animal_classification

August 15, 2024

[1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as td
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,␣
  ↪Dropout
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
```

[2]:

```
train_dir=r'D:\ml\Animal\data\train'
test_dir=r'D:\ml\Animal\data\test'
validation_data_dir=r'D:\ml\Animal\data\validation'
```

[3]:

```
train_para=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')
```

[4]:

```
train_generator=train_para.flow_from_directory(train_dir,
                                  target_size=(300,300),
                                  batch_size=50,
                                  class_mode='categorical',
                                  subset='training')
```

```
Found 13412 images belonging to 6 classes.
```

[5]:

```
val_data = train_para.flow_from_directory(
         validation_data_dir,
         target_size=(300, 300),
         batch_size=50,
         class_mode='categorical',
         shuffle=False)
```

```
Found 2549 images belonging to 6 classes.
```

[6]:
```
model=Sequential([
    Conv2D(32,(3,3), activation='relu', input_shape=(300,300,3)),
    MaxPooling2D((2,2)),

    Conv2D(64,(3,3), activation='relu', input_shape=(300,300,3)),
    MaxPooling2D((2,2)),

    Conv2D(128,(3,3), activation='relu', input_shape=(300,300,3)),
    MaxPooling2D((2,2)),

    Conv2D(128,(3,3), activation='relu', input_shape=(300,300,3)),
    MaxPooling2D((2,2)),

    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(train_generator.num_classes, activation='softmax')
])
```

[7]:
```
early_stopping=EarlyStopping(monitor='val_loss',␣
 ↪patience=5,verbose=2,restore_best_weights=True)
```

[8]:
```
model.compile(

    optimizer=Adam(learning_rate=0.001),

    loss='categorical_crossentropy',

    metrics=['Accuracy']

)
```

[9]:
```
history=model.fit(train_generator,
                 epochs=100,
                  batch_size=64,
                 validation_data=val_data,
                  callbacks=[early_stopping])
```

```
Epoch 1/100
269/269 [==============================] - 646s 2s/step - loss: 1.6042 -
Accuracy: 0.3469 - val_loss: 1.3375 - val_Accuracy: 0.4845
Epoch 2/100
269/269 [==============================] - 615s 2s/step - loss: 1.2729 -
Accuracy: 0.5194 - val_loss: 1.0661 - val_Accuracy: 0.6034
Epoch 3/100
269/269 [==============================] - 610s 2s/step - loss: 1.0687 -
```

```
Accuracy: 0.6118 - val_loss: 0.9452 - val_Accuracy: 0.6548
Epoch 4/100
269/269 [==============================] - 606s 2s/step - loss: 0.9734 -
Accuracy: 0.6439 - val_loss: 0.9467 - val_Accuracy: 0.6536
Epoch 5/100
269/269 [==============================] - 613s 2s/step - loss: 0.9045 -
Accuracy: 0.6770 - val_loss: 0.8642 - val_Accuracy: 0.6881
Epoch 6/100
269/269 [==============================] - 612s 2s/step - loss: 0.8492 -
Accuracy: 0.6988 - val_loss: 0.8523 - val_Accuracy: 0.6960
Epoch 7/100
269/269 [==============================] - 608s 2s/step - loss: 0.8111 -
Accuracy: 0.7088 - val_loss: 0.7921 - val_Accuracy: 0.7199
Epoch 8/100
269/269 [==============================] - 615s 2s/step - loss: 0.7641 -
Accuracy: 0.7302 - val_loss: 0.7309 - val_Accuracy: 0.7481
Epoch 9/100
269/269 [==============================] - 615s 2s/step - loss: 0.7406 -
Accuracy: 0.7369 - val_loss: 0.7961 - val_Accuracy: 0.7191
Epoch 10/100
269/269 [==============================] - 609s 2s/step - loss: 0.7150 -
Accuracy: 0.7443 - val_loss: 0.6875 - val_Accuracy: 0.7662
Epoch 11/100
269/269 [==============================] - 612s 2s/step - loss: 0.6920 -
Accuracy: 0.7592 - val_loss: 0.6821 - val_Accuracy: 0.7599
Epoch 12/100
269/269 [==============================] - 611s 2s/step - loss: 0.6746 -
Accuracy: 0.7630 - val_loss: 0.6757 - val_Accuracy: 0.7595
Epoch 13/100
269/269 [==============================] - 610s 2s/step - loss: 0.6712 -
Accuracy: 0.7657 - val_loss: 0.6896 - val_Accuracy: 0.7650
Epoch 14/100
269/269 [==============================] - 609s 2s/step - loss: 0.6404 -
Accuracy: 0.7745 - val_loss: 0.6098 - val_Accuracy: 0.7901
Epoch 15/100
269/269 [==============================] - 613s 2s/step - loss: 0.6317 -
Accuracy: 0.7768 - val_loss: 0.6860 - val_Accuracy: 0.7678
Epoch 16/100
269/269 [==============================] - 611s 2s/step - loss: 0.6013 -
Accuracy: 0.7913 - val_loss: 0.5862 - val_Accuracy: 0.7944
Epoch 17/100
269/269 [==============================] - 612s 2s/step - loss: 0.5882 -
Accuracy: 0.7955 - val_loss: 0.5900 - val_Accuracy: 0.7980
Epoch 18/100
269/269 [==============================] - 612s 2s/step - loss: 0.5684 -
Accuracy: 0.8036 - val_loss: 0.6788 - val_Accuracy: 0.7591
Epoch 19/100
269/269 [==============================] - 613s 2s/step - loss: 0.5731 -
```

```
Accuracy: 0.8027 - val_loss: 0.5888 - val_Accuracy: 0.7980
Epoch 20/100
269/269 [==============================] - 614s 2s/step - loss: 0.5570 -
Accuracy: 0.8065 - val_loss: 0.5979 - val_Accuracy: 0.7999
Epoch 21/100
269/269 [==============================] - 623s 2s/step - loss: 0.5465 -
Accuracy: 0.8110 - val_loss: 0.5859 - val_Accuracy: 0.8054
Epoch 22/100
269/269 [==============================] - 613s 2s/step - loss: 0.5345 -
Accuracy: 0.8170 - val_loss: 0.5811 - val_Accuracy: 0.7991
Epoch 23/100
269/269 [==============================] - 612s 2s/step - loss: 0.5260 -
Accuracy: 0.8204 - val_loss: 0.5856 - val_Accuracy: 0.8023
Epoch 24/100
269/269 [==============================] - 616s 2s/step - loss: 0.5295 -
Accuracy: 0.8169 - val_loss: 0.6399 - val_Accuracy: 0.7882
Epoch 25/100
269/269 [==============================] - 615s 2s/step - loss: 0.5072 -
Accuracy: 0.8244 - val_loss: 0.5207 - val_Accuracy: 0.8223
Epoch 26/100
269/269 [==============================] - 614s 2s/step - loss: 0.5006 -
Accuracy: 0.8277 - val_loss: 0.5805 - val_Accuracy: 0.8023
Epoch 27/100
269/269 [==============================] - 614s 2s/step - loss: 0.4982 -
Accuracy: 0.8313 - val_loss: 0.4985 - val_Accuracy: 0.8356
Epoch 28/100
269/269 [==============================] - 614s 2s/step - loss: 0.4905 -
Accuracy: 0.8337 - val_loss: 0.5880 - val_Accuracy: 0.8031
Epoch 29/100
269/269 [==============================] - 614s 2s/step - loss: 0.4865 -
Accuracy: 0.8318 - val_loss: 0.5054 - val_Accuracy: 0.8286
Epoch 30/100
269/269 [==============================] - 615s 2s/step - loss: 0.4689 -
Accuracy: 0.8390 - val_loss: 0.5231 - val_Accuracy: 0.8250
Epoch 31/100
269/269 [==============================] - 613s 2s/step - loss: 0.4615 -
Accuracy: 0.8435 - val_loss: 0.5382 - val_Accuracy: 0.8211
Epoch 32/100
269/269 [==============================] - ETA: 0s - loss: 0.4773 - Accuracy:
0.8334Restoring model weights from the end of the best epoch: 27.
269/269 [==============================] - 614s 2s/step - loss: 0.4773 -
Accuracy: 0.8334 - val_loss: 0.5015 - val_Accuracy: 0.8262
Epoch 32: early stopping
```

[14]:

```
model.save(r'D:\ml\Animal\data\animal-classi.h5')
```

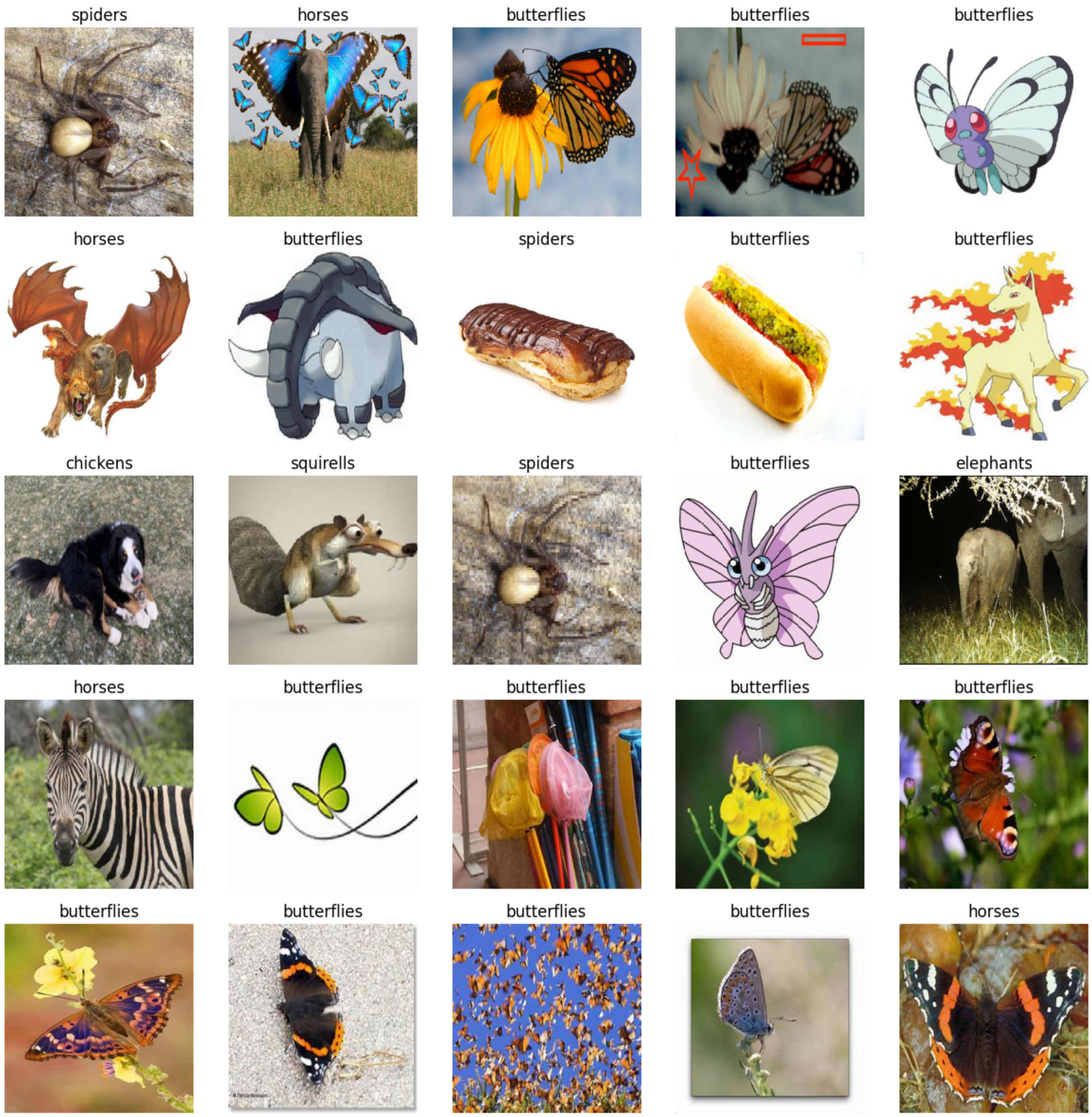
[15]:

```
import numpy as np
def predict(model, img):
    img_array = tf.keras.utils.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100*(np.max(predictions[0])), 0)
    return predicted_class, confidence
```

[27]:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np
import os
import matplotlib.pyplot as plt


model = tf.keras.models.load_model(r'D:\ml\Animal\data\animal-classi.h5')


class_labels = ['butterflies', 'chickens', 'elephants',␣
 ↪'horses','spiders','squirells']

def preprocess_image(image_path):
    img = load_img(image_path, target_size=(300, 300))
    img_array = img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = img_array / 255.0
    return img_array
test_images_dir = r'D:\ml\Animal\data\test'
test_images = []
for root, _, files in os.walk(test_images_dir):
    for file in files:
        if file.endswith(('.jpg', '.jpeg', '.png')):
            test_images.append(os.path.join(root, file))

plt.figure(figsize=(12, 12))
for i, image_path in enumerate(test_images[:25]):
    img_array = preprocess_image(image_path)
    prediction = model.predict(img_array)
    predicted_class = class_labels[np.argmax(prediction)]

    img = load_img(image_path, target_size=(300, 300))

    plt.subplot(5, 5, i + 1)
    plt.imshow(img)
    plt.title(predicted_class)
    plt.axis('off')

plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 0s 84ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 44ms/step
1/1 [==============================] - 0s 42ms/step
1/1 [==============================] - 0s 39ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 48ms/step
```


spiders
horses
butterflies
butterflies
butterflies
horses
butterflies
spiders
butterflies
butterflies
chickens
squirells
spiders
butterflies
elephants
horses
butterflies
butterflies
butterflies
butterflies
butterflies
butterflies
butterflies
butterflies
horses

[ ]:

[ ]: