RetoFinalHoteles

November 12, 2022

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import tensorflow as tf
import pandas as pd

from keras import models
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPooling2D, Dropout
from keras.metrics import TopKCategoricalAccuracy
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras import optimizers
from keras.applications.vgg16 import VGG16
from keras.applications.resnet_v2 import ResNet50V2
```

```
2022-11-12 00:31:45.150546: I
tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully
opened dynamic library libcudart.so.11.0
2022-11-12 00:31:45.868335: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot open shared
object file: No such file or directory; LD_LIBRARY_PATH:
/usr/local/cuda/lib64:/usr/local/nccl2/lib:/usr/local/cuda/extras/CUPTI/lib64
2022-11-12 00:31:45.868440: W
tensorflow/stream_executor/platform/default/dso_loader.cc:59] Could not load
dynamic library 'libnvinfer_plugin.so.7'; dlerror: libnvinfer_plugin.so.7:
cannot open shared object file: No such file or directory; LD_LIBRARY_PATH:
/usr/local/cuda/lib64:/usr/local/nccl2/lib:/usr/local/cuda/extras/CUPTI/lib64
2022-11-12 00:31:45.868450: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:30] Cannot dlopen some
TensorRT libraries. If you would like to use Nvidia GPU with TensorRT, please
make sure the missing libraries mentioned above are installed properly.
Using TensorFlow backend.
```

[3]:
```
train_dir = 'images'
test_dir = 'test'
```

```
dfTrain = pd.read_csv("trainSplit.csv").astype(str)
dfTest = pd.read_csv("validSplit.csv").astype(str)
```

[4]:
```
train_data = ImageDataGenerator(rescale=1/255, rotation_range=40,
                                width_shift_range=0.2,
                                height_shift_range=0.2,
                                shear_range=0.2,
                                zoom_range=0.2,
                                horizontal_flip=True,
                                fill_mode='nearest')

validation_data = ImageDataGenerator(rescale=1/255)


train_generator = train_data.flow_from_dataframe(dfTrain,
                                                 directory=train_dir,
                                                 y_col = "hotel_id",
                                                 x_col= "image_id",
                                                 target_size = (256, 256),
                                                 batch_size=128,
                                                 class_mode="categorical")

validation_generator = validation_data.flow_from_dataframe(dfTest,
                                                      directory=test_dir,
                                                      y_col = "hotel_id",
                                                      x_col= "image_id",
                                                      target_size = (256, 256),
                                                      batch_size=128,
                                                      class_mode="categorical")
```

```
Found 27127 validated image filenames belonging to 1376 classes.
Found 6178 validated image filenames belonging to 1376 classes.
```

1 MODEL METRICS (FUNCTIONS)

[10]:
```
def graphProgress(history):
    acc = history.history['top_k_categorical_accuracy']
    val_acc = history.history['val_top_k_categorical_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs = range(1, len(acc) + 1)
    plt.plot(epochs, acc, 'bo', label='Training accuracy')
    plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.legend()
    plt.figure()
```

```
    plt.plot(epochs, loss, 'bo', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()
    plt.show()

def runModel(model, epochs=30, steps=100):
    model.compile(loss='categorical_crossentropy', optimizer=tf.keras.
 ↪optimizers.RMSprop(learning_rate=2e-5),␣
 ↪metrics=[TopKCategoricalAccuracy(k=5)])
    #model.compile(optimizer=Adam(learning_rate=0.001),␣
 ↪loss='categorical_crossentropy', metrics=[TopKCategoricalAccuracy(k=5)])
    history = model.fit(train_generator, steps_per_epoch=steps, epochs=epochs,␣
 ↪validation_steps = 50, validation_data=validation_generator,verbose=1)
    graphProgress(history)
    return history
```

[11]:
```
new_base = ResNet50V2(weights='imagenet',
                  include_top=False,
                  input_shape=(256, 256, 3))
new_base.trainable = False
```

[13]:
```
model = models.Sequential([
                           new_base,
                           Flatten(),
                           Dense(512,activation='relu'),
                           Dense(1376, activation='softmax'),
                          ])
history = runModel(model,50,80)
model.save("DosAug")
```

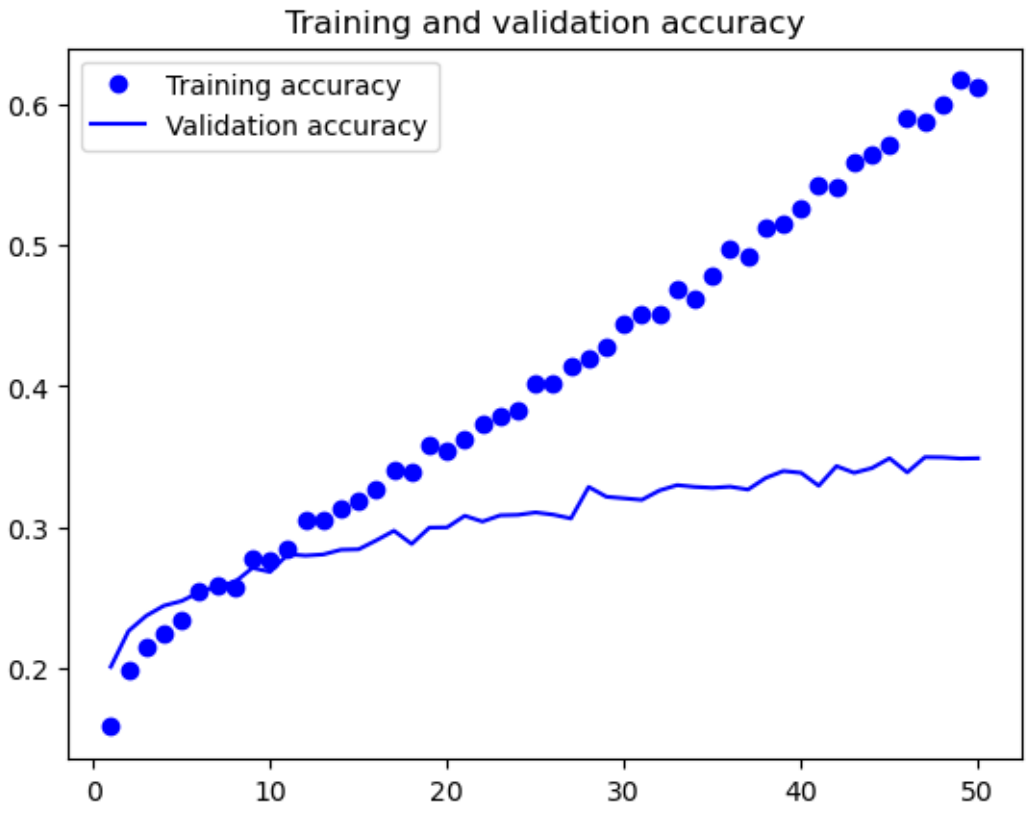
```
Epoch 1/50
80/80 [==============================] - 170s 2s/step - loss: 7.0062 -
top_k_categorical_accuracy: 0.1585 - val_loss: 6.8805 -
val_top_k_categorical_accuracy: 0.2011
Epoch 2/50
80/80 [==============================] - 151s 2s/step - loss: 6.7146 -
top_k_categorical_accuracy: 0.1981 - val_loss: 7.1903 -
val_top_k_categorical_accuracy: 0.2265
Epoch 3/50
80/80 [==============================] - 151s 2s/step - loss: 6.5486 -
top_k_categorical_accuracy: 0.2156 - val_loss: 6.8827 -
val_top_k_categorical_accuracy: 0.2372
Epoch 4/50
80/80 [==============================] - 150s 2s/step - loss: 6.3908 -
top_k_categorical_accuracy: 0.2248 - val_loss: 7.1277 -
val_top_k_categorical_accuracy: 0.2444
```

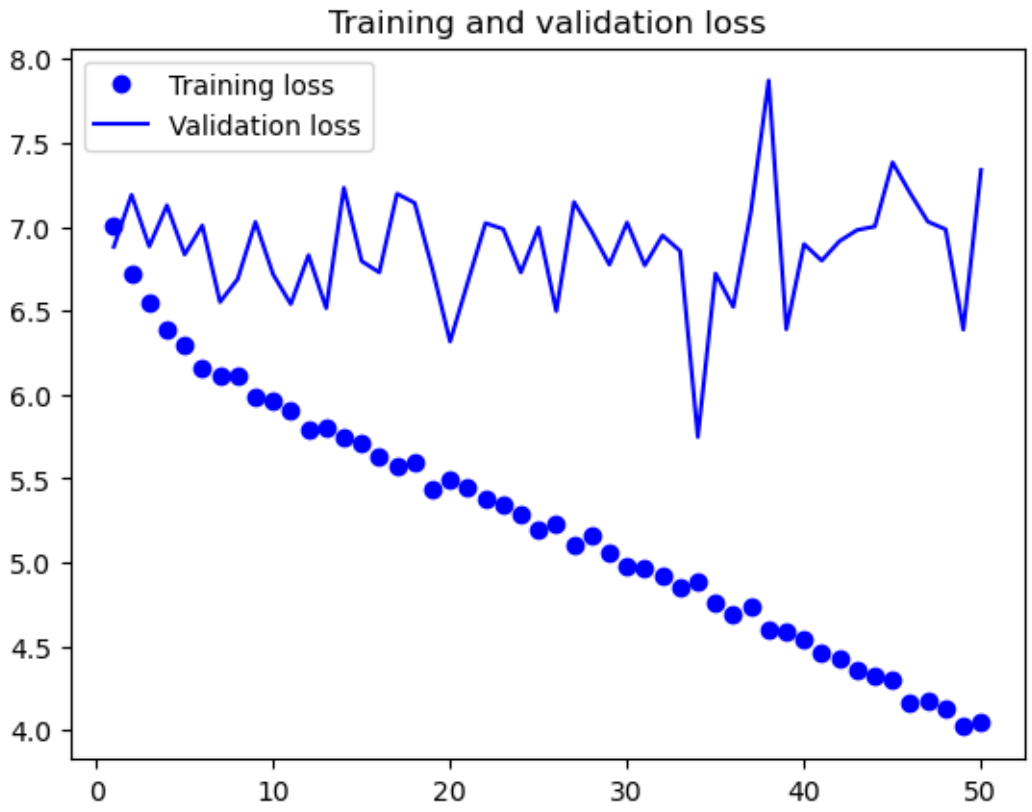
```
Epoch 5/50
80/80 [==============================] - 150s 2s/step - loss: 6.3009 -
top_k_categorical_accuracy: 0.2339 - val_loss: 6.8344 -
val_top_k_categorical_accuracy: 0.2477
Epoch 6/50
80/80 [==============================] - 149s 2s/step - loss: 6.1631 -
top_k_categorical_accuracy: 0.2541 - val_loss: 7.0077 -
val_top_k_categorical_accuracy: 0.2539
Epoch 7/50
80/80 [==============================] - 149s 2s/step - loss: 6.1129 -
top_k_categorical_accuracy: 0.2592 - val_loss: 6.5513 -
val_top_k_categorical_accuracy: 0.2583
Epoch 8/50
80/80 [==============================] - 151s 2s/step - loss: 6.1173 -
top_k_categorical_accuracy: 0.2567 - val_loss: 6.6885 -
val_top_k_categorical_accuracy: 0.2610
Epoch 9/50
80/80 [==============================] - 151s 2s/step - loss: 5.9807 -
top_k_categorical_accuracy: 0.2777 - val_loss: 7.0281 -
val_top_k_categorical_accuracy: 0.2712
Epoch 10/50
80/80 [==============================] - 149s 2s/step - loss: 5.9647 -
top_k_categorical_accuracy: 0.2759 - val_loss: 6.7162 -
val_top_k_categorical_accuracy: 0.2683
Epoch 11/50
80/80 [==============================] - 150s 2s/step - loss: 5.9076 -
top_k_categorical_accuracy: 0.2846 - val_loss: 6.5391 -
val_top_k_categorical_accuracy: 0.2808
Epoch 12/50
80/80 [==============================] - 150s 2s/step - loss: 5.7891 -
top_k_categorical_accuracy: 0.3055 - val_loss: 6.8309 -
val_top_k_categorical_accuracy: 0.2799
Epoch 13/50
80/80 [==============================] - 151s 2s/step - loss: 5.7976 -
top_k_categorical_accuracy: 0.3055 - val_loss: 6.5147 -
val_top_k_categorical_accuracy: 0.2807
Epoch 14/50
80/80 [==============================] - 150s 2s/step - loss: 5.7460 -
top_k_categorical_accuracy: 0.3135 - val_loss: 7.2328 -
val_top_k_categorical_accuracy: 0.2840
Epoch 15/50
80/80 [==============================] - 150s 2s/step - loss: 5.7056 -
top_k_categorical_accuracy: 0.3185 - val_loss: 6.7951 -
val_top_k_categorical_accuracy: 0.2843
Epoch 16/50
80/80 [==============================] - 150s 2s/step - loss: 5.6353 -
top_k_categorical_accuracy: 0.3273 - val_loss: 6.7278 -
val_top_k_categorical_accuracy: 0.2907
```

```
Epoch 17/50
80/80 [==============================] - 151s 2s/step - loss: 5.5686 -
top_k_categorical_accuracy: 0.3412 - val_loss: 7.1965 -
val_top_k_categorical_accuracy: 0.2975
Epoch 18/50
80/80 [==============================] - 150s 2s/step - loss: 5.5926 -
top_k_categorical_accuracy: 0.3390 - val_loss: 7.1416 -
val_top_k_categorical_accuracy: 0.2881
Epoch 19/50
80/80 [==============================] - 152s 2s/step - loss: 5.4399 -
top_k_categorical_accuracy: 0.3584 - val_loss: 6.7488 -
val_top_k_categorical_accuracy: 0.2997
Epoch 20/50
80/80 [==============================] - 150s 2s/step - loss: 5.4905 -
top_k_categorical_accuracy: 0.3539 - val_loss: 6.3157 -
val_top_k_categorical_accuracy: 0.2999
Epoch 21/50
80/80 [==============================] - 150s 2s/step - loss: 5.4473 -
top_k_categorical_accuracy: 0.3621 - val_loss: 6.6699 -
val_top_k_categorical_accuracy: 0.3081
Epoch 22/50
80/80 [==============================] - 151s 2s/step - loss: 5.3818 -
top_k_categorical_accuracy: 0.3733 - val_loss: 7.0213 -
val_top_k_categorical_accuracy: 0.3040
Epoch 23/50
80/80 [==============================] - 150s 2s/step - loss: 5.3446 -
top_k_categorical_accuracy: 0.3792 - val_loss: 6.9862 -
val_top_k_categorical_accuracy: 0.3084
Epoch 24/50
80/80 [==============================] - 150s 2s/step - loss: 5.2912 -
top_k_categorical_accuracy: 0.3823 - val_loss: 6.7279 -
val_top_k_categorical_accuracy: 0.3088
Epoch 25/50
80/80 [==============================] - 150s 2s/step - loss: 5.1976 -
top_k_categorical_accuracy: 0.4019 - val_loss: 6.9953 -
val_top_k_categorical_accuracy: 0.3105
Epoch 26/50
80/80 [==============================] - 150s 2s/step - loss: 5.2310 -
top_k_categorical_accuracy: 0.4018 - val_loss: 6.4966 -
val_top_k_categorical_accuracy: 0.3089
Epoch 27/50
80/80 [==============================] - 152s 2s/step - loss: 5.1060 -
top_k_categorical_accuracy: 0.4145 - val_loss: 7.1478 -
val_top_k_categorical_accuracy: 0.3062
Epoch 28/50
80/80 [==============================] - 150s 2s/step - loss: 5.1552 -
top_k_categorical_accuracy: 0.4194 - val_loss: 6.9713 -
val_top_k_categorical_accuracy: 0.3286
```

```
Epoch 29/50
80/80 [==============================] - 150s 2s/step - loss: 5.0609 -
top_k_categorical_accuracy: 0.4282 - val_loss: 6.7741 -
val_top_k_categorical_accuracy: 0.3216
Epoch 30/50
80/80 [==============================] - 155s 2s/step - loss: 4.9745 -
top_k_categorical_accuracy: 0.4443 - val_loss: 7.0248 -
val_top_k_categorical_accuracy: 0.3205
Epoch 31/50
80/80 [==============================] - 150s 2s/step - loss: 4.9636 -
top_k_categorical_accuracy: 0.4507 - val_loss: 6.7696 -
val_top_k_categorical_accuracy: 0.3194
Epoch 32/50
80/80 [==============================] - 149s 2s/step - loss: 4.9163 -
top_k_categorical_accuracy: 0.4513 - val_loss: 6.9482 -
val_top_k_categorical_accuracy: 0.3260
Epoch 33/50
80/80 [==============================] - 151s 2s/step - loss: 4.8541 -
top_k_categorical_accuracy: 0.4690 - val_loss: 6.8556 -
val_top_k_categorical_accuracy: 0.3298
Epoch 34/50
80/80 [==============================] - 150s 2s/step - loss: 4.8874 -
top_k_categorical_accuracy: 0.4618 - val_loss: 5.7473 -
val_top_k_categorical_accuracy: 0.3286
Epoch 35/50
80/80 [==============================] - 149s 2s/step - loss: 4.7550 -
top_k_categorical_accuracy: 0.4788 - val_loss: 6.7214 -
val_top_k_categorical_accuracy: 0.3279
Epoch 36/50
80/80 [==============================] - 150s 2s/step - loss: 4.6851 -
top_k_categorical_accuracy: 0.4978 - val_loss: 6.5222 -
val_top_k_categorical_accuracy: 0.3287
Epoch 37/50
80/80 [==============================] - 148s 2s/step - loss: 4.7381 -
top_k_categorical_accuracy: 0.4923 - val_loss: 7.0915 -
val_top_k_categorical_accuracy: 0.3267
Epoch 38/50
80/80 [==============================] - 158s 2s/step - loss: 4.6002 -
top_k_categorical_accuracy: 0.5125 - val_loss: 7.8718 -
val_top_k_categorical_accuracy: 0.3349
Epoch 39/50
80/80 [==============================] - 150s 2s/step - loss: 4.5877 -
top_k_categorical_accuracy: 0.5153 - val_loss: 6.3892 -
val_top_k_categorical_accuracy: 0.3398
Epoch 40/50
80/80 [==============================] - 151s 2s/step - loss: 4.5415 -
top_k_categorical_accuracy: 0.5261 - val_loss: 6.8964 -
val_top_k_categorical_accuracy: 0.3386
```

```
Epoch 41/50
80/80 [==============================] - 149s 2s/step - loss: 4.4661 -
top_k_categorical_accuracy: 0.5428 - val_loss: 6.7980 -
val_top_k_categorical_accuracy: 0.3294
Epoch 42/50
80/80 [==============================] - 151s 2s/step - loss: 4.4294 -
top_k_categorical_accuracy: 0.5415 - val_loss: 6.9140 -
val_top_k_categorical_accuracy: 0.3433
Epoch 43/50
80/80 [==============================] - 151s 2s/step - loss: 4.3551 -
top_k_categorical_accuracy: 0.5585 - val_loss: 6.9806 -
val_top_k_categorical_accuracy: 0.3386
Epoch 44/50
80/80 [==============================] - 150s 2s/step - loss: 4.3223 -
top_k_categorical_accuracy: 0.5647 - val_loss: 7.0013 -
val_top_k_categorical_accuracy: 0.3419
Epoch 45/50
80/80 [==============================] - 150s 2s/step - loss: 4.2989 -
top_k_categorical_accuracy: 0.5709 - val_loss: 7.3835 -
val_top_k_categorical_accuracy: 0.3489
Epoch 46/50
80/80 [==============================] - 162s 2s/step - loss: 4.1586 -
top_k_categorical_accuracy: 0.5905 - val_loss: 7.1993 -
val_top_k_categorical_accuracy: 0.3389
Epoch 47/50
80/80 [==============================] - 150s 2s/step - loss: 4.1775 -
top_k_categorical_accuracy: 0.5874 - val_loss: 7.0298 -
val_top_k_categorical_accuracy: 0.3497
Epoch 48/50
80/80 [==============================] - 149s 2s/step - loss: 4.1322 -
top_k_categorical_accuracy: 0.5994 - val_loss: 6.9844 -
val_top_k_categorical_accuracy: 0.3495
Epoch 49/50
80/80 [==============================] - 149s 2s/step - loss: 4.0205 -
top_k_categorical_accuracy: 0.6167 - val_loss: 6.3870 -
val_top_k_categorical_accuracy: 0.3487
Epoch 50/50
80/80 [==============================] - 149s 2s/step - loss: 4.0523 -
top_k_categorical_accuracy: 0.6119 - val_loss: 7.3385 -
val_top_k_categorical_accuracy: 0.3489
```


Training and validation accuracy
Training accuracy
Validation accuracy
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50




```
/opt/conda/lib/python3.7/site-packages/keras/engine/saving.py:165: UserWarning:
TensorFlow optimizers do not make it possible to access optimizer attributes or
optimizer state after instantiation. As a result, we cannot save the optimizer
as part of the model save file.You will have to compile your model again after
loading it. Prefer using a Keras optimizer instead (see keras.io/optimizers).
  'TensorFlow optimizers do not '
```

[8]:
```
print(history.params)
```

```
{'epochs': 50, 'steps': 80, 'verbose': 1, 'do_validation': True, 'metrics':
['loss', 'top_k_categorical_accuracy', 'val_loss',
'val_top_k_categorical_accuracy']}
```

[ ]:
```
#resumedModel = tf.keras.models.load_model('DosAug')
runModel(model,100,80)
```

```
Epoch 1/100
80/80 [==============================] - 172s 2s/step - loss: 3.9468 -
top_k_categorical_accuracy: 0.6292 - val_loss: 6.8109 -
val_top_k_categorical_accuracy: 0.3554
Epoch 2/100
80/80 [==============================] - 152s 2s/step - loss: 3.9007 -
```

```
top_k_categorical_accuracy: 0.6350 - val_loss: 7.3994 -
val_top_k_categorical_accuracy: 0.3524
Epoch 3/100
80/80 [==============================] - 153s 2s/step - loss: 3.8789 -
top_k_categorical_accuracy: 0.6381 - val_loss: 7.6895 -
val_top_k_categorical_accuracy: 0.3557
Epoch 4/100
80/80 [==============================] - 152s 2s/step - loss: 3.8114 -
top_k_categorical_accuracy: 0.6476 - val_loss: 7.1389 -
val_top_k_categorical_accuracy: 0.3509
Epoch 5/100
80/80 [==============================] - 152s 2s/step - loss: 3.7857 -
top_k_categorical_accuracy: 0.6553 - val_loss: 7.2293 -
val_top_k_categorical_accuracy: 0.3584
Epoch 6/100
80/80 [==============================] - 150s 2s/step - loss: 3.6994 -
top_k_categorical_accuracy: 0.6732 - val_loss: 6.6833 -
val_top_k_categorical_accuracy: 0.3614
Epoch 7/100
80/80 [==============================] - 152s 2s/step - loss: 3.6986 -
top_k_categorical_accuracy: 0.6683 - val_loss: 7.4191 -
val_top_k_categorical_accuracy: 0.3579
Epoch 8/100
80/80 [==============================] - 152s 2s/step - loss: 3.6675 -
top_k_categorical_accuracy: 0.6723 - val_loss: 7.2772 -
val_top_k_categorical_accuracy: 0.3693
Epoch 9/100
80/80 [==============================] - 152s 2s/step - loss: 3.5639 -
top_k_categorical_accuracy: 0.6916 - val_loss: 6.9336 -
val_top_k_categorical_accuracy: 0.3627
Epoch 10/100
80/80 [==============================] - 151s 2s/step - loss: 3.5797 -
top_k_categorical_accuracy: 0.6906 - val_loss: 6.8792 -
val_top_k_categorical_accuracy: 0.3696
Epoch 11/100
80/80 [==============================] - 153s 2s/step - loss: 3.4888 -
top_k_categorical_accuracy: 0.7002 - val_loss: 7.1987 -
val_top_k_categorical_accuracy: 0.3668
Epoch 12/100
80/80 [==============================] - 153s 2s/step - loss: 3.4569 -
top_k_categorical_accuracy: 0.7073 - val_loss: 7.2851 -
val_top_k_categorical_accuracy: 0.3611
Epoch 13/100
80/80 [==============================] - 150s 2s/step - loss: 3.4814 -
top_k_categorical_accuracy: 0.7052 - val_loss: 7.0473 -
val_top_k_categorical_accuracy: 0.3708
Epoch 14/100
80/80 [==============================] - 153s 2s/step - loss: 3.3697 -
```

```
top_k_categorical_accuracy: 0.7257 - val_loss: 7.8388 -
val_top_k_categorical_accuracy: 0.3689
Epoch 15/100
80/80 [==============================] - 153s 2s/step - loss: 3.3752 -
top_k_categorical_accuracy: 0.7243 - val_loss: 7.5770 -
val_top_k_categorical_accuracy: 0.3706
Epoch 16/100
80/80 [==============================] - 152s 2s/step - loss: 3.3601 -
top_k_categorical_accuracy: 0.7218 - val_loss: 6.9016 -
val_top_k_categorical_accuracy: 0.3720
Epoch 17/100
80/80 [==============================] - 151s 2s/step - loss: 3.2770 -
top_k_categorical_accuracy: 0.7369 - val_loss: 8.2138 -
val_top_k_categorical_accuracy: 0.3689
Epoch 18/100
80/80 [==============================] - 152s 2s/step - loss: 3.2705 -
top_k_categorical_accuracy: 0.7372 - val_loss: 7.6890 -
val_top_k_categorical_accuracy: 0.3668
Epoch 19/100
80/80 [==============================] - 152s 2s/step - loss: 3.2461 -
top_k_categorical_accuracy: 0.7433 - val_loss: 7.7810 -
val_top_k_categorical_accuracy: 0.3733
Epoch 20/100
80/80 [==============================] - 153s 2s/step - loss: 3.2001 -
top_k_categorical_accuracy: 0.7454 - val_loss: 7.9282 -
val_top_k_categorical_accuracy: 0.3712
Epoch 21/100
80/80 [==============================] - 153s 2s/step - loss: 3.2107 -
top_k_categorical_accuracy: 0.7489 - val_loss: 7.9185 -
val_top_k_categorical_accuracy: 0.3689
Epoch 22/100
80/80 [==============================] - 151s 2s/step - loss: 3.1182 -
top_k_categorical_accuracy: 0.7601 - val_loss: 7.9188 -
val_top_k_categorical_accuracy: 0.3763
Epoch 23/100
80/80 [==============================] - 152s 2s/step - loss: 3.1246 -
top_k_categorical_accuracy: 0.7575 - val_loss: 7.3658 -
val_top_k_categorical_accuracy: 0.3735
Epoch 24/100
80/80 [==============================] - 152s 2s/step - loss: 3.0828 -
top_k_categorical_accuracy: 0.7665 - val_loss: 7.6863 -
val_top_k_categorical_accuracy: 0.3796
Epoch 25/100
80/80 [==============================] - 151s 2s/step - loss: 3.0286 -
top_k_categorical_accuracy: 0.7748 - val_loss: 8.6804 -
val_top_k_categorical_accuracy: 0.3779
Epoch 26/100
80/80 [==============================] - 151s 2s/step - loss: 3.0634 -
```

```
top_k_categorical_accuracy: 0.7650 - val_loss: 7.0030 -
val_top_k_categorical_accuracy: 0.3800
Epoch 27/100
80/80 [==============================] - 152s 2s/step - loss: 2.9664 -
top_k_categorical_accuracy: 0.7766 - val_loss: 8.3708 -
val_top_k_categorical_accuracy: 0.3890
Epoch 28/100
80/80 [==============================] - 151s 2s/step - loss: 2.9467 -
top_k_categorical_accuracy: 0.7840 - val_loss: 7.9227 -
val_top_k_categorical_accuracy: 0.3787
Epoch 29/100
80/80 [==============================] - 150s 2s/step - loss: 3.0388 -
top_k_categorical_accuracy: 0.7723 - val_loss: 7.3303 -
val_top_k_categorical_accuracy: 0.3898
Epoch 30/100
80/80 [==============================] - 155s 2s/step - loss: 2.8873 -
top_k_categorical_accuracy: 0.7912 - val_loss: 7.3430 -
val_top_k_categorical_accuracy: 0.3784
Epoch 31/100
80/80 [==============================] - 151s 2s/step - loss: 2.8937 -
top_k_categorical_accuracy: 0.7878 - val_loss: 7.6946 -
val_top_k_categorical_accuracy: 0.3817
Epoch 32/100
80/80 [==============================] - 151s 2s/step - loss: 2.9382 -
top_k_categorical_accuracy: 0.7826 - val_loss: 8.0193 -
val_top_k_categorical_accuracy: 0.3841
Epoch 33/100
80/80 [==============================] - 152s 2s/step - loss: 2.8474 -
top_k_categorical_accuracy: 0.7996 - val_loss: 8.3228 -
val_top_k_categorical_accuracy: 0.3761
Epoch 34/100
80/80 [==============================] - 150s 2s/step - loss: 2.8536 -
top_k_categorical_accuracy: 0.7954 - val_loss: 7.8796 -
val_top_k_categorical_accuracy: 0.3930
Epoch 35/100
80/80 [==============================] - 151s 2s/step - loss: 2.7765 -
top_k_categorical_accuracy: 0.8056 - val_loss: 7.2327 -
val_top_k_categorical_accuracy: 0.3825
Epoch 36/100
80/80 [==============================] - 151s 2s/step - loss: 2.8150 -
top_k_categorical_accuracy: 0.8034 - val_loss: 8.6071 -
val_top_k_categorical_accuracy: 0.3852
Epoch 37/100
80/80 [==============================] - 151s 2s/step - loss: 2.8048 -
top_k_categorical_accuracy: 0.8010 - val_loss: 8.3490 -
val_top_k_categorical_accuracy: 0.3906
Epoch 38/100
80/80 [==============================] - 159s 2s/step - loss: 2.7558 -
```

```
top_k_categorical_accuracy: 0.8091 - val_loss: 7.9339 -
val_top_k_categorical_accuracy: 0.3877
Epoch 39/100
16/80 [=====>…] - ETA: 44s - loss: 2.6660 -
top_k_categorical_accuracy: 0.8179
```

[ ]: