07.Deep_Learning_Task

December 4, 2021

[ ]:
```
"""
from keras.layers.core import Lambda
import numpy as np
import keras
from keras.models import Sequential,Input,Model
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import BatchNormalization
from keras.layers.advanced_activations import LeakyReLU
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import initializers

from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
"""
```

[4]:
```
from tensorflow.keras import Sequential,Input,Model
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import LeakyReLU
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import initializers
from sklearn.model_selection import train_test_split
import keras
import tensorflow as tf

import matplotlib.pyplot as plt
import numpy as np

!pip install scikeras
from sklearn.model_selection import GridSearchCV
from scikeras.wrappers import KerasClassifier
from sklearn.preprocessing import Normalizer
#from keras.wrappers.scikit_learn import KerasClassifier
```

```
Collecting scikeras
```

```
  Downloading scikeras-0.6.0-py3-none-any.whl (27 kB)
Requirement already satisfied: packaging<22.0,>=0.21 in /usr/local/lib/python3.7
/dist-packages (from scikeras) (21.3)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.7
/dist-packages (from scikeras) (1.0.1)
Collecting importlib-metadata<4,>=3
  Downloading importlib_metadata-3.10.1-py3-none-any.whl (14 kB)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata<4,>=3->scikeras) (3.6.0)
Requirement already satisfied: typing-extensions>=3.6.4 in
/usr/local/lib/python3.7/dist-packages (from importlib-metadata<4,>=3->scikeras)
(3.10.0.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging<22.0,>=0.21->scikeras)
(3.0.6)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=1.0.0->scikeras) (1.1.0)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=1.0.0->scikeras) (1.4.1)
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn>=1.0.0->scikeras) (1.19.5)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7
/dist-packages (from scikit-learn>=1.0.0->scikeras) (3.0.0)
Installing collected packages: importlib-metadata, scikeras
  Attempting uninstall: importlib-metadata
    Found existing installation: importlib-metadata 4.8.2
    Uninstalling importlib-metadata-4.8.2:
      Successfully uninstalled importlib-metadata-4.8.2
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
markdown 3.3.6 requires importlib-metadata>=4.4; python_version < "3.10", but
you have importlib-metadata 3.10.1 which is incompatible.
Successfully installed importlib-metadata-3.10.1 scikeras-0.6.0
```

1 Working Directory

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[7]:
```
%cd /content/drive/MyDrive/American_University/2021_Fall/DATA-642-001_Advanced␣
↪Machine Learning/data/data_ready/np_data
!pwd
```

```
/content/drive/MyDrive/American_University/2021_Fall/DATA-642-001_Advanced
Machine Learning/data/data_ready/np_data
/content/drive/MyDrive/American_University/2021_Fall/DATA-642-001_Advanced
Machine Learning/data/data_ready/np_data
```

[5]:
```
%cd /content/drive/MyDrive/ADVML
!pwd
```

```
[Errno 2] No such file or directory: '/content/drive/MyDrive/ADVML'
/content/drive/MyDrive/American_University/2021_Fall/DATA-642-001_Advanced
Machine Learning/data/data_ready/np_data
/content/drive/MyDrive/American_University/2021_Fall/DATA-642-001_Advanced
Machine Learning/data/data_ready/np_data
```

2 Road the data

[8]:
```
data = np.load("X_data.npy", allow_pickle=True)
```

[9]:
```
print(data.shape)
print(data.dtype)
```

```
(19470, 12588)
float32
```

3 Impute the NAs from Mean

[10]:
```
#print(np.max(data[:, 12288:])
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
imp.fit(data[:, 12288:])
data[:, 12288:] = imp.transform(data[:, 12288:])

print(np.min(data[:, 12288:]))
print(np.max(data[:, 12288:]))
```

```
-0.45410156
0.4814453
```

[11]:
```
fake = np.zeros((9720, 1))
real = np.ones((9750, 1))
label = np.concatenate((fake, real), axis = 0)
print(label.shape)
```

```
(19470, 1)
```

4 Split the Data to 80 % of Train and 20 % of Test

[12]:
```
train_X, valid_X, train_label, valid_label = train_test_split(data, label,␣
↪test_size=0.2, random_state=13)
```

[13]:
```
print(train_X.shape)
print(valid_X.shape)
print(train_label.shape)
print(valid_label.shape)
```

```
(15576, 12588)
(3894, 12588)
(15576, 1)
(3894, 1)
```

5 Grid Search Epochs and Batch Size

[116]:
```
def deepLearning_model():
        # create model
  model = Sequential()
  model.add(Dense(8192, input_dim = len(train_X[1]), activation='relu')) #␣
↪input_dim = one-dimensional flattened arrays,
  model.add(tf.keras.layers.Dropout(0.5))
  model.add(Dense(4096,activation='relu'))
  model.add(tf.keras.layers.Dropout(0.5))
  model.add(Dense(1024))
  model.add(Dense(512, activation='sigmoid'))
  model.add(Dense(256, activation='softmax'))
  model.add(Dense(64))
  model.add(Dense(1, activation='sigmoid'))
        # Compile model
  model.compile(loss='binary_crossentropy',
                      optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                      metrics=['accuracy'])
  return model

np.random.seed(1234)
model = KerasClassifier(build_fn = deepLearning_model, verbose=1)

# define the grid search parameters
batch_size = [64, 128]
epochs = [10, 30]
param_grid = dict(batch_size=batch_size, epochs=epochs)
grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=5)
grid_result = grid.fit(train_X, train_label)
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
    ␣
↪-------------------------------------------------------------------------

      KeyboardInterrupt                         Traceback (most recent call␣
↪last)

      <ipython-input-116-a1d3fc8c7a72> in <module>()
       25 param_grid = dict(batch_size=batch_size, epochs=epochs)
       26 grid = GridSearchCV(estimator=model, param_grid=param_grid,␣
↪n_jobs=-1, cv=5)
  ---> 27 grid_result = grid.fit(train_X, train_label)
       28 print("Best: %f using %s" % (grid_result.best_score_, grid_result.
↪best_params_))


      /usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.
↪py in fit(self, X, y, groups, **fit_params)
      889                 return results
      890
  --> 891             self._run_search(evaluate_candidates)
      892
      893             # multimetric is determined here because in the case of␣
↪a callable


      /usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.
↪py in _run_search(self, evaluate_candidates)
     1390     def _run_search(self, evaluate_candidates):
     1391         """Search all candidates in param_grid"""
  -> 1392         evaluate_candidates(ParameterGrid(self.param_grid))
     1393
     1394


      /usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.
↪py in evaluate_candidates(candidate_params, cv, more_results)
      849                     )
      850                     for (cand_idx, parameters), (split_idx, (train,␣
↪test)) in product(
  --> 851                         enumerate(candidate_params), enumerate(cv.
↪split(X, y, groups))
      852                     )
      853                 )
```

```
      /usr/local/lib/python3.7/dist-packages/joblib/parallel.py in␣
↪__call__(self, iterable)
     1054
     1055             with self._backend.retrieval_context():
  -> 1056                 self.retrieve()
     1057             # Make sure that we get a last message telling us we are␣
↪done
     1058             elapsed_time = time.time() - self._start_time


      /usr/local/lib/python3.7/dist-packages/joblib/parallel.py in␣
↪retrieve(self)
      933             try:
      934                 if getattr(self._backend, 'supports_timeout', False):
  --> 935                     self._output.extend(job.get(timeout=self.
↪timeout))
      936                 else:
      937                     self._output.extend(job.get())


      /usr/local/lib/python3.7/dist-packages/joblib/_parallel_backends.py in␣
↪wrap_future_result(future, timeout)
      540         AsyncResults.get from multiprocessing."""
      541         try:
  --> 542             return future.result(timeout=timeout)
      543         except CfTimeoutError as e:
      544             raise TimeoutError from e


      /usr/lib/python3.7/concurrent/futures/_base.py in result(self, timeout)
      428                 return self.__get_result()
      429
  --> 430             self._condition.wait(timeout)
      431
      432             if self._state in [CANCELLED, CANCELLED_AND_NOTIFIED]:


      /usr/lib/python3.7/threading.py in wait(self, timeout)
      294         try:    # restore state no matter what (e.g.,␣
↪KeyboardInterrupt)
      295             if timeout is None:
  --> 296                 waiter.acquire()
      297                 gotit = True
      298             else:


      KeyboardInterrupt:
```

[ ]:
```
grid_result.best_estimator_
```

[ ]:
```
grid_result.best_params_
```

[14]:
```
np.random.seed(1234)
batch_size = 64
epochs = 50
```

[15]:
```
fashion_model = Sequential()
fashion_model.add(Dense(8192, input_dim = len(train_X[1]), activation='relu'))␣
 ↪# input_dim = one-dimensional flattened arrays,
fashion_model.add(tf.keras.layers.Dropout(0.5))
fashion_model.add(Dense(4096,activation='relu'))
fashion_model.add(tf.keras.layers.Dropout(0.5))
fashion_model.add(Dense(1024))
fashion_model.add(Dense(512, activation='sigmoid'))
fashion_model.add(Dense(256, activation='softmax'))
fashion_model.add(Dense(64))
fashion_model.add(Dense(1, activation='sigmoid'))
```

[16]:
```
fashion_model.compile(loss='binary_crossentropy',
                      optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                      metrics=['accuracy'])
```

[17]:
```
fashion_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8192)              103129088

 dropout (Dropout)           (None, 8192)              0

 dense_1 (Dense)             (None, 4096)              33558528

 dropout_1 (Dropout)         (None, 4096)              0

 dense_2 (Dense)             (None, 1024)              4195328

 dense_3 (Dense)             (None, 512)               524800

 dense_4 (Dense)             (None, 256)               131328

 dense_5 (Dense)             (None, 64)                16448

 dense_6 (Dense)             (None, 1)                 65
```

```
=================================================================
Total params: 141,555,585
Trainable params: 141,555,585
Non-trainable params: 0
_________________________________________________________________
```

[18]:
```
#data = tf.convert_to_tensor(data, dtype=tf.float32)
fashion_train = fashion_model.fit(train_X, train_label,
                                  batch_size=batch_size,epochs=epochs,
                                  verbose=1,validation_data=(valid_X,␣
 ↪valid_label))
```

```
Epoch 1/50
244/244 [==============================] - 386s 2s/step - loss: 0.6775 -
accuracy: 0.5928 - val_loss: 0.6475 - val_accuracy: 0.6685
Epoch 2/50
244/244 [==============================] - 376s 2s/step - loss: 0.6435 -
accuracy: 0.6558 - val_loss: 0.6287 - val_accuracy: 0.6723
Epoch 3/50
244/244 [==============================] - 376s 2s/step - loss: 0.6160 -
accuracy: 0.6846 - val_loss: 0.5779 - val_accuracy: 0.7319
Epoch 4/50
244/244 [==============================] - 376s 2s/step - loss: 0.5875 -
accuracy: 0.7073 - val_loss: 0.5655 - val_accuracy: 0.7275
Epoch 5/50
244/244 [==============================] - 375s 2s/step - loss: 0.5744 -
accuracy: 0.7163 - val_loss: 0.5357 - val_accuracy: 0.7524
Epoch 6/50
244/244 [==============================] - 374s 2s/step - loss: 0.5567 -
accuracy: 0.7300 - val_loss: 0.5654 - val_accuracy: 0.7083
Epoch 7/50
244/244 [==============================] - 377s 2s/step - loss: 0.5323 -
accuracy: 0.7476 - val_loss: 0.4994 - val_accuracy: 0.7763
Epoch 8/50
244/244 [==============================] - 391s 2s/step - loss: 0.5185 -
accuracy: 0.7590 - val_loss: 0.4858 - val_accuracy: 0.7822
Epoch 9/50
244/244 [==============================] - 391s 2s/step - loss: 0.5150 -
accuracy: 0.7576 - val_loss: 0.5137 - val_accuracy: 0.7712
Epoch 10/50
244/244 [==============================] - 393s 2s/step - loss: 0.4950 -
accuracy: 0.7731 - val_loss: 0.4657 - val_accuracy: 0.7889
Epoch 11/50
244/244 [==============================] - 392s 2s/step - loss: 0.4860 -
accuracy: 0.7793 - val_loss: 0.4797 - val_accuracy: 0.7794
Epoch 12/50
244/244 [==============================] - 389s 2s/step - loss: 0.4706 -
```

```
accuracy: 0.7852 - val_loss: 0.4520 - val_accuracy: 0.7979
Epoch 13/50
244/244 [==============================] - 389s 2s/step - loss: 0.4629 -
accuracy: 0.7886 - val_loss: 0.4291 - val_accuracy: 0.8092
Epoch 14/50
244/244 [==============================] - 396s 2s/step - loss: 0.4580 -
accuracy: 0.7927 - val_loss: 0.4828 - val_accuracy: 0.7827
Epoch 15/50
244/244 [==============================] - 393s 2s/step - loss: 0.4414 -
accuracy: 0.8015 - val_loss: 0.4314 - val_accuracy: 0.8172
Epoch 16/50
244/244 [==============================] - 395s 2s/step - loss: 0.4347 -
accuracy: 0.8069 - val_loss: 0.4131 - val_accuracy: 0.8220
Epoch 17/50
244/244 [==============================] - 392s 2s/step - loss: 0.4292 -
accuracy: 0.8087 - val_loss: 0.3881 - val_accuracy: 0.8344
Epoch 18/50
244/244 [==============================] - 395s 2s/step - loss: 0.4192 -
accuracy: 0.8159 - val_loss: 0.3977 - val_accuracy: 0.8187
Epoch 19/50
244/244 [==============================] - 396s 2s/step - loss: 0.4087 -
accuracy: 0.8199 - val_loss: 0.3715 - val_accuracy: 0.8423
Epoch 20/50
244/244 [==============================] - 391s 2s/step - loss: 0.4138 -
accuracy: 0.8181 - val_loss: 0.3599 - val_accuracy: 0.8439
Epoch 21/50
244/244 [==============================] - 377s 2s/step - loss: 0.4067 -
accuracy: 0.8216 - val_loss: 0.3832 - val_accuracy: 0.8346
Epoch 22/50
244/244 [==============================] - 375s 2s/step - loss: 0.3953 -
accuracy: 0.8258 - val_loss: 0.3409 - val_accuracy: 0.8505
Epoch 23/50
244/244 [==============================] - 377s 2s/step - loss: 0.3898 -
accuracy: 0.8317 - val_loss: 0.3644 - val_accuracy: 0.8400
Epoch 24/50
244/244 [==============================] - 381s 2s/step - loss: 0.3902 -
accuracy: 0.8293 - val_loss: 0.3469 - val_accuracy: 0.8582
Epoch 25/50
244/244 [==============================] - 381s 2s/step - loss: 0.3846 -
accuracy: 0.8334 - val_loss: 0.3520 - val_accuracy: 0.8631
Epoch 26/50
244/244 [==============================] - 381s 2s/step - loss: 0.3833 -
accuracy: 0.8344 - val_loss: 0.3800 - val_accuracy: 0.8282
Epoch 27/50
244/244 [==============================] - 380s 2s/step - loss: 0.3991 -
accuracy: 0.8239 - val_loss: 0.3513 - val_accuracy: 0.8485
Epoch 28/50
244/244 [==============================] - 380s 2s/step - loss: 0.3945 -
```

```
accuracy: 0.8263 - val_loss: 0.3448 - val_accuracy: 0.8608
Epoch 29/50
244/244 [==============================] - 380s 2s/step - loss: 0.3868 -
accuracy: 0.8308 - val_loss: 0.3548 - val_accuracy: 0.8649
Epoch 30/50
244/244 [==============================] - 381s 2s/step - loss: 0.3828 -
accuracy: 0.8316 - val_loss: 0.3242 - val_accuracy: 0.8598
Epoch 31/50
244/244 [==============================] - 379s 2s/step - loss: 0.3796 -
accuracy: 0.8342 - val_loss: 0.3518 - val_accuracy: 0.8505
Epoch 32/50
244/244 [==============================] - 377s 2s/step - loss: 0.3738 -
accuracy: 0.8383 - val_loss: 0.3209 - val_accuracy: 0.8629
Epoch 33/50
244/244 [==============================] - 379s 2s/step - loss: 0.3692 -
accuracy: 0.8402 - val_loss: 0.3195 - val_accuracy: 0.8665
Epoch 34/50
244/244 [==============================] - 394s 2s/step - loss: 0.3837 -
accuracy: 0.8272 - val_loss: 0.3511 - val_accuracy: 0.8683
Epoch 35/50
244/244 [==============================] - 393s 2s/step - loss: 0.3815 -
accuracy: 0.8289 - val_loss: 0.3155 - val_accuracy: 0.8724
Epoch 36/50
244/244 [==============================] - 396s 2s/step - loss: 0.3553 -
accuracy: 0.8471 - val_loss: 0.3184 - val_accuracy: 0.8711
Epoch 37/50
244/244 [==============================] - 400s 2s/step - loss: 0.3671 -
accuracy: 0.8367 - val_loss: 0.3491 - val_accuracy: 0.8523
Epoch 38/50
244/244 [==============================] - 396s 2s/step - loss: 0.3623 -
accuracy: 0.8364 - val_loss: 0.3186 - val_accuracy: 0.8729
Epoch 39/50
244/244 [==============================] - 399s 2s/step - loss: 0.3703 -
accuracy: 0.8306 - val_loss: 0.3169 - val_accuracy: 0.8788
Epoch 40/50
244/244 [==============================] - 399s 2s/step - loss: 0.3773 -
accuracy: 0.8204 - val_loss: 0.3312 - val_accuracy: 0.8675
Epoch 41/50
244/244 [==============================] - 397s 2s/step - loss: 0.3719 -
accuracy: 0.8256 - val_loss: 0.3011 - val_accuracy: 0.8765
Epoch 42/50
244/244 [==============================] - 398s 2s/step - loss: 0.3683 -
accuracy: 0.8338 - val_loss: 0.3225 - val_accuracy: 0.8739
Epoch 43/50
244/244 [==============================] - 393s 2s/step - loss: 0.3634 -
accuracy: 0.8338 - val_loss: 0.3095 - val_accuracy: 0.8778
Epoch 44/50
244/244 [==============================] - 399s 2s/step - loss: 0.3657 -
```

```
accuracy: 0.8362 - val_loss: 0.3352 - val_accuracy: 0.8665
Epoch 45/50
244/244 [==============================] - 396s 2s/step - loss: 0.3509 -
accuracy: 0.8447 - val_loss: 0.3028 - val_accuracy: 0.8793
Epoch 46/50
244/244 [==============================] - 398s 2s/step - loss: 0.3477 -
accuracy: 0.8448 - val_loss: 0.3538 - val_accuracy: 0.8526
Epoch 47/50
244/244 [==============================] - 389s 2s/step - loss: 0.3586 -
accuracy: 0.8379 - val_loss: 0.2907 - val_accuracy: 0.8896
Epoch 48/50
244/244 [==============================] - 377s 2s/step - loss: 0.3478 -
accuracy: 0.8454 - val_loss: 0.3028 - val_accuracy: 0.8742
Epoch 49/50
244/244 [==============================] - 376s 2s/step - loss: 0.3503 -
accuracy: 0.8421 - val_loss: 0.2932 - val_accuracy: 0.8857
Epoch 50/50
244/244 [==============================] - 377s 2s/step - loss: 0.3444 -
accuracy: 0.8471 - val_loss: 0.2935 - val_accuracy: 0.8783
```

5.1 Training and Testing Scores

[19]:
```
# evaluate the keras model
train_eval = fashion_model.evaluate(train_X, train_label, verbose=1)
print('Test loss:', train_eval[0])
print('Test accuracy:', train_eval[1])
```

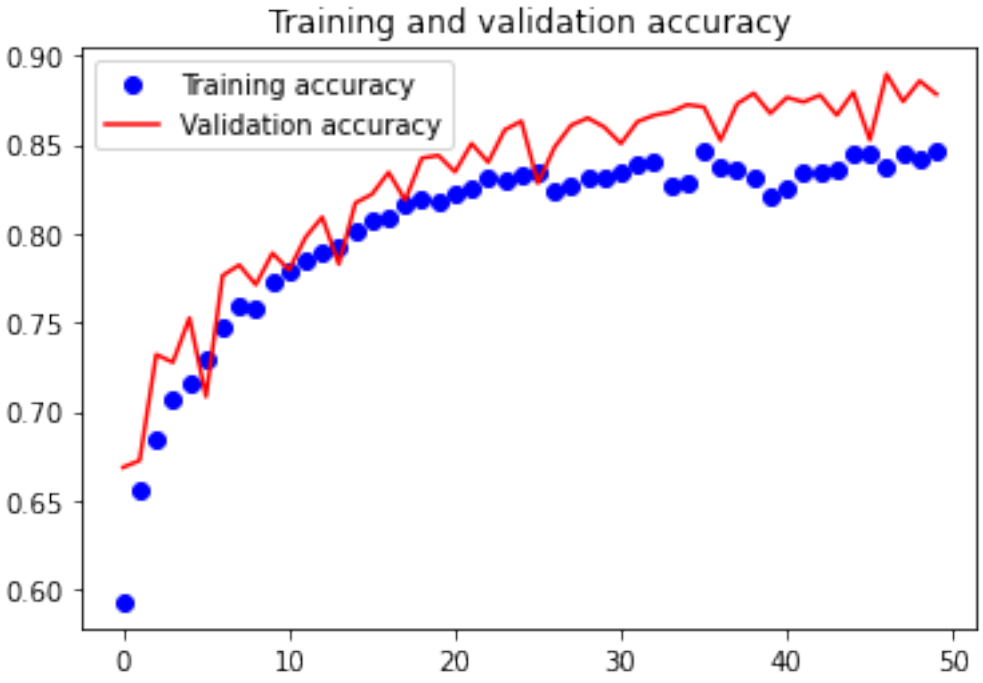
```
487/487 [==============================] - 142s 291ms/step - loss: 0.2609 -
accuracy: 0.8954
Test loss: 0.2609248459339142
Test accuracy: 0.8954160213470459
```

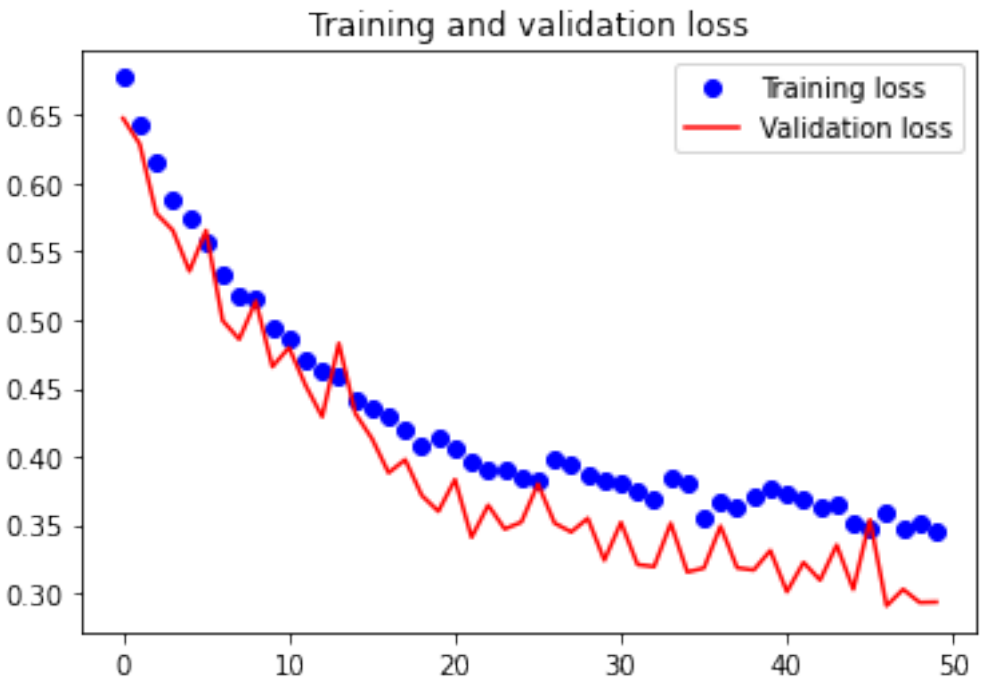
[20]:
```
# evaluate the keras model
test_eval = fashion_model.evaluate(valid_X, valid_label, verbose=1)
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

```
122/122 [==============================] - 35s 289ms/step - loss: 0.2935 -
accuracy: 0.8783
Test loss: 0.29352545738220215
Test accuracy: 0.8782742619514465
```

6 Visualize Outcomes

[21]:
```
accuracy = fashion_train.history['accuracy']
val_accuracy = fashion_train.history['val_accuracy']
loss = fashion_train.history['loss']
val_loss = fashion_train.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'bo', label='Training accuracy')
plt.plot(epochs, val_accuracy, 'r', label='Validation accuracy', color='r')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss', color='r')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





7 Confusion Matrix and Classification Report

[22]:
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import time
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler # standardize features by␣
 ↪removing the mean and scaling to unit variance.
from sklearn.metrics import confusion_matrix
#from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV
```

[23]:
```
# flatten the true labels to 1D
print(valid_label)
valid_label = valid_label.flatten()
print(valid_label.shape)
```

```
[[0.]
 [1.]
 [1.]
 ...
 [0.]
 [1.]
 [1.]]
(3894,)
```

[24]:
```
# extract the predicted probabilities, flatten the prediction to 1D
p_pred = fashion_model.predict(valid_X)
p_pred = p_pred.flatten()
print(p_pred.round(2))
```

```
[0.02 0.87 0.66 ... 0.04 0.94 0.97]
```

[25]:
```
# extract the predicted class labels
y_pred = np.where(p_pred > 0.5, 1, 0)
print(y_pred)
print(y_pred.shape)
```

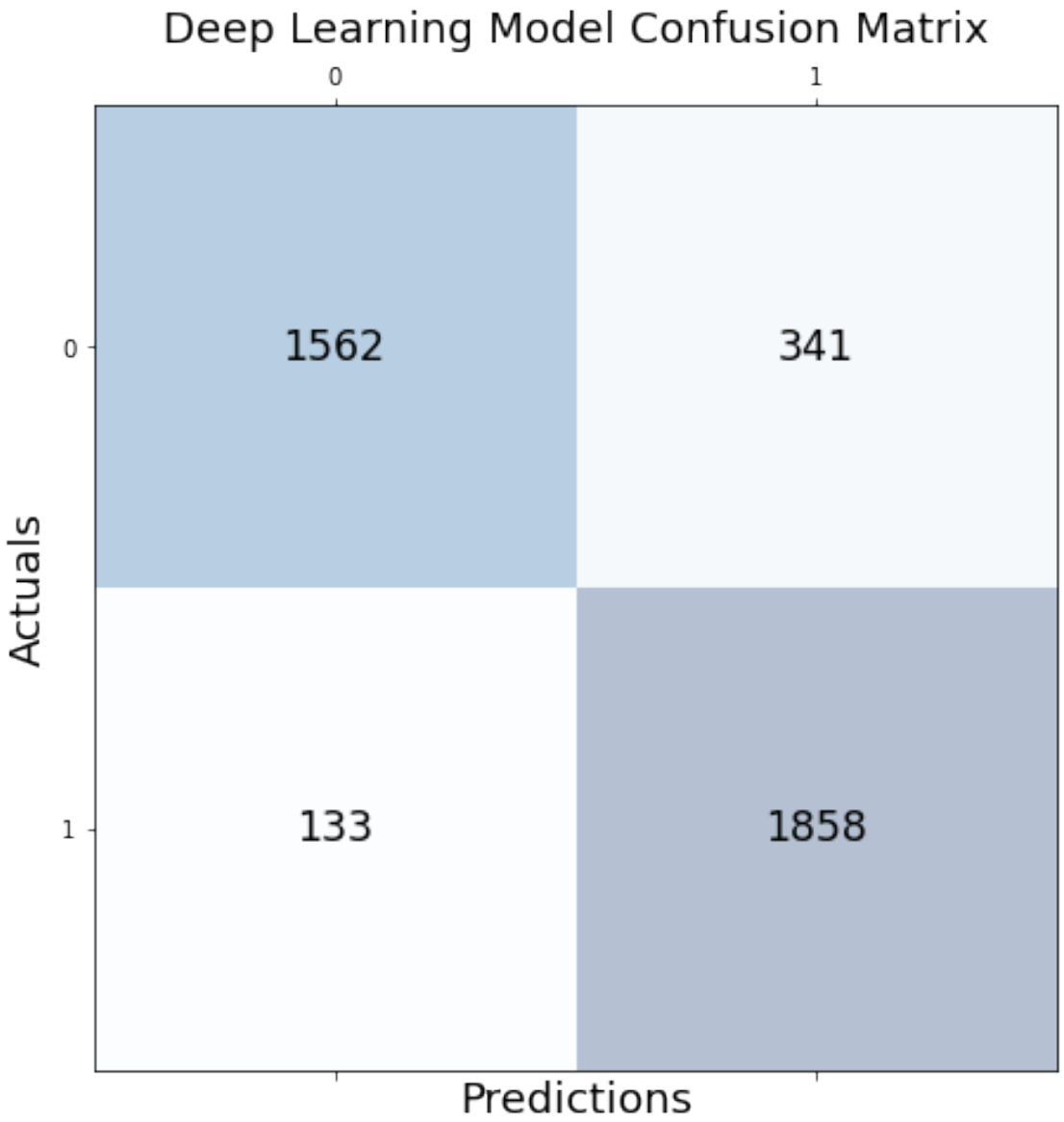
```
[0 1 1 ... 0 1 1]
(3894,)
```

[26]:
```
# confusion matrix
conf_matrix = confusion_matrix(valid_label, y_pred)
print(confusion_matrix(valid_label, y_pred))
fig, ax = plt.subplots(figsize=(7.5, 7.5))
ax.matshow(conf_matrix, cmap=plt.cm.Blues, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center',␣
 ↪size='xx-large')

plt.xlabel('Predictions', fontsize=18)
plt.ylabel('Actuals', fontsize=18)
plt.title('Deep Learning Model Confusion Matrix', fontsize=18)
plt.show()
plt.savefig('dp_confusion_matrix.png')

print("------------Classification Report----------")
target_names = ['fake', 'real']
print(classification_report(valid_label, y_pred, target_names=target_names))
```

```
[[1562  341]
 [ 133 1858]]
```




```
------------Classification Report----------
              precision    recall  f1-score   support

        fake       0.92      0.82      0.87      1903
        real       0.84      0.93      0.89      1991

    accuracy                           0.88      3894
   macro avg       0.88      0.88      0.88      3894
weighted avg       0.88      0.88      0.88      3894


<Figure size 432x288 with 0 Axes>
```

8 References

- https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
- https://www.machinecurve.com/index.php/2020/04/05/how-to-find-the-value-for-keras-input_shape-input_dim/
- https://towardsdatascience.com/activation-functions-in-deep-neural-networks-aae2a598f211
- https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/
- https://stackoverflow.com/questions/69875073/confusion-matrix-valueerror-classification-metrics-cant-handle-a-mix-of-binary

9 Output

[6]:
```
# should access the Google Drive files before running the chunk
%%capture
%cd /content/drive/MyDrive/American_University/2021_Fall/DATA-642-001_Advanced␣
 ↪Machine Learning/Deepfake_Video_Classifier2.0/model_outcomes/nn_model
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-plain-generic
!jupyter nbconvert --to pdf "/content/drive/MyDrive/American_University/
 ↪2021_Fall/DATA-642-001_Advanced Machine Learning/Deepfake_Video_Classifier2.
 ↪0/code/07.Deep_Learning_Task.ipynb"
```