Homework2 – Introduction to Artificial Neural Networks with Keras

Building an Image Classifier

First let's import TensorFlow and Keras.

In [1]:
```
import tensorflow as tf
from tensorflow import keras
```

In [2]:
```
import os
import gzip
import numpy as np

current_dir = os.getcwd()   # First, get the path of the working directory
path =current_dir+'\\Fashion_MNIST_Data'

# Second, import the Fashion MNIST data from the current directory+'\\Fashion
f = gzip.open(path+'\\train-labels-idx1-ubyte.gz','rb')  #Load the training l
y_train_full=np.frombuffer(f.read(), dtype=np.uint8,offset=8) # due to header
f.close()

f = gzip.open(path+'\\train-images-idx3-ubyte.gz','rb')  #Load the training d
X_train_full=np.frombuffer(f.read(), dtype=np.uint8,offset=16) # due to heade
f.close()
X_train_full=X_train_full.reshape(len(y_train_full), 784)
X_train_full1=X_train_full.reshape(len(y_train_full), 28,28)

f = gzip.open(path+'\\t10k-labels-idx1-ubyte.gz','rb')   #Load the test label
y_test=np.frombuffer(f.read(), dtype=np.uint8,offset=8)
f.close()

f = gzip.open(path+'\\t10k-images-idx3-ubyte.gz','rb')   #Load the test data
X_test=np.frombuffer(f.read(), dtype=np.uint8,offset=16).reshape(len(y_test),

f.close()
```

In [3]:
```
# Check the consistency
print(len(X_train_full))
print(len(y_train_full))
print(len(X_test))
print(len(y_test))
y_test
```

```
60000
60000
10000
10000
```

Out[3]:
```
array([9, 2, 1, ..., 8, 1, 5], dtype=uint8)
```

The training set contains 60,000 grayscale images, each 28×28 pixels. The class labels are:

Label: Description

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

In [4]:
```
#Check the shape of the dataset
print(X_train_full.shape)
print(y_train_full.shape)
print(X_test.shape)
```

```
(60000, 784)
(60000,)
(10000, 784)
```

Each pixel intensity is represented as a byte (0 to 255):

Let's split the full training set into a validation set and a (smaller) training set. We also scale the pixel intensities down to the 0-1 range and convert them to floats, by dividing by `255.` .

In [6]:
```
X_valid, X_train = X_train_full[:5000] / 255., X_train_full[5000:] / 255.
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
X_test = X_test / 255.
```

Here are the corresponding class names:

In [7]:
```
class_names = ["T-shirt/top", "Trouser", "Pullover", "Dress", "Coat", "Sandal
               "Shirt", "Sneaker", "Bag", "Ankle boot"]
```

So the first image in the training set is a coat:

In [8]:
```
class_names[y_train[0]]
```

Out[8]: 'Coat'

Let's take a look at a sample of the images in the dataset:

In [11]:
```
import matplotlib.pyplot as plt
n_rows = 5
n_cols = 8
plt.figure(figsize=(n_cols * 1.2, n_rows * 1.2)) # scale up, otherwsie, 28x28
for row in range(n_rows):
    for col in range(n_cols):
        index = n_cols * row + col
        plt.subplot(n_rows, n_cols, index + 1)
        plt.imshow(X_train_full1[index], cmap="binary", interpolation="neares
        #plt.axis('off')
        plt.title(class_names[y_train_full[index]], fontsize=12)
plt.subplots_adjust(wspace=0.2, hspace=1.0)

plt.show()
```

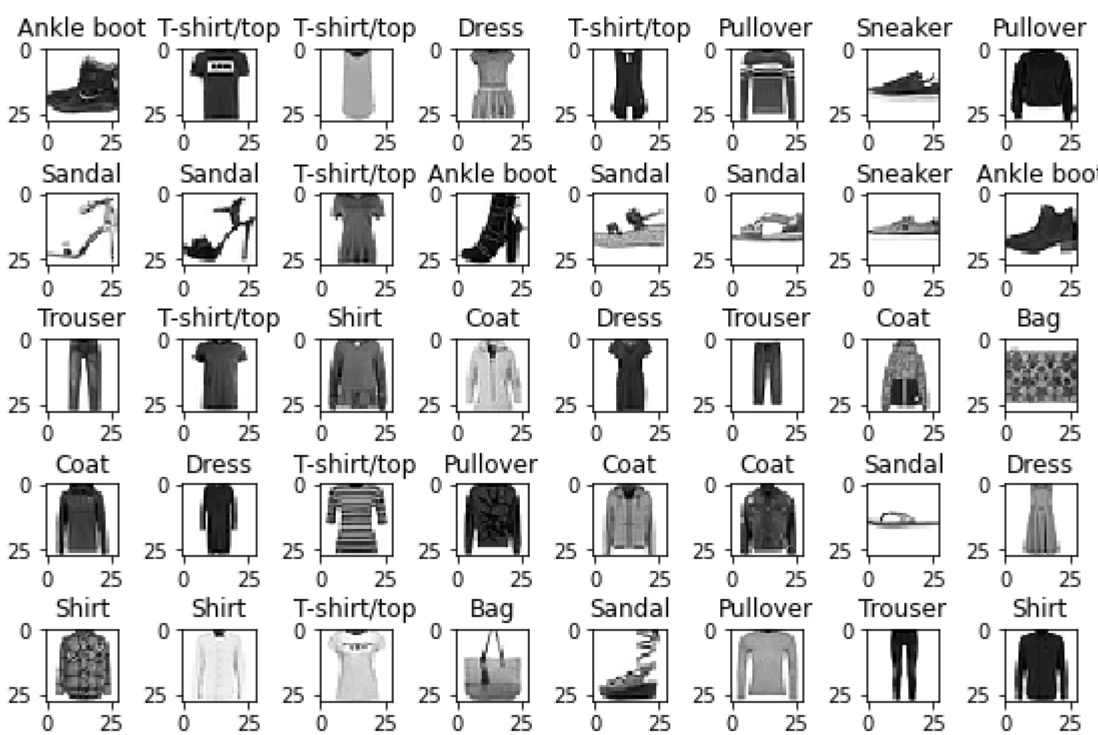


Figure 2: Samples from Fashion MNIST in a 4 x 10 grid.

Model 1

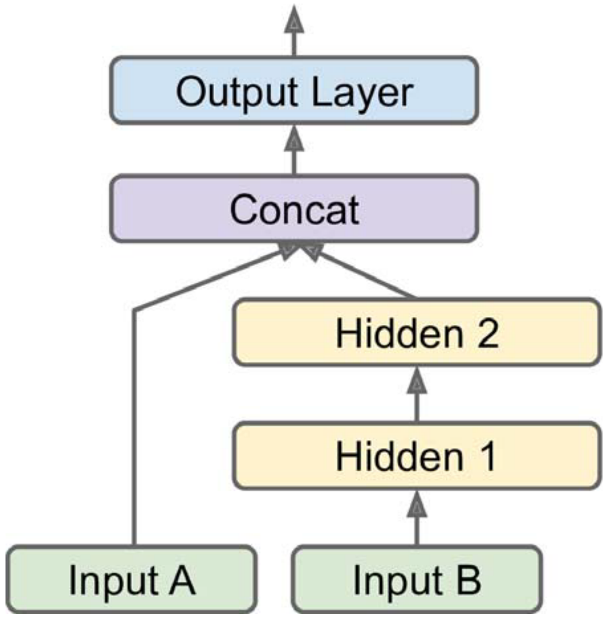


Fig. 6: Handling multiple inputs.

In [12]:
```
np.random.seed(42)
tf.random.set_seed(42)
```

In [13]:
```
input_A = keras.layers.Input(shape=[400], name="wide_input")
input_B = keras.layers.Input(shape=[400], name="deep_input")
hidden1 = keras.layers.Dense(30, activation="relu")(input_B)
hidden2 = keras.layers.Dense(30, activation="relu")(hidden1)
concat = keras.layers.concatenate([input_A, hidden2])
output = keras.layers.Dense(10, name="output",activation="softmax")(concat)
model1 = keras.models.Model(inputs=[input_A, input_B], outputs=[output])
```

In [14]:
```
model1.compile(loss=keras.losses.sparse_categorical_crossentropy,
              optimizer=keras.optimizers.SGD(),
              metrics=[keras.metrics.sparse_categorical_accuracy])
checkpoint_cb = keras.callbacks.ModelCheckpoint("my_keras_model1.h5", save_be
```

In [15]:
```
model1.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 deep_input (InputLayer)        [(None, 400)]        0           []

 dense (Dense)                  (None, 30)           12030       ['deep_input[0][0]']

 wide_input (InputLayer)        [(None, 400)]        0           []

 dense_1 (Dense)                (None, 30)           930         ['dense[0][0]']

 concatenate (Concatenate)      (None, 430)          0           ['wide_input[0][0]',
                                                                  'dense_1[0][0]']

 output (Dense)                 (None, 10)           4310        ['concatenate[0][0]']

==================================================================================================
Total params: 17,270
Trainable params: 17,270
Non-trainable params: 0
__________________________________________________________________________________________________
```

In [16]:
```
X_train_A, X_train_B = X_train[:, :400], X_train[:, 384:]
X_valid_A, X_valid_B= X_valid[:, :400], X_valid[:, 384:]
X_test_A, X_test_B = X_test[:, :400], X_test[:, 384:]
```

In [17]:
```
history1 = model1.fit((X_train_A, X_train_B), y_train, epochs=2000,
                      validation_data=((X_valid_A, X_valid_B), y_valid),callbac
```

```
sparse_categorical_accuracy: 0.8222 - val_loss: 0.4898 - val_sparse_categ
orical_accuracy: 0.8336
Epoch 5/2000
1719/1719 [==============================] - 6s 3ms/step - loss: 0.4893 -
sparse_categorical_accuracy: 0.8294 - val_loss: 0.4658 - val_sparse_categ
orical_accuracy: 0.8424
Epoch 6/2000
1719/1719 [==============================] - 5s 3ms/step - loss: 0.4712 -
sparse_categorical_accuracy: 0.8350 - val_loss: 0.4524 - val_sparse_categ
orical_accuracy: 0.8470
Epoch 7/2000
1719/1719 [==============================] - 5s 3ms/step - loss: 0.4573 -
sparse_categorical_accuracy: 0.8407 - val_loss: 0.4413 - val_sparse_categ
orical_accuracy: 0.8502
Epoch 8/2000
1719/1719 [==============================] - 6s 3ms/step - loss: 0.4454 -
sparse_categorical_accuracy: 0.8449 - val_loss: 0.4448 - val_sparse_categ
orical_accuracy: 0.8428
Epoch 9/2000
1719/1719 [==============================] - 5s 3ms/step - loss: 0.4353 -
```

Loading saved best version of the models and evaluating them using the test dataset.

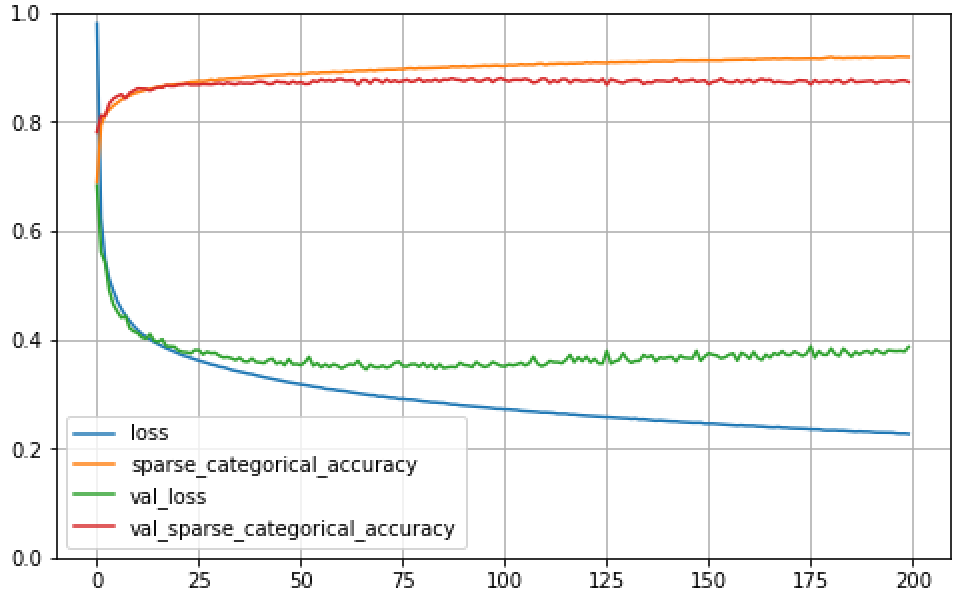
In [21]:
```
model1 = keras.models.load_model("my_keras_model1.h5") # rollback to best moc
```

In [22]:
```
y_pred_main= model1.evaluate((X_test_A, X_test_B),y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 0.3758 - spa
rse_categorical_accuracy: 0.8682
```

In [23]:

```
import pandas as pd
X=pd.DataFrame(history1.history)
X=X[:200]
X.plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



Model 2

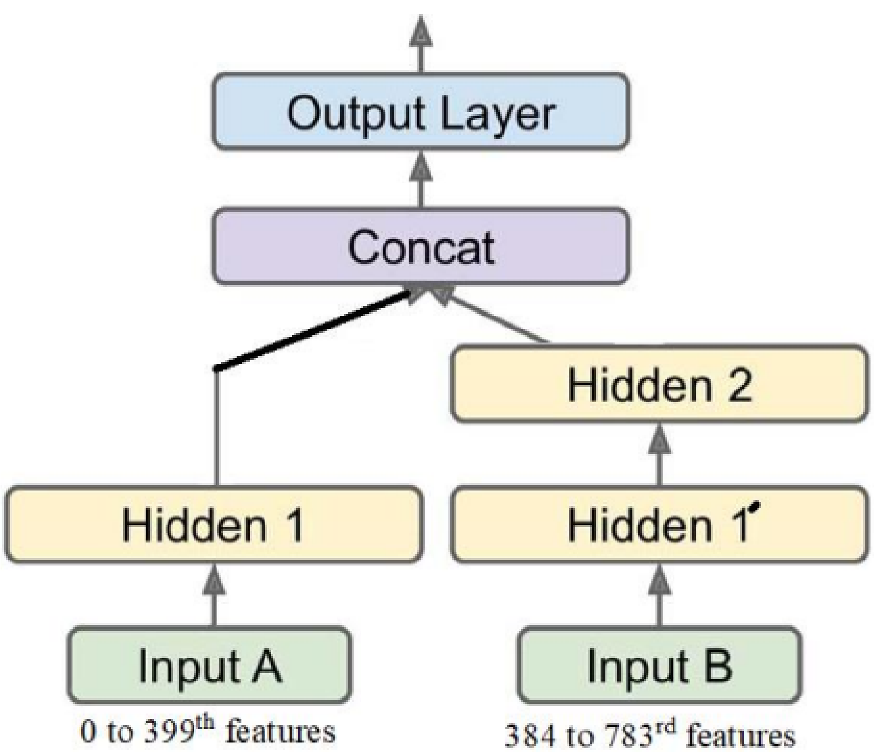


Fig. 7: Handling multiple inputs.

In [12]:
```
np.random.seed(42)
tf.random.set_seed(42)
```

In [13]:
```
input_A = keras.layers.Input(shape=[400], name="wide_input")
hidden1 = keras.layers.Dense(30, activation="relu")(input_A)
input_B = keras.layers.Input(shape=[400], name="deep_input")
hidden11 = keras.layers.Dense(30, activation="relu")(input_B)
hidden12 = keras.layers.Dense(30, activation="relu")(hidden11)
concat = keras.layers.concatenate([hidden1, hidden12])
output = keras.layers.Dense(10, name="output",activation="softmax")(concat)
model2 = keras.models.Model(inputs=[input_A, input_B], outputs=[output])
```

In [14]:
```
model2.compile(loss=keras.losses.sparse_categorical_crossentropy,
              optimizer=keras.optimizers.SGD(),
              metrics=[keras.metrics.sparse_categorical_accuracy])
checkpoint_cb = keras.callbacks.ModelCheckpoint("my_keras_model1.h5", save_be
```

In [15]:
```
model2.summary()
```

```
Model: "model"
__________________________________________________________________________
________________________
 Layer (type)                   Output Shape         Param #     Connected
to
==========================================================================
========================
 deep_input (InputLayer)        [(None, 400)]        0           []

 wide_input (InputLayer)        [(None, 400)]        0           []

 dense_1 (Dense)                (None, 30)           12030       ['deep_inp
ut[0][0]']

 dense (Dense)                  (None, 30)           12030       ['wide_inp
ut[0][0]']

 dense_2 (Dense)                (None, 30)           930         ['dense_1
[0][0]']

 concatenate (Concatenate)      (None, 60)           0           ['dense[0]
[0]',
                                                                  'dense_2
[0][0]']

 output (Dense)                 (None, 10)           610         ['concaten
ate[0][0]']

==========================================================================
========================
Total params: 25,600
Trainable params: 25,600
Non-trainable params: 0
__________________________________________________________________________
________________________
```

In [16]:
```
X_train_A, X_train_B = X_train[:, :400], X_train[:, 384:]
X_valid_A, X_valid_B= X_valid[:, :400], X_valid[:, 384:]
X_test_A, X_test_B = X_test[:, :400], X_test[:, 384:]
X_new_A, X_new_B = X_test_A[:1000], X_test_B[:1000]
```

In [17]:

```
history2 = model2.fit((X_train_A, X_train_B), y_train, epochs=2000,
                      validation_data=((X_valid_A, X_valid_B), y_valid),callbac
```

```
Epoch 1560/2000
1719/1719 [==============================] - 6s 4ms/step - loss: 0.0246 -
sparse_categorical_accuracy: 0.9929 - val_loss: 1.3215 - val_sparse_categ
orical_accuracy: 0.8562
Epoch 1561/2000
1719/1719 [==============================] - 6s 4ms/step - loss: 0.0239 -
sparse_categorical_accuracy: 0.9933 - val_loss: 1.3333 - val_sparse_categ
orical_accuracy: 0.8590
Epoch 1562/2000
1719/1719 [==============================] - 6s 4ms/step - loss: 0.0229 -
sparse_categorical_accuracy: 0.9933 - val_loss: 1.3263 - val_sparse_categ
orical_accuracy: 0.8574
Epoch 1563/2000
1719/1719 [==============================] - 6s 4ms/step - loss: 0.0225 -
sparse_categorical_accuracy: 0.9936 - val_loss: 1.3143 - val_sparse_categ
orical_accuracy: 0.8622
Epoch 1564/2000
1719/1719 [==============================] - 6s 4ms/step - loss: 0.0264 -
sparse_categorical_accuracy: 0.9921 - val_loss: 1.3089 - val_sparse_categ
orical accuracy: 0.8632
```

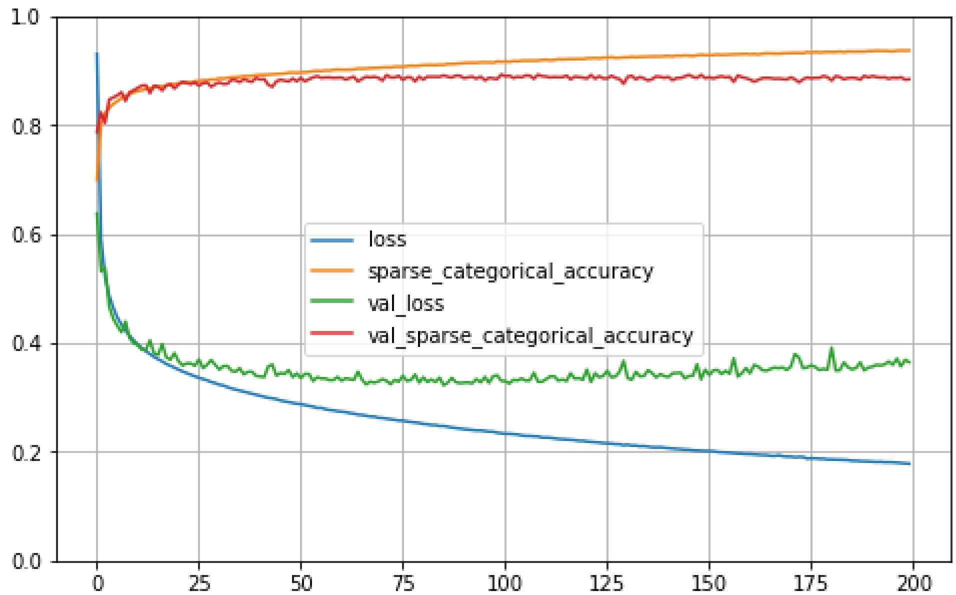
Loading saved best version of the models and evaluating them using the test dataset.

In [18]:

```
model2 = keras.models.load_model("my_keras_model2.h5") # rollback to best moc
y_pred_main2= model2.evaluate((X_test_A, X_test_B),y_test)
```

```
313/313 [==============================] - 1s 2ms/step - loss: 1.6682 - spa
rse_categorical_accuracy: 0.8536
```

In [19]:
```
import pandas as pd
X2=pd.DataFrame(history2.history)
X2=X2[:200]
X2.plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



Model 3

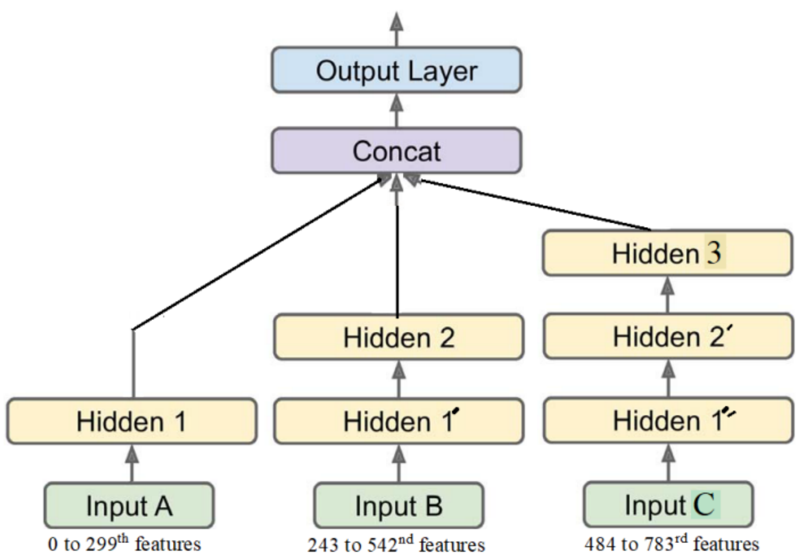


Fig. 7: Handling multiple inputs.

In [12]:
```
np.random.seed(42)
tf.random.set_seed(42)
```

In [13]:

```
input_A = keras.layers.Input(shape=[300], name="wide_input")
input_B = keras.layers.Input(shape=[300], name="deep_input1")
input_c = keras.layers.Input(shape=[300], name="deep_input2")
hidden1 = keras.layers.Dense(30, activation="relu")(input_A)
hidden11 = keras.layers.Dense(30, activation="relu")(input_B)
hidden12 = keras.layers.Dense(30, activation="relu")(hidden11)
hidden21 = keras.layers.Dense(30, activation="relu")(input_c)
hidden22 = keras.layers.Dense(30, activation="relu")(hidden21)
hidden33 = keras.layers.Dense(30, activation="relu")(hidden22)

concat = keras.layers.concatenate([hidden1, hidden12,hidden33])
output = keras.layers.Dense(10, name="output",activation="softmax")(concat)
model3 = keras.models.Model(inputs=[input_A, input_B,input_c], outputs=[outpu
```

In [14]:

```
X_train_A, X_train_B ,X_train_C = X_train[:, :300], X_train[:, 243:543],X_tra
X_valid_A, X_valid_B ,X_valid_C = X_valid[:, :300], X_valid[:, 243:543],X_val
X_test_A, X_test_B,X_test_C= X_test[:, :300], X_test[:, 243:543],X_test[:,484

X_new_A, X_new_B, X_new_C= X_test_A[:1000], X_test_B[:1000],X_test_C[:1000]
```

In [28]:

```
model3.compile(loss=keras.losses.sparse_categorical_crossentropy,
              optimizer=keras.optimizers.SGD(),
              metrics=[keras.metrics.sparse_categorical_accuracy])
checkpoint_cb = keras.callbacks.ModelCheckpoint("my_keras_model3.h5", save_be
```

In [29]:

```
model3.summary()
```

```
Model: "model"
__________________________________________________________________________________________
__________________________
 Layer (type)                   Output Shape         Param #     Connected
to
==========================================================================================
=======================
 deep_input2 (InputLayer)       [(None, 300)]        0           []

 deep_input1 (InputLayer)       [(None, 300)]        0           []

 dense_3 (Dense)                (None, 30)           9030        ['deep_inp
ut2[0][0]']

 wide_input (InputLayer)        [(None, 300)]        0           []

 dense_1 (Dense)                (None, 30)           9030        ['deep_inp
ut1[0][0]']

 dense_4 (Dense)                (None, 30)           930         ['dense_3
[0][0]']

 dense (Dense)                  (None, 30)           9030        ['wide_inp
ut[0][0]']

 dense_2 (Dense)                (None, 30)           930         ['dense_1
[0][0]']

 dense_5 (Dense)                (None, 30)           930         ['dense_4
[0][0]']

 concatenate (Concatenate)      (None, 90)           0           ['dense[0]
[0]',
                                                                  'dense_2
[0][0]',
                                                                  'dense_5
[0][0]']

 output (Dense)                 (None, 10)           910         ['concaten
ate[0][0]']

==========================================================================================
=======================
Total params: 30,790
Trainable params: 30,790
Non-trainable params: 0
__________________________________________________________________________________________
__________________________
```

In [31]:
```
history3 = model3.fit((X_train_A, X_train_B,X_train_C), y_train, epochs=1700,
                      validation_data=((X_valid_A, X_valid_B,X_valid_C), y_vali
```

```
sparse_categorical_accuracy: 1.0000 - val_loss: 1.4241 - val_sparse_categ
orical_accuracy: 0.8770
Epoch 1251/1700
1719/1719 [==============================] - 7s 4ms/step - loss: 0.0019 -

sparse_categorical_accuracy: 1.0000 - val_loss: 1.4267 - val_sparse_categ
orical_accuracy: 0.8772
Epoch 1252/1700
1719/1719 [==============================] - 7s 4ms/step - loss: 0.0016 -
sparse_categorical_accuracy: 1.0000 - val_loss: 1.4599 - val_sparse_categ
orical_accuracy: 0.8762
Epoch 1291/1700
1719/1719 [==============================] - 7s 4ms/step - loss: 0.0017 -
sparse_categorical_accuracy: 1.0000 - val_loss: 1.4558 - val_sparse_categ
orical_accuracy: 0.8768
Epoch 1292/1700
1719/1719 [==============================] - 7s 4ms/step - loss: 0.0016 -
sparse_categorical_accuracy: 1.0000 - val_loss: 1.4566 - val_sparse_categ
orical_accuracy: 0.8762
Epoch 1293/1700
```

In [ ]:

NOTE; my third model was always crushig , hence i had to reduce the number of epoch to 1700.

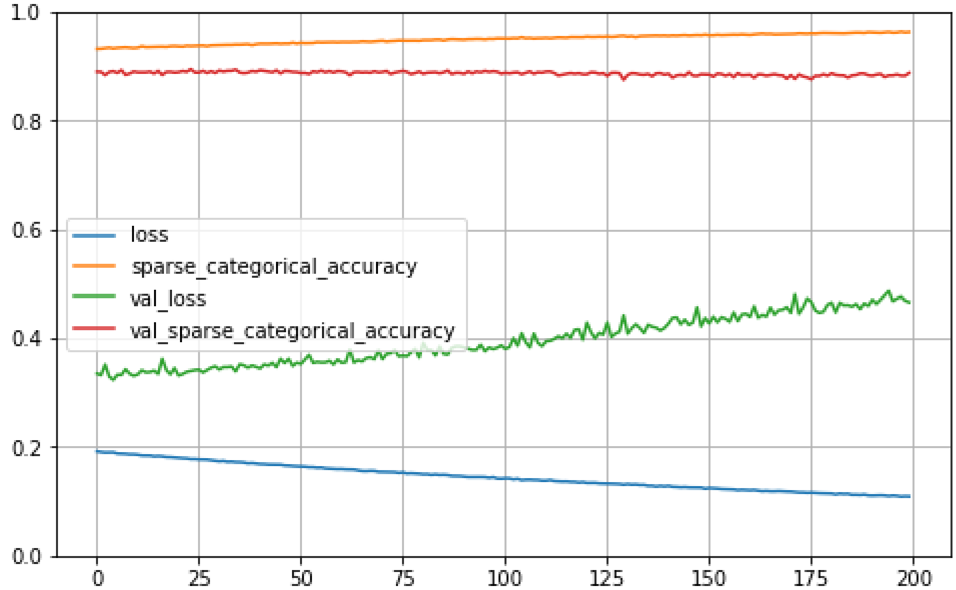
Loading saved best version of the models and evaluating them using the test dataset.

In [32]:
```
model3 = keras.models.load_model("my_keras_model3.h5") # rollback to best moc
y_pred_main3= model3.evaluate((X_test_A, X_test_B,X_test_C),y_test)
```

```
313/313 [==============================] - 3s 3ms/step - loss: 0.3389 - spa
rse_categorical_accuracy: 0.8814
```

In [34]:

```
import pandas as pd
X3=pd.DataFrame(history3.history)
X3=X3[:200]
X3.plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



In [ ]:

```
```