CS 4/510: Computer Vision & Deep Learning, Summer 2022

Programming Assignment #3

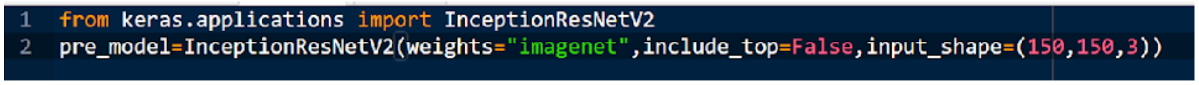
A. Rhodes

Note: This assignment is **due by Sunday, 7/31 at 1000pm**; you will turn in the assignment by email to our TA.

In this assignment you will apply transfer learning to a pre-trained CNN. For this exercise I encourage you to use Keras or a similar standard library for training NNs.

Dataset: Use the dataset I provide on Canvas. The dataset consists of color images of cats and dogs; there are 4,000 images of each category for training and 1,000 images of each category for testing.

Step 1: Load the pre-trained "InceptionResNetV2" model (https://keras.io/api/applications/inceptionresnetv2/). To do so in Keras, you will need to execute:



```
from keras.applications import InceptionResNetV2
pre_model=InceptionResNetV2(weights="imagenet",include_top=False,input_shape=(150,150,3))
```

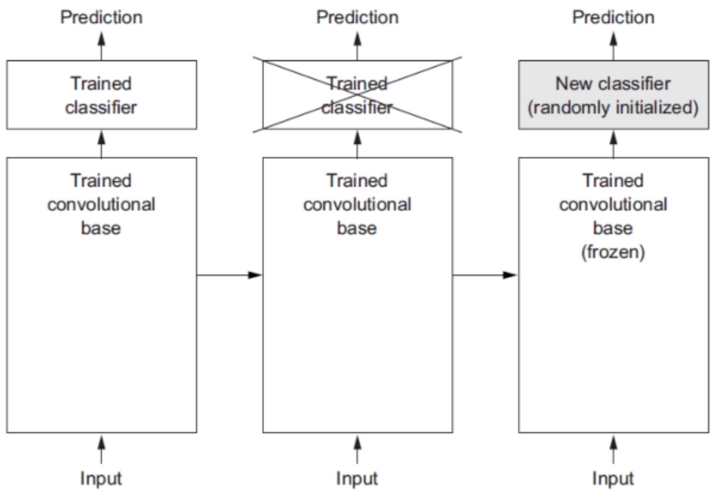
(*) Please see the keras.io doc for more details; "imagenet" weights loads the model weights correspond with training on the ImageNet dataset; "include_top=False" indicates that the model will exclude the final (ImageNet) classification layer, which we replace for the transfer learning exercise.

To ensure that you have loaded the model correctly, execute the command: "**pre_model.summary()**", this will show the model architecture summary – include this, as well as the number of "trainable parameters" in your assignment write-up.

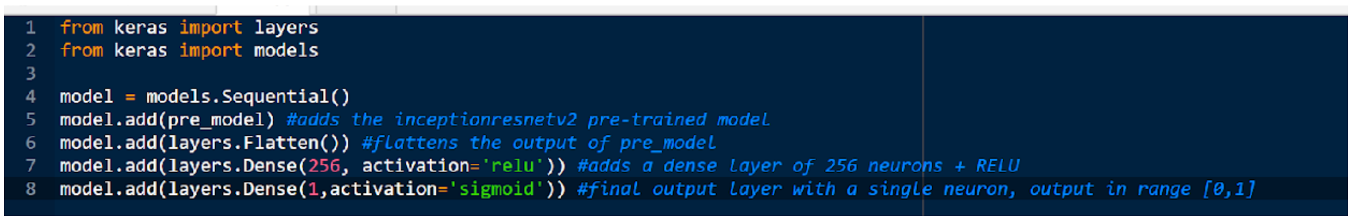
Visualize the first layer filters on your pre-trained model; include this visualization in your write-up. Provide any insightful comments about this visualization.

Step 2: Load and pre-process the cat/dog dataset. Note that you can use any reasonable image pre-processing steps that you prefer (see class lecture, etc.); report your pre-processing method in your assignment write-up. Resize the dataset images to dimensions: **(150,150,3)**.

Step 3: Next we create the "transfer head" for transfer learning, see figure below. To achieve this, we freeze and reuse the weights of the pre-trained inception-resnetv2 model and add a new classification head.


Prediction
Trained classifier
Trained convolutional base
Input
Prediction
Trained classifier
Trained convolutional base
Input
Prediction
New classifier (randomly initialized)
Trained convolutional base (frozen)
Input

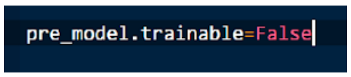
Define a new model using the sequential model class:



```
from keras import layers
from keras import models

model = models.Sequential()
model.add(pre_model) #adds the inceptionresnetv2 pre-trained model
model.add(layers.Flatten()) #flattens the output of pre_model
model.add(layers.Dense(256, activation='relu')) #adds a dense layer of 256 neurons + RELU
model.add(layers.Dense(1,activation='sigmoid')) #final output layer with a single neuron, output in range [0,1]
```

Execute: "**model.summary()**", report the architecture summary and number of model parameters in your assignment write-up.

Now we freeze the pre-trained model weights. Set:



```
pre_model.trainable=False
```

Step 4: (i) Evaluate your transfer model (without training the unfrozen weights) on the test dataset. Report the overall test accuracy, include a confusion matrix.

(ii) Now train the transfer model using **binary cross entropy loss** on the cats/dogs training data until approximate convergence; use may use any standard optimization algorithm (e.g., Adam, RMSProp, SGD, etc.). Report the optimization algorithm that you use along with any associated parameter settings in your write-up. Report the per-epoch test loss for your transfer model; include a confusion matrix on the test data for your final, trained transfer model.

(iii) Finally, perform this same transfer learning exercise using only a sub-network of the original pre-train network – you can decide how the sub-network is constructed (e.g., retain only the first *k* layers). Report the details of the sub-network in your write-up.

Freeze the weights of this sub-network and attach a "transfer head" as before. Train this model as in (ii), report the per-epoch test loss for your transfer model; include a confusion matrix on the test data for your final, trained transfer model. Discuss and analyze these results vs those of 4.(ii).

Report: Your report should include a short description of your experiments, along with the plots and discussion paragraphs requested above and any other relevant information to help shed light on your approach and results.

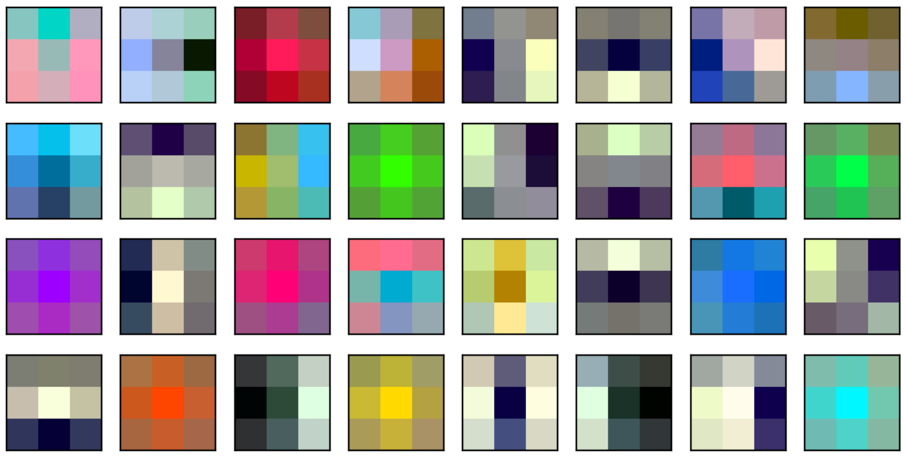
Here is what you need to turn in:

- Your report.
- Readable code.

Report

Step - 1

First Layer Filter Visualization



model summary is included in model.txt

Total trainable Parameters - 54,276,192

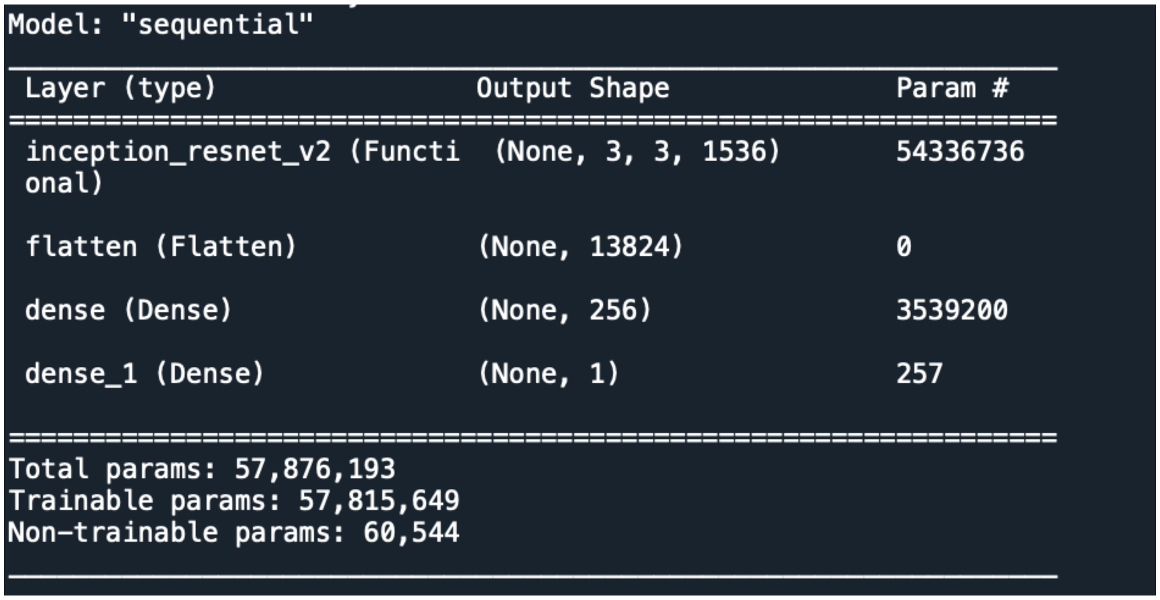
- To visualize the first convolution 2d layer, filters of 3 x 3 with 3 channels are normalized within the range of 0 to 1, then these filters are further processed in range of 0 to 255. In these filters, the dominant color is shades of darkness and bright colors which can be thought of as differentiators for image feature detectors on a small level.

Step - 2

- Firstly, images are loaded in individual arrays of training, testing from the folders in a given dimension of 150 x 150 with 3 channels of RGB. For preprocessing , the image standardization method is used. In this method, for each channel of image, mean and standard deviation is calculated which in turn from formational calculation gives standardization for a given channel. At the end, channels are combined and recreated into images.

Step - 3

Model summary



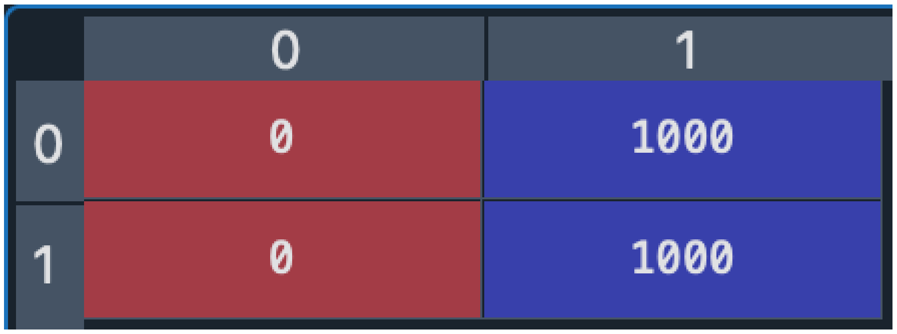
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_resnet_v2 (Functi  (None, 3, 3, 1536)       54336736
 onal)

 flatten (Flatten)           (None, 13824)             0

 dense (Dense)               (None, 256)               3539200

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 57,876,193
Trainable params: 57,815,649
Non-trainable params: 60,544
_________________________________________________________________
```

- new architecture of model combined by the downloaded Inception Resnet version two models and new head layers of 256 neurons and one dense output layer which takes input from the linear array. They use relu function in neuron activation and sigmoid which gives output in range of 0 and 1. Now, total parameters are increased by the addition of dense layers. New parameters to tune are 57,815,649.

Step - 4

(1)

confusion matrix



	0	1
0	0	1000
1	0	1000

Overall test accuracy is 50%.

- It randomly decides all the values as either one or zero. which gives 50% accuracy.

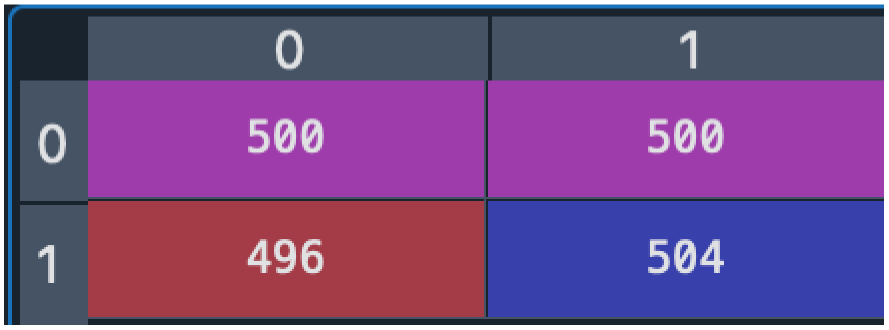
(2)

- Adam(Adaptive Moment Optimiser) optimizer calculates an exponential moving average of the gradient and the squared gradient, and the settings control the decay rates of these moving averages. It is an extended version of Stochastic Gradient Descent.
- Binary accuracy calculates the mean accuracy in overall predictions.
- Binary cross entropy computes the loss between right answers and predicted answers.
- This model works for 5 epochs

1 epoch loss - 0.1276 accuracy : 0.9510

2 epoch loss - 0.0593 accuracy : 0.9760

3 epoch loss - 0.0401 accuracy : 0.9855



	0	1
0	500	500
1	496	504

(3)

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2030 (Conv2D)        (None, 74, 74, 32)        864

 batch_normalization_2030 (B  (None, 74, 74, 32)       96
 atchNormalization)

 activation_2030 (Activation  (None, 74, 74, 32)       0
 )

 conv2d_2031 (Conv2D)        (None, 72, 72, 32)        9216

 batch_normalization_2031 (B  (None, 72, 72, 32)       96
 atchNormalization)

 activation_2031 (Activation  (None, 72, 72, 32)       0
 )

 flatten_5 (Flatten)         (None, 165888)            0

 dense_10 (Dense)            (None, 256)               42467584

 dense_11 (Dense)            (None, 1)                 257
```

```
======================================================
===========
```

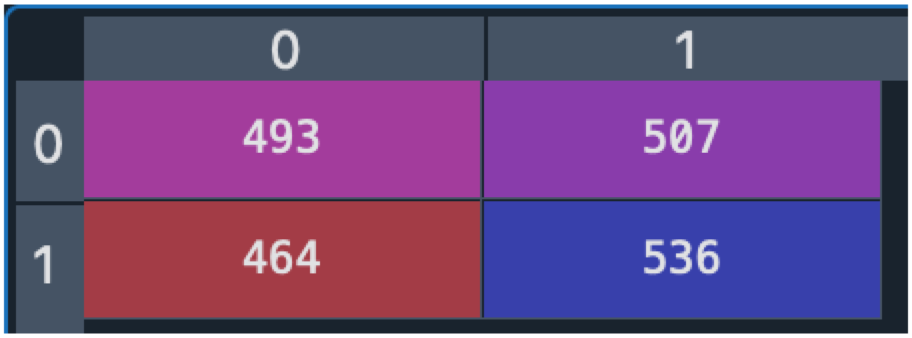
Total params: 42,478,113

Trainable params: 42,467,841

Non-trainable params: 10,272

---

1 epoch loss - 37.4662	accuracy - 0.5879
2 epoch loss - 12.4350	accuracy - 0.6842
3 epoch loss - 7.8501	accuracy - 0.7481
4 epoch loss - 5.1470	accuracy - 0.7937
5 epoch loss - 3.4737	accuracy - 0.8493



	0	1
0	493	507
1	464	536

- In comparison, the selection of the first seven layers gave me good speed and more accuracy. However, loss was high compared to the whole model of pretrained model. The convergence to the right solution is faster but in long term accuracy first may give better results.