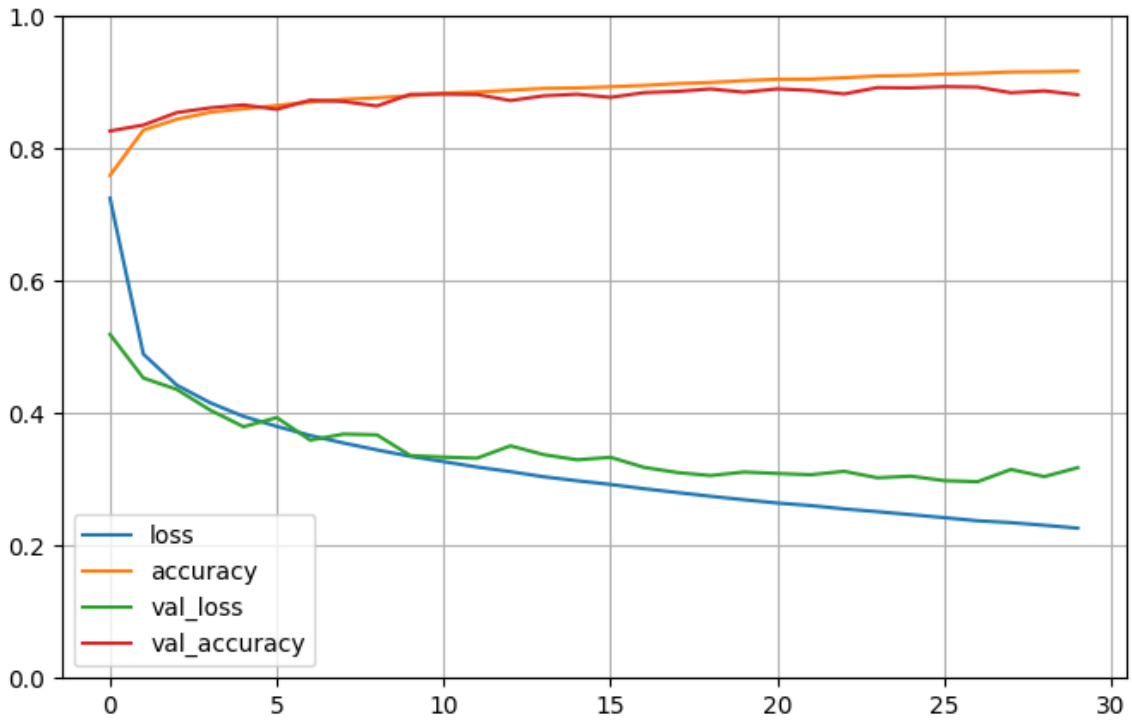
In [ ]:
```
model.compile(loss = "sparse_categorical_crossentropy", optimizer = "sgd"
history = model.fit(x_train,y_train, epochs=30, validation_data = (x_vali
```

```
Epoch 1/30
2023-04-04 14:23:58.879867: W tensorflow/tsl/platform/profile_utils/cpu_
utils.cc:128] Failed to get CPU frequency: 0 Hz
```

```
1719/1719 [==============================] - 10s 6ms/step - loss: 0.7250
- accuracy: 0.7595 - val_loss: 0.5193 - val_accuracy: 0.8266
Epoch 2/30
1719/1719 [==============================] - 9s 5ms/step - loss: 0.4895
- accuracy: 0.8280 - val_loss: 0.4534 - val_accuracy: 0.8356
Epoch 3/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.4426
- accuracy: 0.8443 - val_loss: 0.4361 - val_accuracy: 0.8544
Epoch 4/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.4160
- accuracy: 0.8550 - val_loss: 0.4049 - val_accuracy: 0.8616
Epoch 5/30
1719/1719 [==============================] - 9s 5ms/step - loss: 0.3954
- accuracy: 0.8602 - val_loss: 0.3798 - val_accuracy: 0.8660
Epoch 6/30
1719/1719 [==============================] - 9s 5ms/step - loss: 0.3802
- accuracy: 0.8651 - val_loss: 0.3935 - val_accuracy: 0.8594
Epoch 7/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3665
- accuracy: 0.8705 - val_loss: 0.3594 - val_accuracy: 0.8732
Epoch 8/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3551
- accuracy: 0.8744 - val_loss: 0.3687 - val_accuracy: 0.8712
Epoch 9/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3447
- accuracy: 0.8767 - val_loss: 0.3675 - val_accuracy: 0.8644
Epoch 10/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3350
- accuracy: 0.8798 - val_loss: 0.3362 - val_accuracy: 0.8820
Epoch 11/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3269
- accuracy: 0.8839 - val_loss: 0.3338 - val_accuracy: 0.8824
Epoch 12/30
1719/1719 [==============================] - 10s 6ms/step - loss: 0.3188
- accuracy: 0.8854 - val_loss: 0.3323 - val_accuracy: 0.8818
Epoch 13/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3118
- accuracy: 0.8883 - val_loss: 0.3509 - val_accuracy: 0.8726
Epoch 14/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.3041
- accuracy: 0.8910 - val_loss: 0.3376 - val_accuracy: 0.8798
Epoch 15/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2981
- accuracy: 0.8917 - val_loss: 0.3298 - val_accuracy: 0.8822
Epoch 16/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2925
- accuracy: 0.8934 - val_loss: 0.3335 - val_accuracy: 0.8774
Epoch 17/30
1719/1719 [==============================] - 9s 5ms/step - loss: 0.2861
- accuracy: 0.8955 - val_loss: 0.3184 - val_accuracy: 0.8846
Epoch 18/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2803
- accuracy: 0.8982 - val_loss: 0.3105 - val_accuracy: 0.8864
Epoch 19/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2746
- accuracy: 0.8999 - val_loss: 0.3059 - val_accuracy: 0.8902
Epoch 20/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2694
- accuracy: 0.9027 - val_loss: 0.3113 - val_accuracy: 0.8852
Epoch 21/30
```

```
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2644
- accuracy: 0.9048 - val_loss: 0.3091 - val_accuracy: 0.8902
Epoch 22/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2606
- accuracy: 0.9049 - val_loss: 0.3072 - val_accuracy: 0.8880
Epoch 23/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2555
- accuracy: 0.9071 - val_loss: 0.3124 - val_accuracy: 0.8826
Epoch 24/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2514
- accuracy: 0.9097 - val_loss: 0.3026 - val_accuracy: 0.8922
Epoch 25/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2469
- accuracy: 0.9107 - val_loss: 0.3050 - val_accuracy: 0.8918
Epoch 26/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2423
- accuracy: 0.9126 - val_loss: 0.2983 - val_accuracy: 0.8936
Epoch 27/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2376
- accuracy: 0.9139 - val_loss: 0.2967 - val_accuracy: 0.8930
Epoch 28/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2348
- accuracy: 0.9159 - val_loss: 0.3151 - val_accuracy: 0.8844
Epoch 29/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2308
- accuracy: 0.9162 - val_loss: 0.3044 - val_accuracy: 0.8872
Epoch 30/30
1719/1719 [==============================] - 8s 5ms/step - loss: 0.2265
- accuracy: 0.9171 - val_loss: 0.3179 - val_accuracy: 0.8814
```

In [ ]:
```
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)
plt.show()
```



In [ ]:
```
model.evaluate(x_test, y_test)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 58.6208 -
accuracy: 0.8525
```

Out[ ]:
```
[58.62080383300781, 0.8525000214576721]
```

In [ ]:
```
print(model.predict(x_test[:3]))
print(y_test[3])
```

```
1/1 [==============================] - 0s 55ms/step
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
1
1/1 [==============================] - 0s 55ms/step
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]
1
```

In [ ]:
```
import numpy as np
y_pred = np.argmax(model.predict(x_test[:3]), axis=-1)
y_pred
```

```
1/1 [==============================] - 0s 12ms/step
```

Out[ ]:
```
array([9, 2, 1])
```

In [ ]:
```
np.array(class_names)[y_pred]
```

Out[ ]:
```
array(['Ankle boot', 'pullover', 'Trouser'], dtype='<U11')
```

In [ ]:
```
np.array(class_names)[y_test[:3]]
```

Out[ ]:
```
array(['Ankle boot', 'pullover', 'Trouser'], dtype='<U11')
```

In [ ]:
```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [ ]:
```
housing =  fetch_california_housing()
housing
```

Out[ ]:

```
{'data': array([[   8.3252    ,   41.        ,    6.98412698, ...,    2.
55555556,
           37.88      , -122.23      ],
        [   8.3014    ,   21.        ,    6.23813708, ...,    2.1098418
3,
           37.86      , -122.22      ],
        [   7.2574    ,   52.        ,    8.28813559, ...,    2.8022598
9,
           37.85      , -122.24      ],
        ...,
        [   1.7       ,   17.        ,    5.20554273, ...,    2.3256351
,
           39.43      , -121.22      ],
        [   1.8672    ,   18.        ,    5.32951289, ...,    2.1232091
7,
           39.43      , -121.32      ],
        [   2.3886    ,   16.        ,    5.25471698, ...,    2.6169811
3,
           39.37      , -121.24      ]]),
 'target': array([4.526, 3.585, 3.521, ..., 0.923, 0.847, 0.894]),
 'frame': None,
 'target_names': ['MedHouseVal'],
 'feature_names': ['MedInc',
  'HouseAge',
  'AveRooms',
  'AveBedrms',
  'Population',
  'AveOccup',
  'Latitude',
  'Longitude'],
 'DESCR': '.. _california_housing_dataset:\n\nCalifornia Housing dataset
\n--------------------------\n\n**Data Set Characteristics:**\n\n    :Nu
mber of Instances: 20640\n\n    :Number of Attributes: 8 numeric, predic
tive attributes and the target\n\n    :Attribute Information:\n        -
MedInc        median income in block group\n        - HouseAge      medi
an house age in block group\n        - AveRooms      average number of r
ooms per household\n        - AveBedrms     average number of bedrooms p
er household\n        - Population    block group population\n        -
AveOccup      average number of household members\n        - Latitude
block group latitude\n        - Longitude     block group longitude\n\n
:Missing Attribute Values: None\n\nThis dataset was obtained from the St
atLib repository.\nhttps://www.dcc.fc.up.pt/~ltorgo/Regression/cal_housi
ng.html\n\nThe target variable is the median house value for California
districts,\nexpressed in hundreds of thousands of dollars ($100,000).\n
\nThis dataset was derived from the 1990 U.S. census, using one row per
census\nblock group. A block group is the smallest geographical unit for
which the U.S.\nCensus Bureau publishes sample data (a block group typic
ally has a population\nof 600 to 3,000 people).\n\nA household is a grou
p of people residing within a home. Since the average\nnumber of rooms a
nd bedrooms in this dataset are provided per household, these\ncolumns m
ay take surprisingly large values for block groups with few households\n
and many empty houses, such as vacation resorts.\n\nIt can be downloade
d/loaded using the\n:func:`sklearn.datasets.fetch_california_housing` fu
nction.\n\n.. topic:: References\n\n    - Pace, R. Kelley and Ronald Bar
ry, Sparse Spatial Autoregressions,\n      Statistics and Probability Le
tters, 33 (1997) 291-297\n'}
```


In [ ]:
```
x_train_full, x_test, y_train_full, y_test = train_test_split(housing.dat
x_train, x_valid, y_train, y_valid = train_test_split(x_train_full, y_tra
```

In [ ]:
```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_valid = scaler.transform(x_valid)
x_test = scaler.transform(x_test)
```

In [ ]:
```
print(x_train.shape[1:])
```

```
(8,)
```

In [ ]:
```
model_reg = kr.models.Sequential([
    kr.layers.Dense(30, activation = "relu", input_shape=x_train.shape[1:
    kr.layers.Dense(1)
])
```

In [ ]:
```
opt = kr.optimizers.SGD(0.1, clipnorm=1.)
model_reg.compile(loss=["mse"], loss_weights=[0.9, 0.1], optimizer=opt)
history = model_reg.fit(x_train,y_train, epochs=20, validation_data = (x_
```

```
WARNING:absl:At this time, the v2.11+ optimizer `tf.keras.optimizers.SGD
` runs slowly on M1/M2 Macs, please use the legacy Keras optimizer inste
ad, located at `tf.keras.optimizers.legacy.SGD`.
WARNING:absl:There is a known slowdown when using v2.11+ Keras optimizer
s on M1/M2 Macs. Falling back to the legacy Keras optimizer, i.e., `tf.k
eras.optimizers.legacy.SGD`.
```

```
Epoch 1/20
363/363 [==============================] - 2s 5ms/step - loss: 0.4778 -
val_loss: 1.2084
Epoch 2/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3578 -
val_loss: 0.3476
Epoch 3/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3501 -
val_loss: 0.3867
Epoch 4/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3365 -
val_loss: 0.3255
Epoch 5/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3247 -
val_loss: 0.3042
Epoch 6/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3198 -
val_loss: 0.3543
Epoch 7/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3137 -
val_loss: 1.3766
Epoch 8/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3272 -
val_loss: 0.6769
Epoch 9/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3129 -
val_loss: 0.3528
Epoch 10/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3232 -
val_loss: 0.3652
Epoch 11/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3095 -
val_loss: 0.4045
Epoch 12/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3092 -
val_loss: 0.4414
Epoch 13/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3266 -
val_loss: 1.1040
Epoch 14/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3071 -
val_loss: 0.5528
Epoch 15/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3074 -
val_loss: 1.4967
Epoch 16/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3149 -
val_loss: 1.1309
Epoch 17/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3163 -
val_loss: 1.0139
Epoch 18/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3327 -
val_loss: 0.3613
Epoch 19/20
363/363 [==============================] - 2s 5ms/step - loss: 0.3021 -
val_loss: 0.3599
Epoch 20/20
363/363 [==============================] - 2s 5ms/step - loss: 0.2992 -
val_loss: 0.3787
```

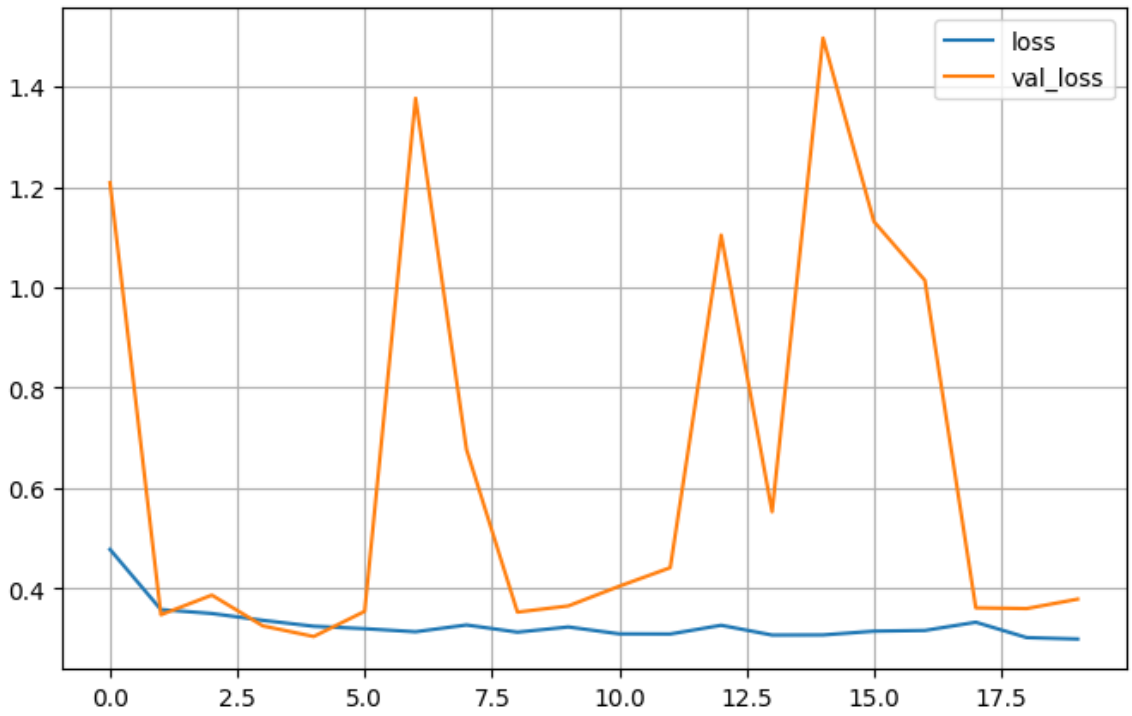
In [ ]:
```
model_reg.evaluate(x_test,y_test)
```

```
162/162 [==============================] - 0s 3ms/step - loss: 0.3178
```

Out[ ]:
```
0.317816823720932
```

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
pd.DataFrame(history.history).plot(figsize=(8,5))
plt.grid(True)

plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: