HW5

May 24, 2020

[1]:
```
# imports for array-handling and plotting
import numpy as np
import matplotlib
matplotlib.use('agg')
import matplotlib.pyplot as plt
%matplotlib inline

# let's keep our keras backend tensorflow quiet
import os
# for testing on GPU
#os.environ['TF_CPP_MIN_LOG_LEVEL']='3'
# for testing on CPU
os.environ['CUDA_VISIBLE_DEVICES'] = ''

# keras imports for the dataset and building our neural network
from keras.datasets import mnist
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation, Dense
from keras.utils import np_utils
from keras import backend as K

from keras.layers import Conv2D, MaxPooling2D, Flatten
from keras.preprocessing.image import ImageDataGenerator
```

```
Using TensorFlow backend.
```

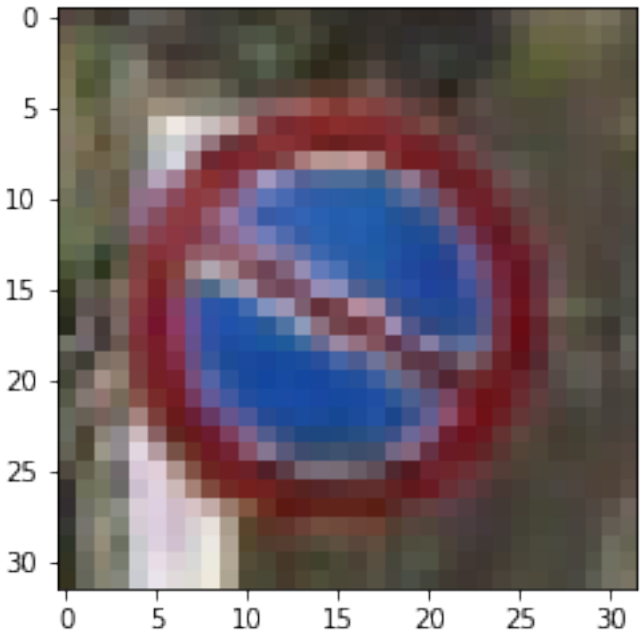
[2]:
```
def classificator (directory):
    childDirectories = next(os.walk(directory))[1]
    for x in range(len(childDirectories)):
        childDirectories[x]=int(childDirectories[x])
    childDirectories.sort()
    for x in range(len(childDirectories)):
        childDirectories[x]=str(childDirectories[x])
    return childDirectories
```

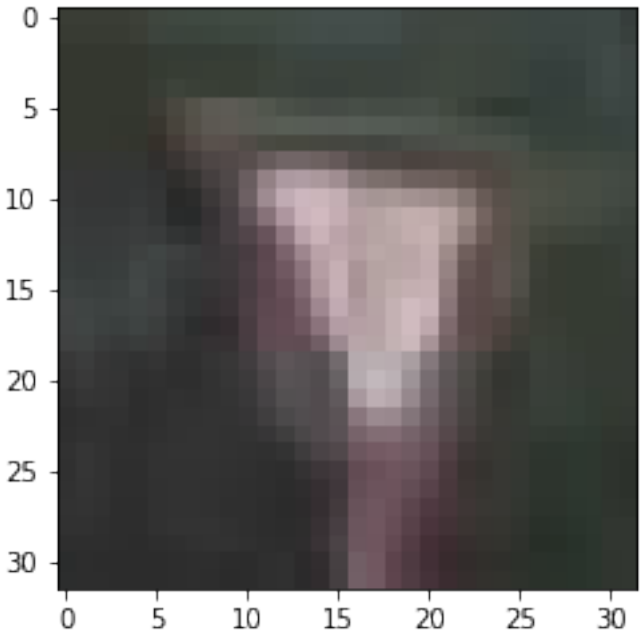
[3]:
```
# Variabili globali
train_data_dir = 'DITS-classification/classification train'
test_data_dir = 'DITS-classification/classification test'
```

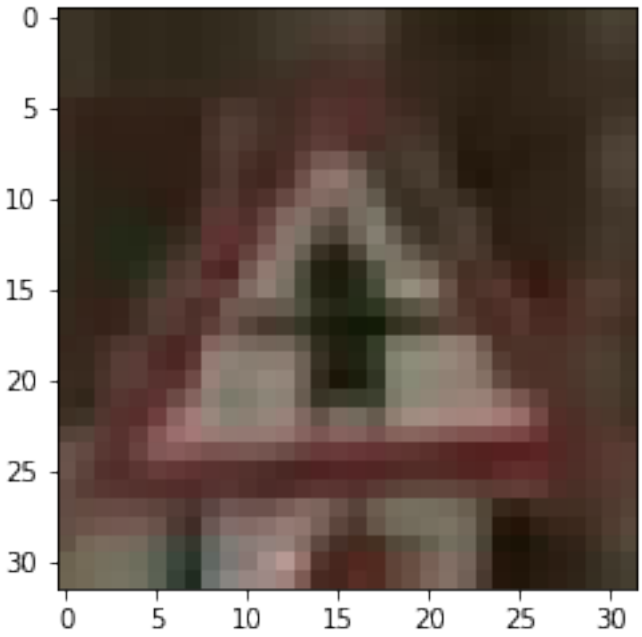
```
nb_train_samples = 7489
nb_test_samples = 1159
epoche = 200
batch_size = 150
img_width, img_height = 32, 32

# this is the augmentation configuration we will use for training
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=False)

# this is the augmentation configuration we will use for testing:
# only rescaling
test_datagen = ImageDataGenerator(rescale=1. / 255.0)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    color_mode="rgb",
    batch_size=nb_train_samples,
    classes = classificator(train_data_dir),
    class_mode='categorical')

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(img_width, img_height),
    color_mode="rgb",
    batch_size=nb_test_samples,
    classes = classificator(test_data_dir),
    class_mode="categorical")

X_train = train_generator[0][0]
Y_train = train_generator[0][1]
X_test = test_generator[0][0]
Y_test = test_generator[0][1]
```

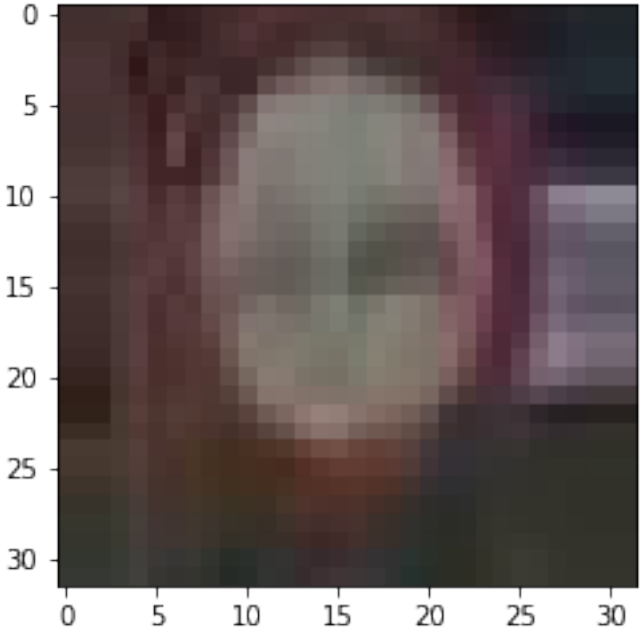
```
Found 7489 images belonging to 59 classes.
Found 1159 images belonging to 59 classes.
```

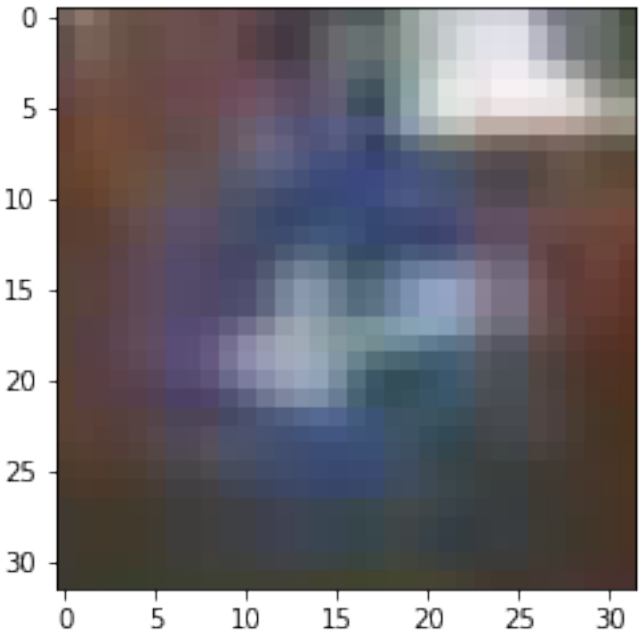
[4]:
```
#Prima stampa di prova per vedere se l'import delle immagini funziona
for i in range (0,8):
    image = X_train[i]
    plt.imshow(image)
    plt.show()
```

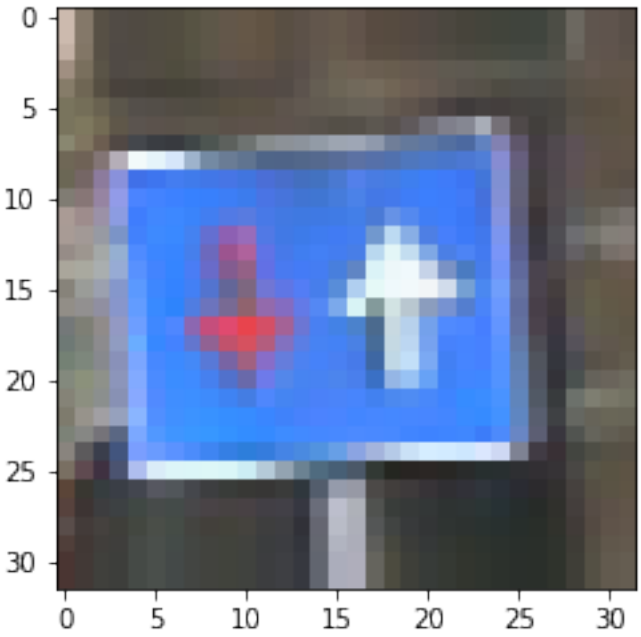

0
5
10
15
20
25
30
0
5
10
15
20
25
30

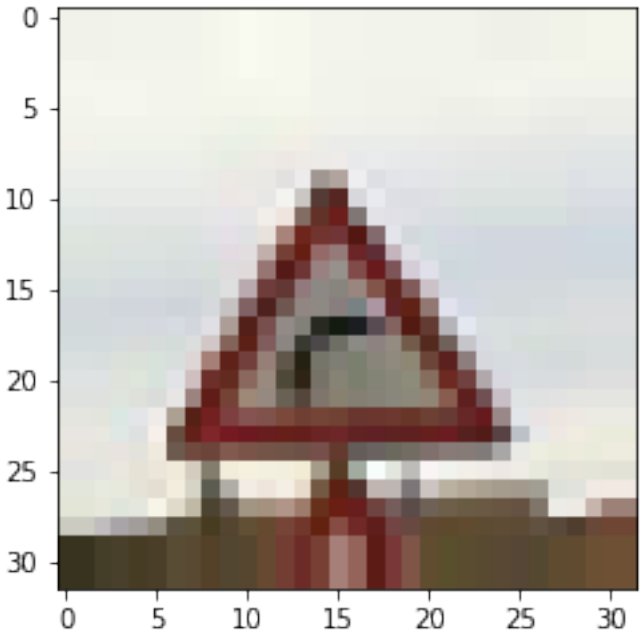

0
5
10
15
20
25
30
0
5
10
15
20
25
30

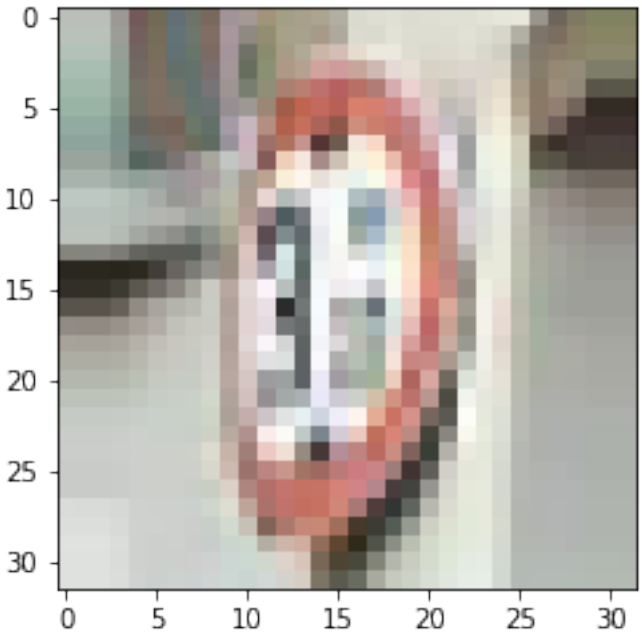

0
5
10
15
20
25
30
0
5
10
15
20
25
30


0
5
10
15
20
25
30
0
5
10
15
20
25
30


0
5
10
15
20
25
30
0
5
10
15
20
25
30


0
5
10
15
20
25
30
0
5
10
15
20
25
30





[5]:
```
#Operazioni la stampa finale
y_test = np.empty(len(Y_test),dtype=int)
for j in range(len(Y_test)):
    #print(j, "/", len(Y_test), end='\r')
```

```
    y_test[j]=(np.where(Y_test[j]==1)[0][0])

#Quest'ultima cosa serve per la stampa finale
immaginiPerLaStampaFinale=X_test
```

[6]:
```
#Stampa di alcuni esempi con le relative digits
for i in range(9):
  plt.subplot(3,3,i+1)
  plt.tight_layout()
  plt.imshow(X_train[i], interpolation='none')
  plt.title("Digit: {}".format(np.where(Y_test[i]==1)[0][0]))
  plt.xticks([])
  plt.yticks([])
plt.show()
```

[7]:
```
# Stampa delle shapes prima della reshapes e prima dell normalizzazione
print("X_train shape", X_train.shape)
print("y_train shape", Y_train.shape)
print("X_test shape", X_test.shape)
print("y_test shape", Y_test.shape)

# Creando il vettore unico di dimensione img_width*img_height*pixels
X_train = X_train.reshape(7489, img_width*img_height*3)
X_test = X_test.reshape(1159, img_width*img_height*3)
```

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# normalizzazione per velocizzare il training
X_train /= 255
X_test /= 255

# Stampa delle shapes finali
print("Train matrix shape", X_train.shape)
print("Test matrix shape", X_test.shape)
```

```
X_train shape (7489, 32, 32, 3)
y_train shape (7489, 59)
X_test shape (1159, 32, 32, 3)
y_test shape (1159, 59)
Train matrix shape (7489, 3072)
Test matrix shape (1159, 3072)
```

[8]:
```
# building a linear stack of layers with the sequential model
modelloCreatoDaMe = Sequential()
modelloCreatoDaMe.add(Dense(256, input_shape=(X_test.shape[1],)))
modelloCreatoDaMe.add(Activation('relu'))
modelloCreatoDaMe.add(Dropout(0.2))

modelloCreatoDaMe.add(Dense(256))
modelloCreatoDaMe.add(Activation('relu'))
modelloCreatoDaMe.add(Dropout(0.2))

modelloCreatoDaMe.add(Dense(59))
modelloCreatoDaMe.add(Activation('softmax'))

modelloCreatoDaMe.
 ↪compile(loss='categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
```

[9]:
```
history = modelloCreatoDaMe.fit(X_train, Y_train,
                                batch_size=batch_size,
                                epochs=epoche,
                                verbose=2,
                                validation_data=(X_test, Y_test))
```

```
Train on 7489 samples, validate on 1159 samples
Epoch 1/200
 - 3s - loss: 3.7677 - accuracy: 0.0698 - val_loss: 3.8945 - val_accuracy:
0.0613
Epoch 2/200
 - 2s - loss: 3.5898 - accuracy: 0.0888 - val_loss: 3.8618 - val_accuracy:
0.0630
```

```
Epoch 3/200
 - 2s - loss: 3.4852 - accuracy: 0.1327 - val_loss: 3.7655 - val_accuracy:
0.1182
Epoch 4/200
 - 2s - loss: 3.2044 - accuracy: 0.1640 - val_loss: 3.5517 - val_accuracy:
0.1199
Epoch 5/200
 - 3s - loss: 2.9572 - accuracy: 0.2015 - val_loss: 3.4329 - val_accuracy:
0.1484
Epoch 6/200
 - 2s - loss: 2.8089 - accuracy: 0.2206 - val_loss: 3.3298 - val_accuracy:
0.1199
Epoch 7/200
 - 2s - loss: 2.6950 - accuracy: 0.2402 - val_loss: 3.2709 - val_accuracy:
0.1501
Epoch 8/200
 - 2s - loss: 2.5768 - accuracy: 0.2696 - val_loss: 3.1992 - val_accuracy:
0.1786
Epoch 9/200
 - 2s - loss: 2.4570 - accuracy: 0.3031 - val_loss: 3.1093 - val_accuracy:
0.2114
Epoch 10/200
 - 2s - loss: 2.3468 - accuracy: 0.3326 - val_loss: 3.0597 - val_accuracy:
0.1941
Epoch 11/200
 - 2s - loss: 2.2248 - accuracy: 0.3665 - val_loss: 2.9844 - val_accuracy:
0.2476
Epoch 12/200
 - 2s - loss: 2.1320 - accuracy: 0.3906 - val_loss: 2.9219 - val_accuracy:
0.2442
Epoch 13/200
 - 2s - loss: 2.0411 - accuracy: 0.4123 - val_loss: 2.8951 - val_accuracy:
0.2554
Epoch 14/200
 - 2s - loss: 1.9624 - accuracy: 0.4261 - val_loss: 2.8541 - val_accuracy:
0.2355
Epoch 15/200
 - 2s - loss: 1.8942 - accuracy: 0.4426 - val_loss: 2.8721 - val_accuracy:
0.2381
Epoch 16/200
 - 2s - loss: 1.8275 - accuracy: 0.4591 - val_loss: 2.8050 - val_accuracy:
0.2683
Epoch 17/200
 - 2s - loss: 1.7670 - accuracy: 0.4684 - val_loss: 2.7590 - val_accuracy:
0.2770
Epoch 18/200
 - 2s - loss: 1.7055 - accuracy: 0.4878 - val_loss: 2.7653 - val_accuracy:
0.2675
```

```
Epoch 19/200
 - 2s - loss: 1.6701 - accuracy: 0.4941 - val_loss: 2.6723 - val_accuracy:
0.2744
Epoch 20/200
 - 1s - loss: 1.6132 - accuracy: 0.5136 - val_loss: 2.7337 - val_accuracy:
0.3089
Epoch 21/200
 - 2s - loss: 1.5762 - accuracy: 0.5132 - val_loss: 2.6464 - val_accuracy:
0.3028
Epoch 22/200
 - 1s - loss: 1.5279 - accuracy: 0.5298 - val_loss: 2.6679 - val_accuracy:
0.3115
Epoch 23/200
 - 1s - loss: 1.4859 - accuracy: 0.5433 - val_loss: 2.7088 - val_accuracy:
0.3028
Epoch 24/200
 - 1s - loss: 1.4668 - accuracy: 0.5476 - val_loss: 2.6404 - val_accuracy:
0.3132
Epoch 25/200
 - 2s - loss: 1.4274 - accuracy: 0.5580 - val_loss: 2.6156 - val_accuracy:
0.3020
Epoch 26/200
 - 2s - loss: 1.3962 - accuracy: 0.5610 - val_loss: 2.5990 - val_accuracy:
0.3322
Epoch 27/200
 - 1s - loss: 1.3732 - accuracy: 0.5674 - val_loss: 2.6637 - val_accuracy:
0.3348
Epoch 28/200
 - 2s - loss: 1.3333 - accuracy: 0.5799 - val_loss: 2.6019 - val_accuracy:
0.3382
Epoch 29/200
 - 2s - loss: 1.3115 - accuracy: 0.5850 - val_loss: 2.6008 - val_accuracy:
0.3279
Epoch 30/200
 - 2s - loss: 1.2964 - accuracy: 0.5894 - val_loss: 2.6595 - val_accuracy:
0.3348
Epoch 31/200
 - 3s - loss: 1.2653 - accuracy: 0.5986 - val_loss: 2.6628 - val_accuracy:
0.3330
Epoch 32/200
 - 2s - loss: 1.2429 - accuracy: 0.6003 - val_loss: 2.6290 - val_accuracy:
0.3451
Epoch 33/200
 - 1s - loss: 1.2271 - accuracy: 0.6062 - val_loss: 2.5961 - val_accuracy:
0.3477
Epoch 34/200
 - 2s - loss: 1.2057 - accuracy: 0.6170 - val_loss: 2.6434 - val_accuracy:
0.3607
```

```
Epoch 35/200
 - 2s - loss: 1.1776 - accuracy: 0.6241 - val_loss: 2.5785 - val_accuracy:
0.3546
Epoch 36/200
 - 2s - loss: 1.1618 - accuracy: 0.6307 - val_loss: 2.6014 - val_accuracy:
0.3770
Epoch 37/200
 - 2s - loss: 1.1503 - accuracy: 0.6316 - val_loss: 2.6308 - val_accuracy:
0.3589
Epoch 38/200
 - 2s - loss: 1.1228 - accuracy: 0.6385 - val_loss: 2.6066 - val_accuracy:
0.3719
Epoch 39/200
 - 2s - loss: 1.1058 - accuracy: 0.6415 - val_loss: 2.6554 - val_accuracy:
0.3779
Epoch 40/200
 - 2s - loss: 1.0893 - accuracy: 0.6498 - val_loss: 2.6352 - val_accuracy:
0.3822
Epoch 41/200
 - 2s - loss: 1.0837 - accuracy: 0.6546 - val_loss: 2.6007 - val_accuracy:
0.3710
Epoch 42/200
 - 1s - loss: 1.0377 - accuracy: 0.6631 - val_loss: 2.6646 - val_accuracy:
0.3822
Epoch 43/200
 - 1s - loss: 1.0386 - accuracy: 0.6692 - val_loss: 2.5871 - val_accuracy:
0.3960
Epoch 44/200
 - 1s - loss: 1.0157 - accuracy: 0.6640 - val_loss: 2.6249 - val_accuracy:
0.3857
Epoch 45/200
 - 2s - loss: 1.0129 - accuracy: 0.6675 - val_loss: 2.6394 - val_accuracy:
0.3934
Epoch 46/200
 - 2s - loss: 0.9895 - accuracy: 0.6825 - val_loss: 2.6081 - val_accuracy:
0.3814
Epoch 47/200
 - 2s - loss: 0.9881 - accuracy: 0.6838 - val_loss: 2.6546 - val_accuracy:
0.3883
Epoch 48/200
 - 1s - loss: 0.9570 - accuracy: 0.6927 - val_loss: 2.6240 - val_accuracy:
0.3822
Epoch 49/200
 - 1s - loss: 0.9467 - accuracy: 0.6902 - val_loss: 2.5598 - val_accuracy:
0.4055
Epoch 50/200
 - 2s - loss: 0.9402 - accuracy: 0.6997 - val_loss: 2.6113 - val_accuracy:
0.4055
```

```
Epoch 51/200
 - 2s - loss: 0.9281 - accuracy: 0.6977 - val_loss: 2.6626 - val_accuracy:
0.4047
Epoch 52/200
 - 2s - loss: 0.9071 - accuracy: 0.7069 - val_loss: 2.6426 - val_accuracy:
0.4090
Epoch 53/200
 - 2s - loss: 0.8931 - accuracy: 0.7125 - val_loss: 2.6134 - val_accuracy:
0.4055
Epoch 54/200
 - 2s - loss: 0.8846 - accuracy: 0.7126 - val_loss: 2.6474 - val_accuracy:
0.4029
Epoch 55/200
 - 3s - loss: 0.8753 - accuracy: 0.7145 - val_loss: 2.6868 - val_accuracy:
0.4047
Epoch 56/200
 - 2s - loss: 0.8626 - accuracy: 0.7240 - val_loss: 2.6940 - val_accuracy:
0.4021
Epoch 57/200
 - 1s - loss: 0.8597 - accuracy: 0.7233 - val_loss: 2.7179 - val_accuracy:
0.4124
Epoch 58/200
 - 3s - loss: 0.8376 - accuracy: 0.7261 - val_loss: 2.7096 - val_accuracy:
0.4064
Epoch 59/200
 - 3s - loss: 0.8335 - accuracy: 0.7320 - val_loss: 2.6768 - val_accuracy:
0.4124
Epoch 60/200
 - 1s - loss: 0.8125 - accuracy: 0.7372 - val_loss: 2.6759 - val_accuracy:
0.4142
Epoch 61/200
 - 2s - loss: 0.8159 - accuracy: 0.7347 - val_loss: 2.6586 - val_accuracy:
0.4107
Epoch 62/200
 - 3s - loss: 0.8055 - accuracy: 0.7402 - val_loss: 2.6733 - val_accuracy:
0.4176
Epoch 63/200
 - 2s - loss: 0.7988 - accuracy: 0.7406 - val_loss: 2.7571 - val_accuracy:
0.4176
Epoch 64/200
 - 2s - loss: 0.7798 - accuracy: 0.7483 - val_loss: 2.7218 - val_accuracy:
0.4124
Epoch 65/200
 - 2s - loss: 0.7719 - accuracy: 0.7547 - val_loss: 2.7481 - val_accuracy:
0.4254
Epoch 66/200
 - 2s - loss: 0.7663 - accuracy: 0.7483 - val_loss: 2.7182 - val_accuracy:
0.4297
```

```
Epoch 67/200
 - 4s - loss: 0.7501 - accuracy: 0.7544 - val_loss: 2.7339 - val_accuracy:
0.4323
Epoch 68/200
 - 2s - loss: 0.7445 - accuracy: 0.7563 - val_loss: 2.7769 - val_accuracy:
0.4254
Epoch 69/200
 - 2s - loss: 0.7297 - accuracy: 0.7629 - val_loss: 2.7116 - val_accuracy:
0.4245
Epoch 70/200
 - 2s - loss: 0.7282 - accuracy: 0.7606 - val_loss: 2.7519 - val_accuracy:
0.4349
Epoch 71/200
 - 2s - loss: 0.7153 - accuracy: 0.7739 - val_loss: 2.7215 - val_accuracy:
0.4262
Epoch 72/200
 - 2s - loss: 0.7000 - accuracy: 0.7718 - val_loss: 2.7140 - val_accuracy:
0.4331
Epoch 73/200
 - 2s - loss: 0.6970 - accuracy: 0.7771 - val_loss: 2.7893 - val_accuracy:
0.4271
Epoch 74/200
 - 1s - loss: 0.7088 - accuracy: 0.7686 - val_loss: 2.7568 - val_accuracy:
0.4366
Epoch 75/200
 - 3s - loss: 0.6943 - accuracy: 0.7739 - val_loss: 2.7498 - val_accuracy:
0.4305
Epoch 76/200
 - 2s - loss: 0.6872 - accuracy: 0.7737 - val_loss: 2.8317 - val_accuracy:
0.4323
Epoch 77/200
 - 1s - loss: 0.6747 - accuracy: 0.7858 - val_loss: 2.7908 - val_accuracy:
0.4323
Epoch 78/200
 - 2s - loss: 0.6624 - accuracy: 0.7811 - val_loss: 2.7290 - val_accuracy:
0.4202
Epoch 79/200
 - 2s - loss: 0.6617 - accuracy: 0.7833 - val_loss: 2.8730 - val_accuracy:
0.4323
Epoch 80/200
 - 2s - loss: 0.6531 - accuracy: 0.7876 - val_loss: 2.8594 - val_accuracy:
0.4254
Epoch 81/200
 - 2s - loss: 0.6437 - accuracy: 0.7904 - val_loss: 2.8511 - val_accuracy:
0.4280
Epoch 82/200
 - 2s - loss: 0.6253 - accuracy: 0.7948 - val_loss: 2.8273 - val_accuracy:
0.4297
```

```
Epoch 83/200
 - 3s - loss: 0.6175 - accuracy: 0.8016 - val_loss: 2.8952 - val_accuracy:
0.4288
Epoch 84/200
 - 2s - loss: 0.6268 - accuracy: 0.7925 - val_loss: 2.9662 - val_accuracy:
0.4271
Epoch 85/200
 - 2s - loss: 0.6185 - accuracy: 0.7990 - val_loss: 2.8947 - val_accuracy:
0.4314
Epoch 86/200
 - 2s - loss: 0.6237 - accuracy: 0.7952 - val_loss: 2.8549 - val_accuracy:
0.4366
Epoch 87/200
 - 2s - loss: 0.6098 - accuracy: 0.8006 - val_loss: 3.0120 - val_accuracy:
0.4142
Epoch 88/200
 - 2s - loss: 0.6019 - accuracy: 0.8081 - val_loss: 2.8410 - val_accuracy:
0.4409
Epoch 89/200
 - 2s - loss: 0.5918 - accuracy: 0.8087 - val_loss: 2.9676 - val_accuracy:
0.4366
Epoch 90/200
 - 2s - loss: 0.5925 - accuracy: 0.8101 - val_loss: 2.9605 - val_accuracy:
0.4383
Epoch 91/200
 - 2s - loss: 0.5645 - accuracy: 0.8109 - val_loss: 2.9909 - val_accuracy:
0.4236
Epoch 92/200
 - 2s - loss: 0.5750 - accuracy: 0.8143 - val_loss: 2.9820 - val_accuracy:
0.4288
Epoch 93/200
 - 2s - loss: 0.5700 - accuracy: 0.8085 - val_loss: 2.9142 - val_accuracy:
0.4374
Epoch 94/200
 - 2s - loss: 0.5700 - accuracy: 0.8123 - val_loss: 2.8806 - val_accuracy:
0.4219
Epoch 95/200
 - 2s - loss: 0.5585 - accuracy: 0.8172 - val_loss: 2.9065 - val_accuracy:
0.4374
Epoch 96/200
 - 2s - loss: 0.5490 - accuracy: 0.8249 - val_loss: 3.0096 - val_accuracy:
0.4349
Epoch 97/200
 - 2s - loss: 0.5378 - accuracy: 0.8219 - val_loss: 2.9287 - val_accuracy:
0.4409
Epoch 98/200
 - 2s - loss: 0.5423 - accuracy: 0.8277 - val_loss: 2.9933 - val_accuracy:
0.4392
```

```
Epoch 99/200
 - 3s - loss: 0.5349 - accuracy: 0.8244 - val_loss: 2.9217 - val_accuracy:
0.4426
Epoch 100/200
 - 2s - loss: 0.5241 - accuracy: 0.8302 - val_loss: 2.9202 - val_accuracy:
0.4435
Epoch 101/200
 - 2s - loss: 0.5188 - accuracy: 0.8259 - val_loss: 2.9613 - val_accuracy:
0.4435
Epoch 102/200
 - 3s - loss: 0.5206 - accuracy: 0.8327 - val_loss: 2.9727 - val_accuracy:
0.4366
Epoch 103/200
 - 2s - loss: 0.5137 - accuracy: 0.8326 - val_loss: 3.0549 - val_accuracy:
0.4374
Epoch 104/200
 - 2s - loss: 0.5095 - accuracy: 0.8344 - val_loss: 2.9957 - val_accuracy:
0.4495
Epoch 105/200
 - 2s - loss: 0.5032 - accuracy: 0.8370 - val_loss: 3.0395 - val_accuracy:
0.4418
Epoch 106/200
 - 2s - loss: 0.4989 - accuracy: 0.8402 - val_loss: 3.0465 - val_accuracy:
0.4349
Epoch 107/200
 - 2s - loss: 0.4854 - accuracy: 0.8434 - val_loss: 3.0578 - val_accuracy:
0.4435
Epoch 108/200
 - 1s - loss: 0.4964 - accuracy: 0.8416 - val_loss: 3.0519 - val_accuracy:
0.4366
Epoch 109/200
 - 2s - loss: 0.4881 - accuracy: 0.8400 - val_loss: 3.0180 - val_accuracy:
0.4452
Epoch 110/200
 - 2s - loss: 0.4847 - accuracy: 0.8446 - val_loss: 3.1238 - val_accuracy:
0.4487
Epoch 111/200
 - 2s - loss: 0.4789 - accuracy: 0.8478 - val_loss: 3.1108 - val_accuracy:
0.4383
Epoch 112/200
 - 2s - loss: 0.4716 - accuracy: 0.8496 - val_loss: 2.9889 - val_accuracy:
0.4461
Epoch 113/200
 - 2s - loss: 0.4728 - accuracy: 0.8463 - val_loss: 3.0245 - val_accuracy:
0.4538
Epoch 114/200
 - 1s - loss: 0.4704 - accuracy: 0.8464 - val_loss: 3.0387 - val_accuracy:
0.4409
```

```
Epoch 115/200
 - 2s - loss: 0.4613 - accuracy: 0.8539 - val_loss: 3.1031 - val_accuracy:
0.4426
Epoch 116/200
 - 2s - loss: 0.4671 - accuracy: 0.8479 - val_loss: 3.1134 - val_accuracy:
0.4443
Epoch 117/200
 - 2s - loss: 0.4503 - accuracy: 0.8535 - val_loss: 3.1785 - val_accuracy:
0.4400
Epoch 118/200
 - 3s - loss: 0.4468 - accuracy: 0.8535 - val_loss: 3.2179 - val_accuracy:
0.4521
Epoch 119/200
 - 2s - loss: 0.4474 - accuracy: 0.8533 - val_loss: 3.1985 - val_accuracy:
0.4357
Epoch 120/200
 - 2s - loss: 0.4414 - accuracy: 0.8525 - val_loss: 3.1490 - val_accuracy:
0.4495
Epoch 121/200
 - 1s - loss: 0.4402 - accuracy: 0.8582 - val_loss: 3.1771 - val_accuracy:
0.4288
Epoch 122/200
 - 1s - loss: 0.4325 - accuracy: 0.8529 - val_loss: 3.1715 - val_accuracy:
0.4435
Epoch 123/200
 - 2s - loss: 0.4256 - accuracy: 0.8593 - val_loss: 3.1494 - val_accuracy:
0.4487
Epoch 124/200
 - 2s - loss: 0.4254 - accuracy: 0.8599 - val_loss: 3.2035 - val_accuracy:
0.4443
Epoch 125/200
 - 2s - loss: 0.4137 - accuracy: 0.8694 - val_loss: 3.2932 - val_accuracy:
0.4521
Epoch 126/200
 - 2s - loss: 0.4178 - accuracy: 0.8645 - val_loss: 3.2833 - val_accuracy:
0.4426
Epoch 127/200
 - 2s - loss: 0.4149 - accuracy: 0.8647 - val_loss: 3.2434 - val_accuracy:
0.4573
Epoch 128/200
 - 2s - loss: 0.3996 - accuracy: 0.8718 - val_loss: 3.2764 - val_accuracy:
0.4538
Epoch 129/200
 - 2s - loss: 0.4081 - accuracy: 0.8735 - val_loss: 3.3349 - val_accuracy:
0.4400
Epoch 130/200
 - 2s - loss: 0.4075 - accuracy: 0.8675 - val_loss: 3.2027 - val_accuracy:
0.4582
```

```
Epoch 131/200
 - 2s - loss: 0.3910 - accuracy: 0.8738 - val_loss: 3.2474 - val_accuracy:
0.4538
Epoch 132/200
 - 2s - loss: 0.4011 - accuracy: 0.8730 - val_loss: 3.1761 - val_accuracy:
0.4599
Epoch 133/200
 - 2s - loss: 0.3987 - accuracy: 0.8723 - val_loss: 3.3081 - val_accuracy:
0.4469
Epoch 134/200
 - 2s - loss: 0.3824 - accuracy: 0.8764 - val_loss: 3.3219 - val_accuracy:
0.4530
Epoch 135/200
 - 2s - loss: 0.3861 - accuracy: 0.8752 - val_loss: 3.2451 - val_accuracy:
0.4487
Epoch 136/200
 - 1s - loss: 0.3823 - accuracy: 0.8747 - val_loss: 3.3186 - val_accuracy:
0.4530
Epoch 137/200
 - 2s - loss: 0.3798 - accuracy: 0.8794 - val_loss: 3.3537 - val_accuracy:
0.4435
Epoch 138/200
 - 2s - loss: 0.3700 - accuracy: 0.8801 - val_loss: 3.2897 - val_accuracy:
0.4418
Epoch 139/200
 - 2s - loss: 0.3680 - accuracy: 0.8836 - val_loss: 3.3849 - val_accuracy:
0.4418
Epoch 140/200
 - 2s - loss: 0.3666 - accuracy: 0.8806 - val_loss: 3.3322 - val_accuracy:
0.4538
Epoch 141/200
 - 2s - loss: 0.3569 - accuracy: 0.8814 - val_loss: 3.2637 - val_accuracy:
0.4599
Epoch 142/200
 - 2s - loss: 0.3657 - accuracy: 0.8790 - val_loss: 3.3402 - val_accuracy:
0.4513
Epoch 143/200
 - 1s - loss: 0.3555 - accuracy: 0.8881 - val_loss: 3.3582 - val_accuracy:
0.4547
Epoch 144/200
 - 2s - loss: 0.3581 - accuracy: 0.8853 - val_loss: 3.4028 - val_accuracy:
0.4513
Epoch 145/200
 - 2s - loss: 0.3444 - accuracy: 0.8920 - val_loss: 3.4239 - val_accuracy:
0.4469
Epoch 146/200
 - 2s - loss: 0.3542 - accuracy: 0.8858 - val_loss: 3.4288 - val_accuracy:
0.4452
```

```
Epoch 147/200
 - 2s - loss: 0.3516 - accuracy: 0.8878 - val_loss: 3.3173 - val_accuracy:
0.4530
Epoch 148/200
 - 2s - loss: 0.3524 - accuracy: 0.8866 - val_loss: 3.3987 - val_accuracy:
0.4504
Epoch 149/200
 - 2s - loss: 0.3382 - accuracy: 0.8916 - val_loss: 3.3928 - val_accuracy:
0.4513
Epoch 150/200
 - 2s - loss: 0.3350 - accuracy: 0.8886 - val_loss: 3.4303 - val_accuracy:
0.4504
Epoch 151/200
 - 2s - loss: 0.3449 - accuracy: 0.8890 - val_loss: 3.3883 - val_accuracy:
0.4582
Epoch 152/200
 - 2s - loss: 0.3358 - accuracy: 0.8968 - val_loss: 3.3886 - val_accuracy:
0.4599
Epoch 153/200
 - 1s - loss: 0.3244 - accuracy: 0.8946 - val_loss: 3.3121 - val_accuracy:
0.4573
Epoch 154/200
 - 2s - loss: 0.3253 - accuracy: 0.8973 - val_loss: 3.3380 - val_accuracy:
0.4642
Epoch 155/200
 - 2s - loss: 0.3220 - accuracy: 0.8973 - val_loss: 3.5576 - val_accuracy:
0.4392
Epoch 156/200
 - 4s - loss: 0.3230 - accuracy: 0.8977 - val_loss: 3.4333 - val_accuracy:
0.4573
Epoch 157/200
 - 3s - loss: 0.3181 - accuracy: 0.8996 - val_loss: 3.3880 - val_accuracy:
0.4547
Epoch 158/200
 - 4s - loss: 0.3218 - accuracy: 0.8929 - val_loss: 3.4199 - val_accuracy:
0.4564
Epoch 159/200
 - 3s - loss: 0.3103 - accuracy: 0.8969 - val_loss: 3.4651 - val_accuracy:
0.4556
Epoch 160/200
 - 2s - loss: 0.3079 - accuracy: 0.9035 - val_loss: 3.4622 - val_accuracy:
0.4495
Epoch 161/200
 - 3s - loss: 0.3080 - accuracy: 0.9009 - val_loss: 3.6068 - val_accuracy:
0.4582
Epoch 162/200
 - 2s - loss: 0.3076 - accuracy: 0.9027 - val_loss: 3.4581 - val_accuracy:
0.4685
```

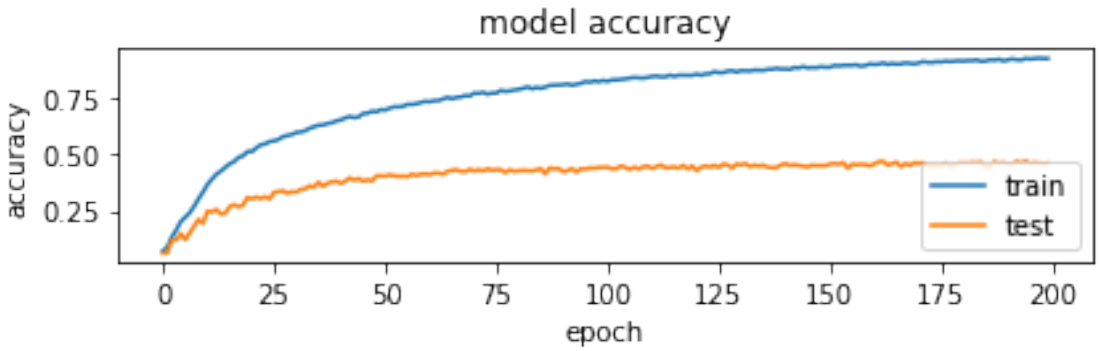
```
Epoch 163/200
 - 2s - loss: 0.2923 - accuracy: 0.9081 - val_loss: 3.5539 - val_accuracy:
0.4694
Epoch 164/200
 - 2s - loss: 0.3007 - accuracy: 0.9008 - val_loss: 3.5658 - val_accuracy:
0.4530
Epoch 165/200
 - 2s - loss: 0.2910 - accuracy: 0.9055 - val_loss: 3.4627 - val_accuracy:
0.4590
Epoch 166/200
 - 2s - loss: 0.2999 - accuracy: 0.9021 - val_loss: 3.4994 - val_accuracy:
0.4659
Epoch 167/200
 - 2s - loss: 0.2965 - accuracy: 0.9036 - val_loss: 3.4770 - val_accuracy:
0.4461
Epoch 168/200
 - 2s - loss: 0.2866 - accuracy: 0.9083 - val_loss: 3.4290 - val_accuracy:
0.4607
Epoch 169/200
 - 2s - loss: 0.2902 - accuracy: 0.9060 - val_loss: 3.5886 - val_accuracy:
0.4521
Epoch 170/200
 - 2s - loss: 0.2883 - accuracy: 0.9037 - val_loss: 3.4402 - val_accuracy:
0.4633
Epoch 171/200
 - 1s - loss: 0.2900 - accuracy: 0.9081 - val_loss: 3.6103 - val_accuracy:
0.4607
Epoch 172/200
 - 1s - loss: 0.2787 - accuracy: 0.9144 - val_loss: 3.5162 - val_accuracy:
0.4633
Epoch 173/200
 - 2s - loss: 0.2840 - accuracy: 0.9105 - val_loss: 3.6149 - val_accuracy:
0.4530
Epoch 174/200
 - 5s - loss: 0.2784 - accuracy: 0.9101 - val_loss: 3.5323 - val_accuracy:
0.4530
Epoch 175/200
 - 3s - loss: 0.2733 - accuracy: 0.9139 - val_loss: 3.5340 - val_accuracy:
0.4521
Epoch 176/200
 - 2s - loss: 0.2727 - accuracy: 0.9121 - val_loss: 3.7043 - val_accuracy:
0.4513
Epoch 177/200
 - 2s - loss: 0.2660 - accuracy: 0.9161 - val_loss: 3.4984 - val_accuracy:
0.4633
Epoch 178/200
 - 2s - loss: 0.2718 - accuracy: 0.9139 - val_loss: 3.4970 - val_accuracy:
0.4504
```

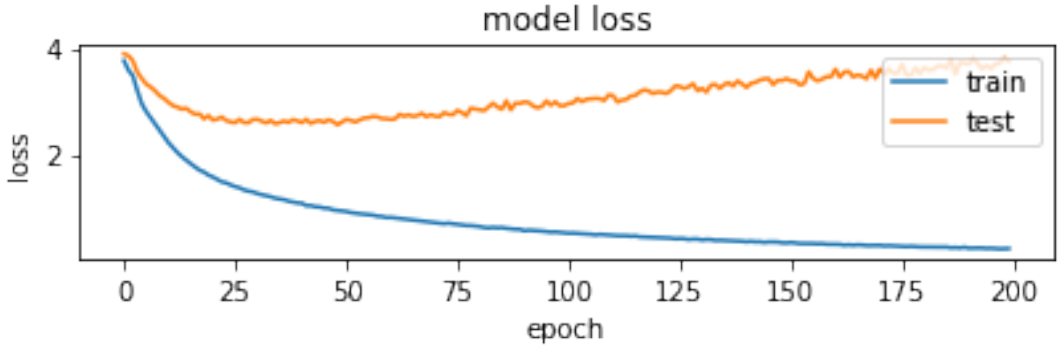
```
Epoch 179/200
 - 2s - loss: 0.2634 - accuracy: 0.9159 - val_loss: 3.6344 - val_accuracy:
0.4547
Epoch 180/200
 - 2s - loss: 0.2603 - accuracy: 0.9180 - val_loss: 3.5228 - val_accuracy:
0.4625
Epoch 181/200
 - 3s - loss: 0.2586 - accuracy: 0.9160 - val_loss: 3.6386 - val_accuracy:
0.4625
Epoch 182/200
 - 2s - loss: 0.2566 - accuracy: 0.9188 - val_loss: 3.5858 - val_accuracy:
0.4625
Epoch 183/200
 - 1s - loss: 0.2550 - accuracy: 0.9180 - val_loss: 3.6049 - val_accuracy:
0.4642
Epoch 184/200
 - 2s - loss: 0.2562 - accuracy: 0.9214 - val_loss: 3.6636 - val_accuracy:
0.4633
Epoch 185/200
 - 2s - loss: 0.2569 - accuracy: 0.9151 - val_loss: 3.6741 - val_accuracy:
0.4668
Epoch 186/200
 - 2s - loss: 0.2580 - accuracy: 0.9160 - val_loss: 3.6278 - val_accuracy:
0.4495
Epoch 187/200
 - 2s - loss: 0.2433 - accuracy: 0.9222 - val_loss: 3.7957 - val_accuracy:
0.4461
Epoch 188/200
 - 1s - loss: 0.2493 - accuracy: 0.9204 - val_loss: 3.5190 - val_accuracy:
0.4685
Epoch 189/200
 - 1s - loss: 0.2411 - accuracy: 0.9246 - val_loss: 3.7354 - val_accuracy:
0.4607
Epoch 190/200
 - 1s - loss: 0.2556 - accuracy: 0.9171 - val_loss: 3.7192 - val_accuracy:
0.4642
Epoch 191/200
 - 1s - loss: 0.2359 - accuracy: 0.9222 - val_loss: 3.7043 - val_accuracy:
0.4590
Epoch 192/200
 - 2s - loss: 0.2396 - accuracy: 0.9259 - val_loss: 3.8010 - val_accuracy:
0.4582
Epoch 193/200
 - 1s - loss: 0.2398 - accuracy: 0.9223 - val_loss: 3.7204 - val_accuracy:
0.4694
Epoch 194/200
 - 1s - loss: 0.2394 - accuracy: 0.9210 - val_loss: 3.6144 - val_accuracy:
0.4625
```

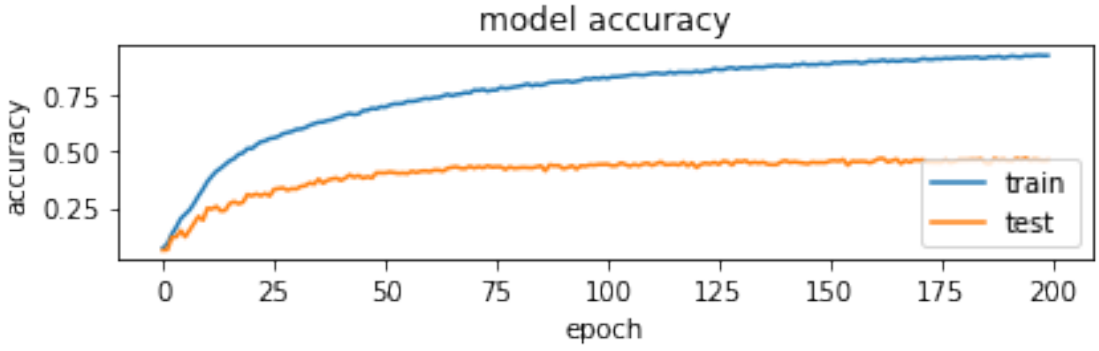
```
Epoch 195/200
 - 1s - loss: 0.2316 - accuracy: 0.9275 - val_loss: 3.6796 - val_accuracy:
0.4711
Epoch 196/200
 - 1s - loss: 0.2353 - accuracy: 0.9254 - val_loss: 3.6186 - val_accuracy:
0.4599
Epoch 197/200
 - 2s - loss: 0.2268 - accuracy: 0.9272 - val_loss: 3.7052 - val_accuracy:
0.4642
Epoch 198/200
 - 2s - loss: 0.2233 - accuracy: 0.9292 - val_loss: 3.7284 - val_accuracy:
0.4616
Epoch 199/200
 - 2s - loss: 0.2282 - accuracy: 0.9274 - val_loss: 3.8153 - val_accuracy:
0.4599
Epoch 200/200
 - 2s - loss: 0.2287 - accuracy: 0.9278 - val_loss: 3.7623 - val_accuracy:
0.4616
```

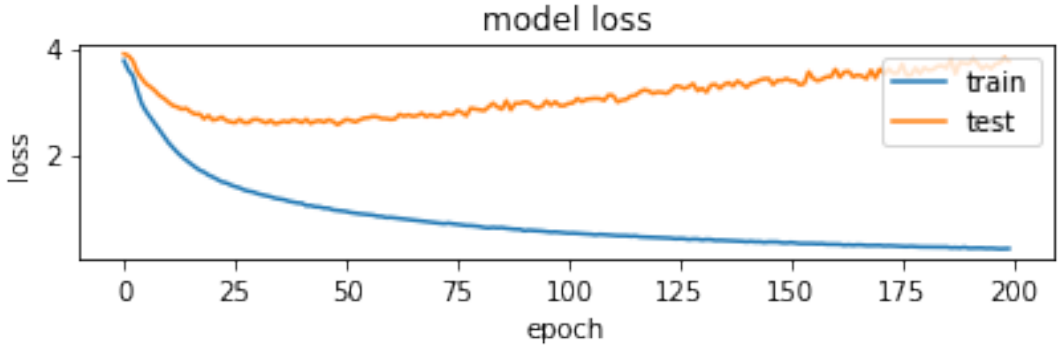
[10]:
```
# Stampa degli andamenti dell'accuracy e della loss sia del train che della␣
 ↪validation/test
fig = plt.figure()
plt.subplot(2,1,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='lower right')

plt.subplot(2,1,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper right')

plt.tight_layout()

fig
```

[10]:


model accuracy
accuracy
0.75
0.50
0.25
0
25
50
75
100
125
150
175
200
epoch
train
test


model loss
loss
4
2
0
25
50
75
100
125
150
175
200
epoch
train
test


model accuracy
accuracy
0.75
0.50
0.25
0
25
50
75
100
125
150
175
200
epoch
train
test


model loss
loss
4
2
0
25
50
75
100
125
150
175
200
epoch
train
test

[11]:
```
#Salvataggio dell'operato
save_dir = "results/"
model_name = 'HW5.h5'
model_path = os.path.join(save_dir, model_name)
modelloCreatoDaMe.save(model_path)
print('Saved trained model at %s ' % model_path)
```

```
Saved trained model at results/HW5.h5
```

[12]:
```
modello_cartelli = load_model("results/HW5.h5")
loss_and_metrics = modello_cartelli.evaluate(X_test, Y_test, verbose=2)

print("Test Loss", loss_and_metrics[0])
print("Test Accuracy", loss_and_metrics[1])
```

```
Test Loss 3.7622733317746286
Test Accuracy 0.4616048336029053
```

[13]:
```
# Creazione delle predizioni sul test set sulla base del modello caricato
carica_model = load_model('results/HW5.h5')
predicted_classes = carica_model.predict_classes(X_test)

# Distinguo cosa è stato predetto bene e cosa no
correct_indices = np.nonzero(predicted_classes == y_test)[0]
incorrect_indices = np.nonzero(predicted_classes != y_test)[0]
print()
print(len(correct_indices)," classified correctly")
print(len(incorrect_indices)," classified incorrectly")

# adapt figure size to accomodate 18 subplots
plt.rcParams['figure.figsize'] = (7,14)

figure_evaluation = plt.figure()

# Stampa delle 9 predizioni corrette
for i, correct in enumerate(correct_indices[:9]):
    plt.subplot(6,3,i+1)
    plt.imshow(immaginiPerLaStampaFinale[correct], interpolation='none')
    plt.title(
      "Predicted: {}, Truth: {}".format(predicted_classes[correct],
                                        y_test[correct]))
    plt.xticks([])
    plt.yticks([])

# Stampa delle 9 predizioni incorrette
for i, incorrect in enumerate(incorrect_indices[:9]):
    plt.subplot(6,3,i+10)
```

```
    plt.imshow(immaginiPerLaStampaFinale[incorrect], interpolation='none')
    plt.title(
      "Predicted {}, Truth: {}".format(predicted_classes[incorrect],
                                       y_test[incorrect]))
    plt.xticks([])
    plt.yticks([])

figure_evaluation
```

```
    535  classified correctly
    624  classified incorrectly
```

[13]:

Predicted: 11, Truth: 11 Predicted: 21, Truth: 21 Predicted: 27, Truth: 27

Predicted: 45, Truth: 45 Predicted: 16, Truth: 16 Predicted: 45, Truth: 45

Predicted: 21, Truth: 21 Predicted: 7, Truth: 7 Predicted: 34, Truth: 34

Predicted 49, Truth: 10 Predicted 46, Truth: 19 Predicted 27, Truth: 13

Predicted 11, Truth: 12 Predicted 44, Truth: 25 Predicted 25, Truth: 21

Predicted 19, Truth: 37 Predicted 33, Truth: 27 Predicted 19, Truth: 14

Predicted: 11, Truth: 11 Predicted: 21, Truth: 21 Predicted: 27, Truth: 27

Predicted: 45, Truth: 45 Predicted: 16, Truth: 16 Predicted: 45, Truth: 45

Predicted: 21, Truth: 21 Predicted: 7, Truth: 7 Predicted: 34, Truth: 34

Predicted 49, Truth: 10 Predicted 46, Truth: 19 Predicted 27, Truth: 13

Predicted 11, Truth: 12 Predicted 44, Truth: 25 Predicted 25, Truth: 21

Predicted 19, Truth: 37 Predicted 33, Truth: 27 Predicted 19, Truth: 14