Convolutional Networks

Task: To create a custom dataset for an image classification task, which will contain at least 5 different classes for classification (with a reasonable number of images for each class), choose 3 pre-trained models and retrain them to classify the created data.

Dataset

- The dataset was created manually by gathering images of 5 different classes: cats, dogs, birds, fish, and insects. A total of 100 images were collected for each class.
- The dataset was chosen for its simplicity and ease of understanding since it involves common animals and insects that can be easily recognized and classified.
- The dimensions of all the images are 224x224 pixels.
- Examples of images:

  Cats:

Dogs:

Birds:

Fish:

Insects:

- The images were loaded, converted to arrays, and preprocessed using the `**preprocess_input**` function from the TensorFlow library.

- The data augmentation used included rotation, width and height shift, zoom, and horizontal flip.

Models

The VGG16, VGG19, and ResNet50 models were used. The motivation behind the choice was to compare the two traditional serial models VGG16 and VGG19 with each other (to see if VGG19 is a stronger model than VGG16 due to a higher number of layers) and to compare them with a "network-in-network architectures" type model (ResNet50).

The original models were modified for classification purposes by replacing the last dense layers of the models with new layers. GlobalAveragePooling2D layers were added, along with Dense layers with 1024 neurons and ReLU activation, and Dense layers with 5 neurons and softmax activation.

Results

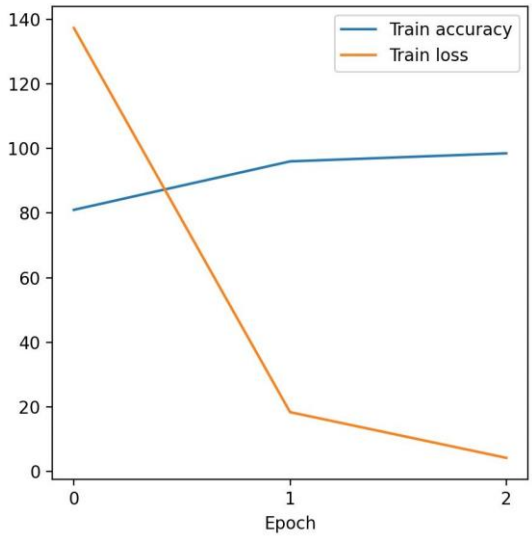
Model	Train accuracy	Test accuracy
VGG16	99.75%	97.00%
VGG19	99.25%	96.00%
ResNet50	99.75%	98.00%

In detail:

- VGG16:

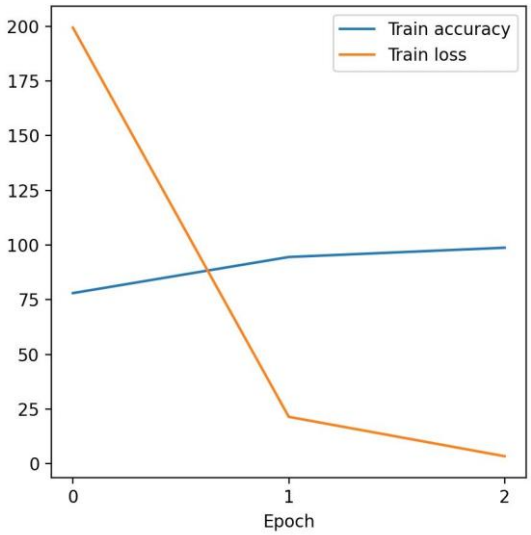
```
❯ python3 hw.py --train=True --model=vgg16            3m 24s
[INFO] loading vgg16...
[INFO] training the model...
Epoch 1/3
25/25 [==============================] - 49s 2s/step - loss: 1.3731 - accuracy:
0.8100
Epoch 2/3
25/25 [==============================] - 52s 2s/step - loss: 0.1836 - accuracy:
0.9600
Epoch 3/3
25/25 [==============================] - 55s 2s/step - loss: 0.0420 - accuracy:
0.9850
13/13 [==============================] - 50s 4s/step - loss: 0.0037 - accuracy:
0.9975
4/4 [==============================] - 13s 3s/step - loss: 0.2291 - accuracy:
0.9700
```

Train accuracy: 0.9975000023841858
Test accuracy: 0.9700000286102295



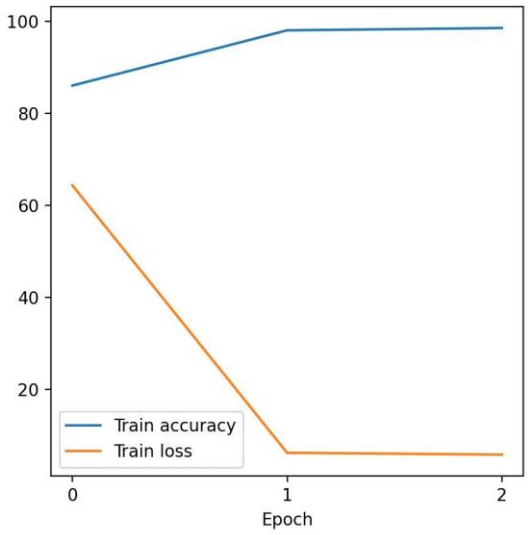
- VGG19:

❯ python3 hw.py --train=True --model=vgg19 1m 20s
[INFO] loading vgg19...
[INFO] training the model...
Epoch 1/3
25/25 [==============================] - 55s 2s/step - loss: 1.9936 - accuracy: 0.7800
Epoch 2/3
25/25 [==============================] - 68s 3s/step - loss: 0.2142 - accuracy: 0.9450
Epoch 3/3
25/25 [==============================] - 72s 3s/step - loss: 0.0341 - accuracy: 0.9875
13/13 [==============================] - 68s 5s/step - loss: 0.0086 - accuracy: 0.9925
4/4 [==============================] - 16s 4s/step - loss: 0.1301 - accuracy: 0.9600
Train accuracy: 0.9925000071525574
Test accuracy: 0.9599999785423279



- ResNet50:

❯ python3 hw.py --train=True --model=resnet 5m 1s
[INFO] loading resnet...
[INFO] training the model...
Epoch 1/3
25/25 [==============================] - 14s 499ms/step - loss: 0.6431 - accuracy: 0.8600
Epoch 2/3
25/25 [==============================] - 14s 547ms/step - loss: 0.0623 - accuracy: 0.9800
Epoch 3/3
25/25 [==============================] - 14s 574ms/step - loss: 0.0583 - accuracy: 0.9850
13/13 [==============================] - 15s 1s/step - loss: 0.0143 - accuracy: 0.9975
4/4 [==============================] - 3s 763ms/step - loss: 0.0771 - accuracy: 0.9800
Train accuracy: 0.9975000023841858
Test accuracy: 0.9800000190734863



Results indicate that all three models achieved high accuracy on both the training and test sets. The ResNet50 model demonstrated the best performance on the test set with an accuracy of 98%. The reason might be the deeper architecture of ResNet50 compared to the VGG models and the lesser number of parameters (ResNet50 has ~25.5 million parameters, while VGG16 has ~138 million parameters). As for VGG16 and VG19, the former model showed better accuracy. It might again be due to the latter having a larger number of parameters and therefore being less effective during the training process given the small dataset. When comparing the performance of the models, the factor of randomness must also not be forgotten. Overall, the results that have been achieved are quite good.

In conclusion, I will present an example of the models' performance on an image from the Internet: