Portland State University

CS 4/510: Computer Vision & Deep Learning

Programming Assignment #3

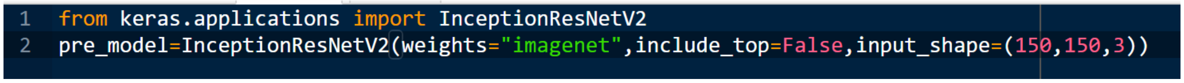
A. Rhodes

Note: This assignment is **due by Monday, 5/3 at 800pm**; 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 D2L. 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))
```

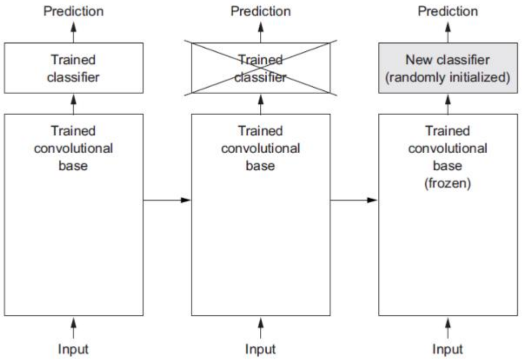
(*) Please see the keras.io doc for more details; "imagenet" weights loads the model weights correspond with training on the ImageNet dataset; include_top=False indicates that the model will exclude the final (ImageNet) classification layer, which we replace for the transfer learning exercise.

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.

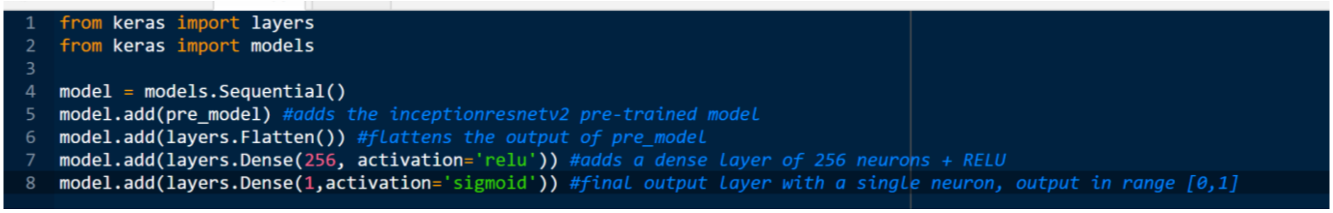
Visualize the first layer filters on your pre-trained model; include this visualization in your write-up. Provide any insightful comments about this visualization.

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:

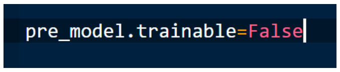


```python
from keras import layers
from keras import models

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:



```python
pre_model.trainable=False
```

<u>**Step 4**</u>: (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, RSMProp, 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 <u>sub-network</u> 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.