```
# importing required libraries, modules and pretrained model

import os
import tensorflow as tf
from tensorflow import keras
from skimage import io
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPool2D, Dropout, Flatten, Ave
from tensorflow.keras.optimizers import RMSprop, SGD
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.applications.xception import Xception


SEED = 42
SIZE = (224, 224)
BATCH_SIZE = 32

pd.set_option('display.max_rows', None)


# training and validation data

from google.colab import files
uploaded = files.upload()







import io
labels = pd.read_csv(io.BytesIO(uploaded['horse breeds.csv']))


labels.head()









labels['id'] = labels['id'].apply(lambda x: x + '.png')
```

```
labels.head()
```

```
labels['breed'].value_counts().plot.bar(figsize=(16, 8))
```

```
# ImageDatagenerator to load the images in batches and perform data augmentation

data_generator = ImageDataGenerator(rescale= 1./255, validation_split=0.2, rotation_range=
                                    zoom_range=0.1, width_shift_range=0.2, height_shift_ra
                                    shear_range=0.1, horizontal_flip=True, fill_mode="near
```

```
train_generator = data_generator.flow_from_dataframe(labels, directory='/content/Images',
val_generator = data_generator.flow_from_dataframe(labels, directory='/content/Images', x_
```

```
Found 536 validated image filenames belonging to 7 classes.
Found 134 validated image filenames belonging to 7 classes.
```

```
# 12 images after augmentation

img, label = next(train_generator)

fig = plt.figure(figsize=(15, 10))

for i in range(12):
    fig.add_subplot(3, 4, i+1)
    plt.imshow(img[i])
    plt.axis('off')
```

```
# callbacks that will be used during training

early_stopping = EarlyStopping(monitor='val_loss', mode = 'min', patience=10)
checkpoint = ModelCheckpoint(filepath = './weights.hdf5', verbose=1, save_best_only=True)\


base_model = Xception(weights="imagenet", include_top=False, input_tensor=Input(shape=(224
```

```
base_model.summary()
```

```
alization)

add_10 (Add)                    (None, 14, 14, 728)  0           ['block12_sepcon
                                                                 'add_9[0][0]']

block13_sepconv1_act (Activati  (None, 14, 14, 728)  0          ['add_10[0][0]']
on)

block13_sepconv1 (SeparableCon  (None, 14, 14, 728)  536536     ['block13_sepcon
v2D)

block13_sepconv1_bn (BatchNorm  (None, 14, 14, 728)  2912       ['block13_sepcon
alization)

block13_sepconv2_act (Activati  (None, 14, 14, 728)  0          ['block13_sepcon
on)

block13_sepconv2 (SeparableCon  (None, 14, 14, 1024  752024     ['block13_sepcon
v2D)                           )

block13_sepconv2_bn (BatchNorm  (None, 14, 14, 1024  4096       ['block13_sepcon
alization)                     )

conv2d_3 (Conv2D)               (None, 7, 7, 1024)   745472      ['add_10[0][0]']

block13_pool (MaxPooling2D)     (None, 7, 7, 1024)   0           ['block13_sepcon

batch_normalization_3 (BatchNo  (None, 7, 7, 1024)  4096        ['conv2d_3[0][0]
rmalization)

add_11 (Add)                    (None, 7, 7, 1024)   0           ['block13_pool[0
                                                                 'batch_normaliz

block14_sepconv1 (SeparableCon  (None, 7, 7, 1536)  1582080     ['add_11[0][0]']
v2D)

block14_sepconv1_bn (BatchNorm  (None, 7, 7, 1536)  6144        ['block14_sepcon
alization)

block14_sepconv1_act (Activati  (None, 7, 7, 1536)  0           ['block14_sepcon
on)

block14_sepconv2 (SeparableCon  (None, 7, 7, 2048)  3159552     ['block14_sepcon
v2D)

block14_sepconv2_bn (BatchNorm  (None, 7, 7, 2048)  8192        ['block14_sepcon
alization)

block14_sepconv2_act (Activati  (None, 7, 7, 2048)  0           ['block14_sepcon
on)

==================================================================================
Total params: 20,861,480
Trainable params: 20,806,952
Non-trainable params: 54,528
__________________________________________________________________________________
```

```
for layer in base_model.layers:
    layer.trainable = False


head_model = AveragePooling2D(pool_size=(4, 4))(base_model.output)
head_model = Flatten(name='flatten')(head_model)
head_model = Dense(1024, activation='relu')(head_model)
head_model = Dropout(0.3)(head_model)
head_model = Dense(512, activation='relu')(head_model)
head_model = Dropout(0.3)(head_model)
head_model = Dense(7, activation='softmax')(head_model)


model = Model(inputs=base_model.input, outputs=head_model)
optimizer = SGD(learning_rate=0.1, momentum=0.9, decay=0.01)
model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
#model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=['categorical_ac


#first cycle

history1 = model.fit(train_generator, epochs=5, callbacks=[checkpoint], validation_data=va
```

```
Epoch 1/5
17/17 [==============================] - ETA: 0s - loss: 2.1739 - accuracy: 0.3638
Epoch 1: val_loss improved from inf to 2.50501, saving model to ./weights.hdf5
17/17 [==============================] - 129s 7s/step - loss: 2.1739 - accuracy: 0.3
Epoch 2/5
17/17 [==============================] - ETA: 0s - loss: 1.1716 - accuracy: 0.5896
Epoch 2: val_loss improved from 2.50501 to 2.11176, saving model to ./weights.hdf5
17/17 [==============================] - 126s 7s/step - loss: 1.1716 - accuracy: 0.5
Epoch 3/5
17/17 [==============================] - ETA: 0s - loss: 1.1627 - accuracy: 0.6119
Epoch 3: val_loss did not improve from 2.11176
17/17 [==============================] - 126s 7s/step - loss: 1.1627 - accuracy: 0.6
Epoch 4/5
17/17 [==============================] - ETA: 0s - loss: 1.0599 - accuracy: 0.6586
Epoch 4: val_loss did not improve from 2.11176
17/17 [==============================] - 124s 7s/step - loss: 1.0599 - accuracy: 0.6
Epoch 5/5
17/17 [==============================] - ETA: 0s - loss: 1.0217 - accuracy: 0.6586
Epoch 5: val_loss did not improve from 2.11176
17/17 [==============================] - 127s 8s/step - loss: 1.0217 - accuracy: 0.6
```

```
plt.plot(history1.history['accuracy'], label='training accuracy')
plt.plot(history1.history['val_accuracy'], label='validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Training and Validation Accuracy')
plt.legend(loc='lower right')
```

```
plt.plot(history1.history['loss'], label='training loss')
plt.plot(history1.history['val_loss'], label='validation loss')
plt.xlabel('Epochs')
plt.ylabel('Training and Validation Loss')
plt.legend()
```

```
#second cycle

for layer in base_model.layers[len(base_model.layers)//2:]:
    layer.trainable = True

optimizer = SGD(learning_rate=0.01, momentum=0.9, decay=0.001)
model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])


history2 = model.fit(train_generator, epochs=10, validation_data=val_generator, callbacks=
```

```
Epoch 1/10
17/17 [==============================] - ETA: 0s - loss: 0.9240 - accuracy: 0.7052
Epoch 1: val_loss improved from 2.11176 to 2.10266, saving model to ./weights.hdf5
17/17 [==============================] - 282s 17s/step - loss: 0.9240 - accuracy: 0.
Epoch 2/10
17/17 [==============================] - ETA: 0s - loss: 0.5914 - accuracy: 0.8134
Epoch 2: val_loss did not improve from 2.10266
17/17 [==============================] - 273s 16s/step - loss: 0.5914 - accuracy: 0.
Epoch 3/10
17/17 [==============================] - ETA: 0s - loss: 0.4818 - accuracy: 0.8153
```

```
Epoch 3: val_loss did not improve from 2.10266
17/17 [==============================] - 251s 15s/step - loss: 0.4818 - accuracy: 0.
Epoch 4/10
17/17 [==============================] - ETA: 0s - loss: 0.4170 - accuracy: 0.8601
Epoch 4: val_loss did not improve from 2.10266
17/17 [==============================] - 251s 15s/step - loss: 0.4170 - accuracy: 0.
Epoch 5/10
17/17 [==============================] - ETA: 0s - loss: 0.3055 - accuracy: 0.8918
Epoch 5: val_loss did not improve from 2.10266
17/17 [==============================] - 251s 15s/step - loss: 0.3055 - accuracy: 0.
Epoch 6/10
17/17 [==============================] - ETA: 0s - loss: 0.2024 - accuracy: 0.9328
Epoch 6: val_loss improved from 2.10266 to 1.80999, saving model to ./weights.hdf5
17/17 [==============================] - 253s 15s/step - loss: 0.2024 - accuracy: 0.
Epoch 7/10
17/17 [==============================] - ETA: 0s - loss: 0.2447 - accuracy: 0.9310
Epoch 7: val_loss did not improve from 1.80999
17/17 [==============================] - 256s 15s/step - loss: 0.2447 - accuracy: 0.
Epoch 8/10
17/17 [==============================] - ETA: 0s - loss: 0.2101 - accuracy: 0.9235
Epoch 8: val_loss did not improve from 1.80999
17/17 [==============================] - 253s 15s/step - loss: 0.2101 - accuracy: 0.
Epoch 9/10
17/17 [==============================] - ETA: 0s - loss: 0.1705 - accuracy: 0.9422
Epoch 9: val_loss did not improve from 1.80999
17/17 [==============================] - 250s 15s/step - loss: 0.1705 - accuracy: 0.
Epoch 10/10
17/17 [==============================] - ETA: 0s - loss: 0.1220 - accuracy: 0.9478
Epoch 10: val_loss did not improve from 1.80999
17/17 [==============================] - 255s 15s/step - loss: 0.1220 - accuracy: 0.
```

```
plt.plot(history2.history['accuracy'], label='training accuracy')
plt.plot(history2.history['val_accuracy'], label='validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Training and Validation Accuracy')
plt.legend(loc='lower right')
```

```
plt.plot(history2.history['loss'], label='training loss')
plt.plot(history2.history['val_loss'], label='validation loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Training and Validation Loss')
plt.legend()
```

The model clearly started overfitting after after couple of epochs so the best weights saved in the weigh.hdf5 file will be loaded before the third cycle.

```
model.load_weights('./weights.hdf5')
```

```
# third cycle

for layer in base_model.layers:
    layer.trainable = True

optimizer = SGD(learning_rate=0.01, momentum=0.9, decay=0.001)
model.compile(loss="categorical_crossentropy", optimizer=optimizer, metrics=["accuracy"])
```

```
history3 = model.fit(train_generator, epochs=100, validation_data=val_generator, callbacks
```

```
Epoch 1/100
17/17 [==============================] - ETA: 0s - loss: 0.2455 - accuracy: 0.908
Epoch 1: val_loss did not improve from 1.80999
17/17 [==============================] - 454s 26s/step - loss: 0.2455 - accuracy:
Epoch 2/100
17/17 [==============================] - ETA: 0s - loss: 0.2566 - accuracy: 0.919
Epoch 2: val_loss did not improve from 1.80999
17/17 [==============================] - 451s 27s/step - loss: 0.2566 - accuracy:
Epoch 3/100
17/17 [==============================] - ETA: 0s - loss: 0.1879 - accuracy: 0.949
Epoch 3: val_loss did not improve from 1.80999
17/17 [==============================] - 455s 27s/step - loss: 0.1879 - accuracy:
Epoch 4/100
17/17 [==============================] - ETA: 0s - loss: 0.1680 - accuracy: 0.932
Epoch 4: val_loss did not improve from 1.80999
17/17 [==============================] - 456s 27s/step - loss: 0.1680 - accuracy:
Epoch 5/100
17/17 [==============================] - ETA: 0s - loss: 0.1698 - accuracy: 0.934
```

```
Epoch 5: val_loss did not improve from 1.80999
17/17 [==============================] - 456s 27s/step - loss: 0.1698 - accuracy:
Epoch 6/100
17/17 [==============================] - ETA: 0s - loss: 0.1548 - accuracy: 0.951
Epoch 6: val_loss did not improve from 1.80999
17/17 [==============================] - 454s 27s/step - loss: 0.1548 - accuracy:
Epoch 7/100
17/17 [==============================] - ETA: 0s - loss: 0.1405 - accuracy: 0.953
Epoch 7: val_loss did not improve from 1.80999
17/17 [==============================] - 455s 27s/step - loss: 0.1405 - accuracy:
Epoch 8/100
17/17 [==============================] - ETA: 0s - loss: 0.1576 - accuracy: 0.964
Epoch 8: val_loss did not improve from 1.80999
17/17 [==============================] - 457s 27s/step - loss: 0.1576 - accuracy:
Epoch 9/100
17/17 [==============================] - ETA: 0s - loss: 0.1283 - accuracy: 0.964
Epoch 9: val_loss did not improve from 1.80999
17/17 [==============================] - 457s 27s/step - loss: 0.1283 - accuracy:
Epoch 10/100
17/17 [==============================] - ETA: 0s - loss: 0.1040 - accuracy: 0.966
Epoch 10: val_loss did not improve from 1.80999
17/17 [==============================] - 448s 26s/step - loss: 0.1040 - accuracy:
Epoch 11/100
17/17 [==============================] - ETA: 0s - loss: 0.0572 - accuracy: 0.979
Epoch 11: val_loss did not improve from 1.80999
17/17 [==============================] - 435s 26s/step - loss: 0.0572 - accuracy:
Epoch 12/100
17/17 [==============================] - ETA: 0s - loss: 0.1024 - accuracy: 0.979
Epoch 12: val_loss did not improve from 1.80999
17/17 [==============================] - 427s 25s/step - loss: 0.1024 - accuracy:
Epoch 13/100
17/17 [==============================] - ETA: 0s - loss: 0.0599 - accuracy: 0.979
Epoch 13: val_loss did not improve from 1.80999
17/17 [==============================] - 425s 25s/step - loss: 0.0599 - accuracy:
Epoch 14/100
17/17 [==============================] - ETA: 0s - loss: 0.0697 - accuracy: 0.977
Epoch 14: val_loss did not improve from 1.80999
17/17 [==============================] - 429s 25s/step - loss: 0.0697 - accuracy:
```

```
plt.plot(history3.history['accuracy'], label='training accuracy')
plt.plot(history3.history['val_accuracy'], label='validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Training and Validation Accuracy')
plt.legend(loc='lower right')
```

```
plt.plot(history3.history['loss'], label='training loss')
plt.plot(history3.history['val_loss'], label='validation loss')
plt.xlabel('Epochs')
plt.ylabel('Training and Validation Loss')
plt.legend()
```

The model performance did not improve at all so we will return to the best weights reached in the second cycle.

```
# loading the testset

test_images_files_names = os.listdir('/content/Images')
test_set = pd.DataFrame(test_images_files_names, columns=['id'])
test_set.head()
```

```
test_data_generator = ImageDataGenerator(rescale= 1./255)
test_generator = test_data_generator.flow_from_dataframe(test_set, directory='/content/Ima
```

```
Found 670 validated image filenames.
```

```
model.load_weights('./weights.hdf5')


y_prop = model.predict(test_generator)


results = pd.DataFrame(columns=["id"] + [*train_generator.class_indices.keys()])
results
```

id	Akhal-Teke	Appaloosa	Arabian	Friesian	Orlov Trotter	Percheron	Vladimir Heavy Draft

```
results["id"] = [os.path.splitext(file)[0] for file in os.listdir('/content/Images')]
results.head()
```

	id	Akhal-Teke	Appaloosa	Arabian	Friesian	Orlov Trotter	Percheron	Vladimir Heavy Draft
0	06_113	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	04_012	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	06_046	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	04_001	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
results[[*train_generator.class_indices.keys()]] = y_prop
results.head()

results.to_csv("results.csv",index=False)
```

✓ 0s completed at 14:12