Learn Keras for Deep Neural Networks

An Introduction to Deep Learning and Keras

A sneak peek in the Keras Framework

```
import os
import logging

import pandas as pd
import tensorflow.keras as keras

from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.utils import plot_model

# Log setting
logging.basicConfig(format="%(asctime)s %(levelname)s %(message)s", datefmt="%H:%M:%S", level

# Change display.max_rows to show all features.
pd.set_option("display.max_rows", 85)


df_train = pd.read_csv(('/content/drive/MyDrive/datasets/train_MachineLearningCVE.csv'), skip
logging.info("Class distribution\n{}".format(df_train.Label.value_counts()))
df_test = pd.read_csv(('/content/drive/MyDrive/datasets/test_MachineLearningCVE.csv'), skipin
logging.info("Class distribution\n{}".format(df_test.Label.value_counts()))


df_train.Label.unique()
df_test.Label.unique()
df = pd.concat([df_train, df_test], axis=0, copy=True)
df.Label.unique()

     array([ 0, 10,  4,  7,  3,  5,  6, 11,  1, 12, 14,  9,  8, 13,  2])


import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

from sklearn.metrics import classification_report
from sklearn.preprocessing import MinMaxScaler

def preprocessing(df: pd.DataFrame) -> (np.ndarray, np.ndarray):
    # Shuffle the dataset
    df = df.sample(frac=1)
```

```
    # Split features and labels
    x = df.iloc[:, df.columns != 'Label']
    y = df[['Label']].to_numpy()

    # Scale the features between 0 ~ 1
    scaler = MinMaxScaler()
    x = scaler.fit_transform(x)

    return x, y

x, y = preprocessing(df)


from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test = train_test_split(x, y, train_size=0.7, random_state=42)


print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)

     (1981520, 78)
     (849223, 78)
     (1981520, 1)
     (849223, 1)


logging.info("Class distribution\n{}".format(df.Label.value_counts()))


#Import required packages
from keras.models import Sequential
from keras.layers import Dense
import numpy as np


# Getting the data ready
# Generate train dummy data for 1000 Students and dummy test for 500
#Columns :Age, Hours of Study & Avg Previous test scores
np.random.seed(2018)
tr_data, te_data = x,y
#Generate dummy results for 1000 students : Whether Passed (1) or Failed (0)
labels = np.random.randint(2, size=(1000, 1))


print(tr_data.shape)
train_data = tr_data.reshape(220797954,1)
train_data = train_data[1:3001]
train_data = train_data.reshape(1000,3)
print(train_data.shape)
print(labels.shape)
```

```
(2830743, 78)
(1000, 3)
(1000, 1)
```

```
print(te_data.shape)
test_data = te_data[1:3001]
test_data = test_data.reshape(1000,3);
print(test_data.shape)
```

```
(2830743, 1)
(1000, 3)
```

```
#Defining the model structure with the required layers, # of neurons, activation function and
model = Sequential()
model.add(Dense(5, input_dim=3, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 5)                 20

 dense_1 (Dense)             (None, 4)                 24

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 49
Trainable params: 49
Non-trainable params: 0
_________________________________________________________________
```

```
#Train the model and make predictions
model.fit(train_data, labels, epochs=10, batch_size=32)
```

```
Epoch 1/10
32/32 [==============================] - 1s 2ms/step - loss: 0.6976 - accuracy: 0.5040
Epoch 2/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6963 - accuracy: 0.5190
Epoch 3/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6954 - accuracy: 0.5120
Epoch 4/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6947 - accuracy: 0.5180
Epoch 5/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6944 - accuracy: 0.4850
Epoch 6/10
```

```
32/32 [==============================] - 0s 2ms/step - loss: 0.6939 - accuracy: 0.5070
Epoch 7/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6933 - accuracy: 0.5150
Epoch 8/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5130
Epoch 9/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6925 - accuracy: 0.5160
Epoch 10/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6921 - accuracy: 0.5160
<keras.callbacks.History at 0x7f2275cb4b90>
```

```
#Make predictions from the trained model
predictions = model.predict(test_data)
```

```
predictions
```

```
array([[0.49773353],
       [0.99556875],
       [0.9887357 ],
       [0.49773353],
       [0.49773353],
       [0.39844453],
       [0.49773353],
       [0.8617834 ],
       [0.8469171 ],
       [0.49773353],
       [0.49773353],
       [0.9887357 ],
       [0.49773353],
       [0.49773353],
       [0.49773353],
       [0.49773353],
       [0.49773353],
       [0.49773353],
       [0.9887357 ],
       [0.96227753],
       [0.8617834 ],
       [0.22142348],
       [0.8617834 ],
       [0.99050933],
       [0.49773353],
       [0.49773353],
       [0.32201588],
       [0.687667  ],
       [0.8617834 ],
       [0.99286425],
       [0.9887357 ],
       [0.49773353],
       [0.9887357 ],
       [0.49773353],
       [0.49773353],
       [0.9887357 ],
       [0.8984884 ],
       [0.14902857],
```

```
            [0.95659083],
            [0.49773353],
            [0.8984884 ],
            [0.49773353],
            [0.14902857],
            [0.49773353],
            [0.47918898],
            [0.49773353],
            [0.687667  ],
            [0.49773353],
            [0.49773353],
            [0.99050933],
            [0.8469171 ],
            [0.8984884 ],
            [0.49773353],
            [0.687667  ],
            [0.39844453],
            [0.8984884 ],
            [0.14902857],
            [0.49773353],
```

```
#Import required packages
from keras.models import Sequential
from keras.layers import Dense
import numpy as np

# Getting the data ready
# Generate train dummy data for 1000 Students and dummy test for 500
#Columns :Age, Hours of Study & Avg Previous test scores
np.random.seed(2018)
# train_data, test_data = np.random.random((1000, 3)), np.random.random((500, 3))
#Generate dummy results for 1000 students : Whether Passed (1) or Failed (0)
# labels = np.random.randint(2, size=(1000, 1))

#Defining the model structure with the required layers, # of neurons, activation function and
model = Sequential()
model.add(Dense(5, input_dim=3, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

#Train the model and make predictions
model.fit(train_data, labels, epochs=10, batch_size=32)

#Make predictions from the trained model
predictions = model.predict(test_data)
```

```
Epoch 1/10
32/32 [==============================] - 1s 2ms/step - loss: 0.6933 - accuracy: 0.4960
Epoch 2/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5020
Epoch 3/10
```

```
32/32 [==============================] - 0s 2ms/step - loss: 0.6931 - accuracy: 0.5020
Epoch 4/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5010
Epoch 5/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6928 - accuracy: 0.5160
Epoch 6/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6927 - accuracy: 0.5100
Epoch 7/10
32/32 [==============================] - 0s 3ms/step - loss: 0.6925 - accuracy: 0.5140
Epoch 8/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6926 - accuracy: 0.5130
Epoch 9/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6924 - accuracy: 0.5090
Epoch 10/10
32/32 [==============================] - 0s 2ms/step - loss: 0.6923 - accuracy: 0.5160
```

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation

np.random.seed(2018)

# Generate dummy training dataset
x_train = np.random.random((6000,10))
x_train = tr_data.reshape(220797954,1)
x_tr = x_train[1:60001]
x_train = x_tr.reshape(6000,10)
y_train = np.random.randint(2, size=(6000, 1))

# Generate dummy validation dataset
x_val = np.random.random((2000,10))
y_val = np.random.randint(2, size=(2000, 1))

# Generate dummy test dataset
x_test = np.random.random((2000,10))
x_t = te_data[1:20001]
x_test = x_t.reshape(2000,10)
y_test = np.random.randint(2, size=(2000, 1))

#Define the model architecture
model = Sequential()
model.add(Dense(64, input_dim=10,activation = "relu")) #Layer 1
model.add(Dense(32,activation = "relu"))               #Layer 2
model.add(Dense(16,activation = "relu"))               #Layer 3
model.add(Dense(8,activation = "relu"))                #Layer 4
model.add(Dense(4,activation = "relu"))                #Layer 5
model.add(Dense(1,activation = "sigmoid"))             #Output Layer

#Configure the model
model.compile(optimizer='Adam',loss='binary_crossentropy',metrics=['accuracy'])
```

```
#Train the model
model.fit(x_train, y_train, batch_size=64, epochs=50, validation_data=(x_val,y_val))
```

```
Epoch 1/50
94/94 [==============================] - 1s 5ms/step - loss: 0.6932 - accuracy: 0.494
Epoch 2/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6931 - accuracy: 0.499
Epoch 3/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6929 - accuracy: 0.502
Epoch 4/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6927 - accuracy: 0.515
Epoch 5/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6926 - accuracy: 0.503
Epoch 6/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6925 - accuracy: 0.516
Epoch 7/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6925 - accuracy: 0.510
Epoch 8/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6923 - accuracy: 0.515
Epoch 9/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6924 - accuracy: 0.512
Epoch 10/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6923 - accuracy: 0.512
Epoch 11/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6921 - accuracy: 0.514
Epoch 12/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6921 - accuracy: 0.513
Epoch 13/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6917 - accuracy: 0.514
Epoch 14/50
94/94 [==============================] - 0s 4ms/step - loss: 0.6921 - accuracy: 0.516
Epoch 15/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6917 - accuracy: 0.520
Epoch 16/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6915 - accuracy: 0.517
Epoch 17/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6912 - accuracy: 0.521
Epoch 18/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6912 - accuracy: 0.514
Epoch 19/50
94/94 [==============================] - 0s 3ms/step - loss: 0.6909 - accuracy: 0.522
Epoch 20/50
94/94 [==============================] - 0s 5ms/step - loss: 0.6910 - accuracy: 0.521
Epoch 21/50
94/94 [==============================] - 0s 5ms/step - loss: 0.6906 - accuracy: 0.524
Epoch 22/50
94/94 [==============================] - 0s 5ms/step - loss: 0.6904 - accuracy: 0.520
Epoch 23/50
94/94 [==============================] - 1s 6ms/step - loss: 0.6895 - accuracy: 0.517
Epoch 24/50
94/94 [==============================] - 0s 5ms/step - loss: 0.6896 - accuracy: 0.525
Epoch 25/50
94/94 [==============================] - 0s 5ms/step - loss: 0.6893 - accuracy: 0.524
Epoch 26/50
94/94 [==============================] - 0s 5ms/step - loss: 0.6886 - accuracy: 0.527
Epoch 27/50
```

```
94/94 [==============================] - 0s 5ms/step - loss: 0.6889 - accuracy: 0.522
Epoch 28/50
94/94 [==============================] - 0s 4ms/step - loss: 0.6884 - accuracy: 0.530
Epoch 29/50
```

```
#Make predictions from the trained model
predictions = model.predict(x_test)
```

```
predictions
```

```
array([[2.3245035e-10],
       [4.7659242e-01],
       [3.8888156e-03],
       ...,
       [5.5876434e-01],
       [4.1089683e-07],
       [4.6738625e-01]], dtype=float32)
```