Cameron Lee

5/8/2023

Rohit Kate

COMPSCI – 411; Assignment 4

Sea Animals Dataset

Sea Animals Used: Dolphin, Penguin, Sea Rays, Seal, Sharks

CNN Architectures Used

1. my_cnn1 was the same CNN architecture as in the A4.pptx. This architecture featured:
    a. Convolutional layer with 32 filters and a filter size of 3 x 3. Input shape parameter is of (100, 100, 3) representing the 100 x 100-pixel images and 3 channels of red, green, and blue. Activation always is ReLU.
    b. A pooling layer was added using a max function with a pooling window of size 2 x 2.
    c. Added a dropout layer with 20% of inputs equal to 0.
    d. Added another convolutional layer with 64 filters and a 3 x 3 filter size.
    e. Add another pooling layer using a max function with a pooling window size of 2 x 2.
    f. Dropout layer with 20% of inputs equal to 0.
    g. Flatten to get nodes from 2D layer to 1D layer.
    h. Add dense layer with 128 units and ReLU activation function.
    i. Dropout another 20% of inputs equal to 0.
    j. Add output layer with 5 nodes and softmax activation for classification task.
        i. 5 nodes since 5 different classes
2. my_cnn2 was a similar CNN architecture as my_cnn1. Its differences included:
    a. The first convolutional layer started with a larger filter size of 4 x 4.
    b. All dropout layers were changed to make 15% of inputs equal to 0.

Epochs and Training Accuracy

my_cnn1 Epochs used: 35

I used 35 epochs because the training accuracy decreased on epoch 34 and then rose again on 35 signaling a decline in overall accuracy. Ultimately, this may lead to an oscillation in accuracy signifying that the accuracy may not improve any further with additional epochs.

```
Epoch: 33 Training Accuracy: [0.9542190432548523]
Epoch: 34 Training Accuracy: [0.9515260457992554]
Epoch: 35 Training Accuracy: [0.9582585096359253]
```

```
>> import Assignment_4
Found 2785 files belonging to 5 classes.
Using 2228 files for training.
Found 2785 files belonging to 5 classes.
Using 2228 files for training.
Classes: ['Dolphin', 'Penguin', 'Sea Rays', 'Seal', 'Sharks']
Training:
Epoch: 1 Training Accuracy: [0.2859066426753998]
Epoch: 2 Training Accuracy: [0.37073609232902527]
Epoch: 3 Training Accuracy: [0.42369839549064636]
Epoch: 4 Training Accuracy: [0.4537701904773712]
Epoch: 5 Training Accuracy: [0.4753141701221466]
Epoch: 6 Training Accuracy: [0.49506282806396484]
Epoch: 7 Training Accuracy: [0.5201975107192993]
Epoch: 8 Training Accuracy: [0.5605924725532532]
Epoch: 9 Training Accuracy: [0.586624801158905]
Epoch: 10 Training Accuracy: [0.5991920828819275]
Epoch: 11 Training Accuracy: [0.631956934928894]
Epoch: 12 Training Accuracy: [0.6656193733215332]
Epoch: 13 Training Accuracy: [0.6831238865852356]
Epoch: 14 Training Accuracy: [0.7248653769493103]
Epoch: 15 Training Accuracy: [0.7495511770248413]
Epoch: 16 Training Accuracy: [0.7841113209724426]
Epoch: 17 Training Accuracy: [0.7962297797203064]
Epoch: 18 Training Accuracy: [0.8132854700088501]
Epoch: 19 Training Accuracy: [0.8254039287567139]
Epoch: 20 Training Accuracy: [0.8662477731704712]
Epoch: 21 Training Accuracy: [0.8662477731704712]
Epoch: 22 Training Accuracy: [0.8707360625267029]
Epoch: 23 Training Accuracy: [0.8940753936767578]
Epoch: 24 Training Accuracy: [0.9048473834991455]
Epoch: 25 Training Accuracy: [0.919658899307251]
Epoch: 26 Training Accuracy: [0.9219030737876892]
Epoch: 27 Training Accuracy: [0.9362657070159912]
Epoch: 28 Training Accuracy: [0.9317773580551147]
Epoch: 29 Training Accuracy: [0.9385098814964294]
Epoch: 30 Training Accuracy: [0.9398563504219055]
Epoch: 31 Training Accuracy: [0.9425493478775024]
Epoch: 32 Training Accuracy: [0.9479353427886963]
Epoch: 33 Training Accuracy: [0.9542190432548523]
Epoch: 34 Training Accuracy: [0.9515260457992554]
Epoch: 35 Training Accuracy: [0.9582585096359253]
Model: "sequential"
```

```
=================================================================
Total params: 1,732,421
Trainable params: 1,732,421
Non-trainable params: 0
_________________________________________________________________
Testing:
Test Accuracy: 0.9658886790275574
Saving the model in my_cnn1.h5
>>
```

my_cnn2 Epochs used: 30

I used 30 Epochs because there was a decrease in accuracy in the 29th Epoch. This signals that an additional number of epochs may not lead to any improvement.

```
import Assignment_4
Found 2785 files belonging to 5 classes.
Using 2228 files for training.
Found 2785 files belonging to 5 classes.
Using 2228 files for training.
Classes: ['Dolphin', 'Penguin', 'Sea Rays', 'Seal', 'Sharks']
Training:
Epoch: 1 Training Accuracy: [0.2814183235168457]
Epoch: 2 Training Accuracy: [0.35053861141204834]
Epoch: 3 Training Accuracy: [0.4052962362766266]
Epoch: 4 Training Accuracy: [0.45915618538856506]
Epoch: 5 Training Accuracy: [0.49506282806396484]
Epoch: 6 Training Accuracy: [0.5184021592140198]
Epoch: 7 Training Accuracy: [0.5394973158836365]
Epoch: 8 Training Accuracy: [0.5709156394004822]
Epoch: 9 Training Accuracy: [0.6090664267539978]
Epoch: 10 Training Accuracy: [0.64497309923172]
Epoch: 11 Training Accuracy: [0.6705565452575684]
Epoch: 12 Training Accuracy: [0.729353666305542]
Epoch: 13 Training Accuracy: [0.7562836408615112]
Epoch: 14 Training Accuracy: [0.8016157746315002]
Epoch: 15 Training Accuracy: [0.8240574598312378]
Epoch: 16 Training Accuracy: [0.8523339033126831]
Epoch: 17 Training Accuracy: [0.8684919476509094]
Epoch: 18 Training Accuracy: [0.884649932384491]
Epoch: 19 Training Accuracy: [0.906642735004425]
Epoch: 20 Training Accuracy: [0.9263913631439209]
Epoch: 21 Training Accuracy: [0.9214541912078857]
Epoch: 22 Training Accuracy: [0.9398563504219055]
Epoch: 23 Training Accuracy: [0.9376122355461121]
Epoch: 24 Training Accuracy: [0.9515260457992554]
Epoch: 25 Training Accuracy: [0.9614003300666809]
Epoch: 26 Training Accuracy: [0.9600538611412048]
Epoch: 27 Training Accuracy: [0.9596050381660461]
Epoch: 28 Training Accuracy: [0.9712746739387512]
Epoch: 29 Training Accuracy: [0.9672352075576782]
Epoch: 30 Training Accuracy: [0.9717234969139099]
Testing:
Test Accuracy: 0.9874326586723328
Saving the model in my_cnn2.h5
```

my_fine_tuned Epochs used: 12

I used 12 epochs because there was a decrease in the accuracy at epoch 12. Although there was a slight decrease from epoch 7 to 8, it was a very insignificant amount compared to the jump from epoch 11 to 12.

```
>>> ========== RESTART: Shell ==========
>>> import Assignment_4
Found 2785 files belonging to 5 classes.
Using 2228 files for training.
Found 2785 files belonging to 5 classes.
Using 557 files for validation.
Classes: ['Dolphin', 'Penguin', 'Sea Rays', 'Seal', 'Sharks']
Printing x: KerasTensor(type_spec=TensorSpec(shape=(None, None, None, 512), dtype=tf.float32, name=None), name='block5_pool/MaxPool:0', description="created by layer 'block5_pool'")
Training:
Epoch: 1 Training Accuracy: [0.5552064776420593]
Epoch: 2 Training Accuracy: [0.7226212024688721]
Epoch: 3 Training Accuracy: [0.830341100692749]
Epoch: 4 Training Accuracy: [0.9026032090187073]
Epoch: 5 Training Accuracy: [0.9322262406349182]
Epoch: 6 Training Accuracy: [0.9497306942939758]
Epoch: 7 Training Accuracy: [0.9582585096359253]
Epoch: 8 Training Accuracy: [0.9573608636856079]
Epoch: 9 Training Accuracy: [0.9712746739387512]
Epoch: 10 Training Accuracy: [0.9775583744049072]
Epoch: 11 Training Accuracy: [0.9815978407859802]
Epoch: 12 Training Accuracy: [0.9739676713943481]
Testing:
Test Accuracy: 0.6570915579795837
Saving the model in my_fine_tuned.h5
>>>
```

Accuracy Results on Test Data

Model	my_cnn1	my_cnn2	fine_tuned
Accuracy	0.965889	0.987433	0.657092

Table: According to the results of the models, the best CNN architecture was my_cnn2 with a testing accuracy of 0.987433. The worst testing architecture, fine_tuned, used the VGG16 function and it's testing accuracy was a measly 0.657092. Ultimately, the CCN 2 architecture outshined the rest of the neural networks and is the best CNN to classify these specific images with.

Image Predictions

Class Prediction Order:

```
Classes: ['Dolphin', 'Penguin', 'Sea Rays', 'Seal', 'Sharks']
```

Image 1 (Dolphin 1):

```
Image 1 (Dolphin)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 261ms/step
[[0.487 0.407 0.001 0.105 0.   ]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 232ms/step
[[1. 0. 0. 0. 0.]]
```

According to the results, the fine_tuned model was able to correctly predict the dolphin while the CNN model was about halfway between classifying it as a dolphin or a Penguin. The fine_tuned model was better for this image. I believe because of the shape the dolphin was making and the color scheme of the picture, there was a greater chance for error for the CNN architecture that was used.

Image 2 (Dolphin):

```
Image 2 (Dolphin)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 31ms/step
[[0.974 0.026 0.    0.001 0.   ]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 78ms/step
[[1. 0. 0. 0. 0.]]
```

The fine_tuned model was able to again correctly identify the dolphin. Unlike the first time around, the CNN model was able to almost completely classify this image as a dolphin with small bias towards the penguin again.

Dolphin Overall:

I believe that the CNN architecture model has trouble classifying a dolphin image without seeing the animal's face. The VGG16 architecture was completely spot on and shows how insanely good it is at classification.

Image 3 (Penguin):

```
Image 3 (Penguin)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 47ms/step
[[0. 1. 0. 0. 0.]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 78ms/step
[[0. 1. 0. 0. 0.]]
```

Both models were able to correctly identify the penguin.

Image 4 (Penguin):

```
Image 4 (Penguin)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 47ms/step
[[0.    0.972 0.    0.    0.027]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 69ms/step
[[0. 1. 0. 0. 0.]]
```

In this case, the CNN was strongly biased towards classifying this image as a penguin, but it still had some doubt on whether or not it was fully a penguin and gave a little bit of classification probability to a shark. The fine_tuned architecture was able to correctly identify the image as a penguin.

Penguin Overall:

Ultimately, the color schemes of the image for the CNN seem to be a recurring issue. If the colors from specific photos that the model trained on have not been seemed before it does not seem to be fully confident in making predictions.

Image 5 (Sea Ray):

```
Image 5 (Sea Ray)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 53ms/step
[[0.134 0.    0.001 0.    0.866]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 78ms/step
[[0. 0. 1. 0. 0.]]
```

The CNN architecture got this prediction completely wrong. It predicted that this image was a shark. Unfortunately, I believe this view of the sea ray can easily be confused with a shark view (swimming), so that is why it was misinterpreted. The fine_tuned model continues to accurately predict.

Image 6 (Sea Ray):

```
Image 6 (Sea Ray)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 94ms/step
[[0.009 0.958 0.032 0.001 0.   ]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 100ms/step
[[1. 0. 0. 0. 0.]]
```

This one was the worst outcome for both models combined. Clearly, the fine_tuned model has gotten the picture completely wrong classifying it as a dolphin. The CNN model also predicted this sea ray to be a penguin which literally makes no sense. I believe that the color and shape of the animal in the picture has fooled both models.

Sea Ray Overview:

Clearly it is important to choose several pictures from different angles. These pictures are harder to classify since they don't show the sea ray's face and it's body is swimming in a similar way to a dolphins.

Image 7 (Seal [Baby Seal]):

```
Image 7 (Seal)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 47ms/step
[[0.233 0.767 0.    0.    0.   ]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 116ms/step
[[0. 0. 0. 1. 0.]]
```

I put this picture in to try and trick both models, but it only tricked the CNN model. This predicted that the animal was a penguin. I think the landscape in the background and the shape of the baby can be similar to the penguin, which is why it was classified as such. As for the fine_tuned model, it was classified perfectly as a seal.

Image 8 (Seal):

```
Image 8 (Seal)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 22ms/step
[[0.    0.001 0.969 0.03  0.   ]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 78ms/step
[[0. 0. 0. 1. 0.]]
```

This model was not classified correctly again as a seal by the CNN architecture. The seals positioning and it's color is not usually of the same kind found on the training and testing set making it difficult for the model to predict. The fine_tuned model was able to accurately predict this animal as a seal.

Seal Overall:

The CNN architecture is very lacking for this category. The various shapes and colors have easily allowed the model to be fooled. The fine_tuned model is as accurate as ever.

Image 9 (Shark):

```
Image 9 (Shark)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 69ms/step
[[0.072 0.    0.    0.    0.928]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 69ms/step
[[0.    0.027 0.    0.    0.973]]
```

Both this models were not able to fully commit to classify this animal, but their majority classified it as a shark.

Image 10 (Shark):

```
Image 10 (Shark)
CNN 2 Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 54ms/step
[[1. 0. 0. 0. 0.]]
fine_tuned Model:
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 0s 62ms/step
[[0.244 0.    0.001 0.063 0.692]]
```

I purposely chose this very hard picture. According to the CNN model it was a dolphin (very incorrect). And the VGG16 model could not choose but gave majority classification to the shark.

Conclusion:

The CNN architecture is very lacking when it comes to more abstract images to be classified. It generally has to show a face, a common looking body, and a distinctive background to confidently classify an animal.

Unlike the CNN, the fine_tuned model showed its greatness after being tested with several hard images. The baby seal, and the shark picture with teeth were very difficult to expect good results, but they still resulted in high classification predictions.

If the CNN had more training/testing images I believe it would be able to be almost equal to the VGG16, but because it is so lacking it would take a very long time.