ENGS 108 Fall 2022 Assignment 1

Due September 15, 2022 at 11:59PM on Canvas

Instructors: George Cybenko

TAs: Chase Yakaboski and Clement Nyanhongo

In [1]:
```
#TODO: Put any import statements in this code block.
# Imports
import matplotlib.pyplot
import numpy
import tensorflow
from tensorflow.keras.layers import Conv2D, Dense, Flatten, MaxPool2D
from tensorflow.keras.models import Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
```

Question 1

Go to https://www.kaggle.com/datasets/coloradokb/dandelionimages (https://www.kaggle.com/datasets/coloradokb/dandelionimages) and download the dandelion and other images.

In [2]:
```
#TODO: Put your code to download the dandelion images in this code block.
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

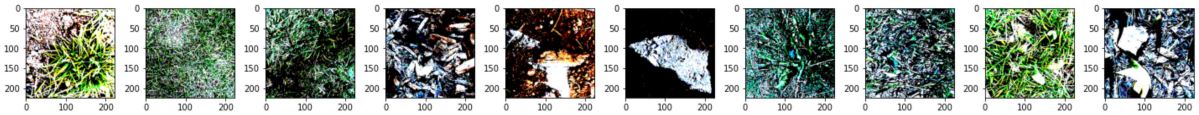
In [3]:
```
# Image Paths from Mounted Drive
train_path = '/content/drive/MyDrive/Images/train'
valid_path = '/content/drive/MyDrive/Images/valid'
test_path = '/content/drive/MyDrive/Images/test'
```

In [4]:
```
# Number of Samples to Work Through: 10.
train_batches = ImageDataGenerator(preprocessing_function=tensorflow.keras.app
lications.vgg16.preprocess_input)\
    .flow_from_directory(directory=train_path,target_size=(224,224),classes=[
"dandelion","other"],batch_size=10)
valid_batches = ImageDataGenerator(preprocessing_function=tensorflow.keras.app
lications.vgg16.preprocess_input)\
    .flow_from_directory(directory=valid_path,target_size=(224,224),classes=[
"dandelion","other"],batch_size=10)
test_batches = ImageDataGenerator(preprocessing_function=tensorflow.keras.appl
ications.vgg16.preprocess_input)\
    .flow_from_directory(directory=test_path,target_size=(224,224),classes=["d
andelion","other"],batch_size=10,shuffle=False)
```

```
Found 900 images belonging to 2 classes.
Found 200 images belonging to 2 classes.
Found 100 images belonging to 2 classes.
```

In [5]:
```
# Plot Images
# Reference: https://www.tensorflow.org/tutorials/images/classification
images, labels = next(train_batches)

def plotImages(images):
    figure, axes = matplotlib.pyplot.subplots(1, 10, figsize=(20,20))
    axes = axes.flatten()
    for image, axes_image in zip(images, axes):
        axes_image.imshow(image)
    matplotlib.pyplot.tight_layout()
    matplotlib.pyplot.show()

plotImages(images)
print(labels)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
```



```
[[0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [0. 1.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [0. 1.]]
```

In [6]:
```
# Sequential model
# Reference: https://keras.io/guides/sequential_model/

model = Sequential([
    Conv2D(filters=32,kernel_size=(3,3),activation="relu",padding="same",input
_shape=(224,224,3)),
    MaxPool2D(pool_size=(2,2),strides=2),
    Conv2D(filters=64,kernel_size=(3,3),activation="relu",padding="same"),
    MaxPool2D(pool_size=(2,2),strides=2),
    Flatten(),
    Dense(units=2,activation="softmax")
])
```

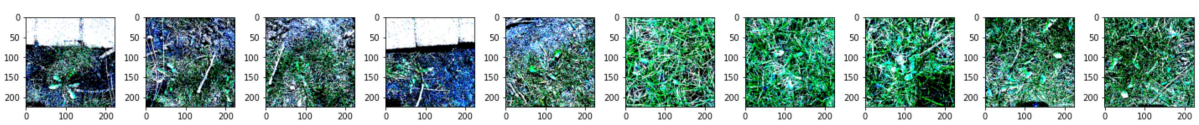
In [7]:
```
# Training The Neural Network
model.summary()
model.compile(optimizer=Adam(learning_rate=0.0001), loss="categorical_crossent
ropy",metrics=["accuracy"])
model.fit(x=train_batches,validation_data=valid_batches,epochs=10,verbose=2)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 2)                 401410

=================================================================
Total params: 420,802
Trainable params: 420,802
Non-trainable params: 0
_________________________________________________________________
Epoch 1/10
90/90 - 367s - loss: 43.6538 - accuracy: 0.6167 - val_loss: 9.6684 - val_accu
racy: 0.4850 - 367s/epoch - 4s/step
Epoch 2/10
90/90 - 260s - loss: 4.1798 - accuracy: 0.7744 - val_loss: 8.6800 - val_accur
acy: 0.5000 - 260s/epoch - 3s/step
Epoch 3/10
90/90 - 262s - loss: 1.0678 - accuracy: 0.9000 - val_loss: 11.0477 - val_accu
racy: 0.5350 - 262s/epoch - 3s/step
Epoch 4/10
90/90 - 238s - loss: 0.6188 - accuracy: 0.9344 - val_loss: 7.8728 - val_accur
acy: 0.5450 - 238s/epoch - 3s/step
Epoch 5/10
90/90 - 236s - loss: 0.2466 - accuracy: 0.9689 - val_loss: 10.6967 - val_accu
racy: 0.5100 - 236s/epoch - 3s/step
Epoch 6/10
90/90 - 234s - loss: 0.0259 - accuracy: 0.9911 - val_loss: 8.8412 - val_accur
acy: 0.5550 - 234s/epoch - 3s/step
Epoch 7/10
90/90 - 241s - loss: 0.0376 - accuracy: 0.9933 - val_loss: 8.2826 - val_accur
acy: 0.5600 - 241s/epoch - 3s/step
Epoch 8/10
90/90 - 234s - loss: 0.0783 - accuracy: 0.9833 - val_loss: 8.5765 - val_accur
acy: 0.5500 - 234s/epoch - 3s/step
Epoch 9/10
90/90 - 239s - loss: 0.0151 - accuracy: 0.9967 - val_loss: 8.8853 - val_accur
acy: 0.5650 - 239s/epoch - 3s/step
Epoch 10/10
90/90 - 236s - loss: 0.1867 - accuracy: 0.9733 - val_loss: 8.2695 - val_accur
acy: 0.5400 - 236s/epoch - 3s/step
```

Out[7]:
```
<keras.callbacks.History at 0x7fcc3422cc10>
```

In [8]:
```
# Plot Images
test_images, test_labels = next(test_batches)

plotImages(test_images) # plotImages Definded: Line 25
print(test_labels)
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
WARNING:matplotlib.image:Clipping input data to the valid range for imshow wi
th RGB data ([0..1] for floats or [0..255] for integers).
```



```
[[1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]
 [1. 0.]]
```

(a) Define a reasonable distance metric between images and explain it with an example.

TODO: Explain your distance metric with text, equations or both.

In my opinion, the best way to deal with this dataset is by image classification. Another way to deal with this dataset is by shape detection (OpenCV). However, in this assignment, I utilized TensorFlow to build and train a model to classifiy images as 'dandelion' or 'other.' In the following blocks of code, I am able to make a prediction based on my saved model. While you can run the script above to see how my model was trained. Please be advised that this process takes 45-60 minutes. Which is why I saved a version of this model to my google drive. While there is some definite overfitting; This basic model is adiquate for the assignment. I believe that it will much better at finding dandelions in photos when they have yet to bloom - no yellow pixels. As well as fining dandelions in photos when they are missing their capitulum - again, no yellow pixels. If you have any questions, please do not hesitate to reach out!

In [10]:
```
!mkdir -p saved_model
model.save('/content/drive/MyDrive/saved_model/my_model')
```

In [11]:
```
#TODO: Test your distance metric on two example images from the dandelions dat
aset.
# Step 1: Import Saved Model
new_model = tensorflow.keras.models.load_model('/content/drive/MyDrive/saved_m
odel/my_model')
new_model.summary()

# Evaluateing the restored model
loss, acc = new_model.evaluate(test_images, test_labels, verbose=2)
print('Restored model, accuracy: {:5.2f}%'.format(100 * acc))
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 224, 224, 32)      896

 max_pooling2d (MaxPooling2D  (None, 112, 112, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 112, 112, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 56, 56, 64)       0
 2D)

 flatten (Flatten)           (None, 200704)            0

 dense (Dense)               (None, 2)                 401410

=================================================================
Total params: 420,802
Trainable params: 420,802
Non-trainable params: 0
_________________________________________________________________
1/1 - 0s - loss: 3.7160 - accuracy: 0.6000 - 441ms/epoch - 441ms/step
Restored model, accuracy: 60.00%
```

In [14]:
```
# Step 2: Test my model on two (I did 100) example images from the dandelions
 dataset.
predictions = new_model.predict(x=test_batches, verbose=0)
print(predictions)
```

```
[[1.00000000e+00 6.52168319e-17]
 [6.95142720e-04 9.99304891e-01]
 [1.00000000e+00 1.81728486e-11]
 [1.00000000e+00 7.13025194e-10]
 [1.00000000e+00 9.44907139e-11]
 [4.19761892e-03 9.95802343e-01]
 [1.00000000e+00 7.53866067e-11]
 [1.00000000e+00 2.21244304e-16]
 [1.12527363e-01 8.87472630e-01]
 [2.21552832e-10 1.00000000e+00]
 [2.05062202e-06 9.99997973e-01]
 [1.23782831e-10 1.00000000e+00]
 [1.00000000e+00 3.85855187e-23]
 [9.99997973e-01 1.96950577e-06]
 [6.36277742e-09 1.00000000e+00]
 [9.99999762e-01 1.92235333e-07]
 [2.63781520e-04 9.99736249e-01]
 [9.48193908e-01 5.18060550e-02]
 [9.92200792e-01 7.79927429e-03]
 [8.66291642e-01 1.33708328e-01]
 [9.99998808e-01 1.17277295e-06]
 [9.74968314e-01 2.50317231e-02]
 [9.99867797e-01 1.32133413e-04]
 [5.07090380e-03 9.94929075e-01]
 [1.52050525e-11 1.00000000e+00]
 [1.00000000e+00 2.69665887e-12]
 [8.42007867e-04 9.99158025e-01]
 [2.02095048e-14 1.00000000e+00]
 [9.89270806e-01 1.07291378e-02]
 [6.99060321e-01 3.00939649e-01]
 [1.76507656e-10 1.00000000e+00]
 [1.99768633e-01 8.00231397e-01]
 [9.78957715e-08 9.99999881e-01]
 [5.34386251e-12 1.00000000e+00]
 [5.71446189e-11 1.00000000e+00]
 [9.39186793e-05 9.99906063e-01]
 [3.13619165e-11 1.00000000e+00]
 [2.18237414e-10 1.00000000e+00]
 [2.36451410e-08 1.00000000e+00]
 [1.14030045e-05 9.99988556e-01]
 [1.45915467e-11 1.00000000e+00]
 [3.30798912e-07 9.99999642e-01]
 [1.00000000e+00 8.20685750e-13]
 [1.81968391e-01 8.18031609e-01]
 [3.89993460e-08 1.00000000e+00]
 [8.08478391e-04 9.99191582e-01]
 [5.21451511e-05 9.99947906e-01]
 [6.21571131e-02 9.37842906e-01]
 [5.82732308e-23 1.00000000e+00]
 [1.36523260e-04 9.99863505e-01]
 [4.33058700e-13 1.00000000e+00]
 [9.98158753e-01 1.84129179e-03]
 [6.47358577e-13 1.00000000e+00]
 [3.98348687e-09 1.00000000e+00]
 [1.70241493e-14 1.00000000e+00]
 [8.57719873e-18 1.00000000e+00]
 [7.10658144e-30 1.00000000e+00]
```

```
 [1.61259998e-38 1.00000000e+00]
 [1.31400364e-23 1.00000000e+00]
 [7.16782449e-12 1.00000000e+00]
 [1.51473842e-12 1.00000000e+00]
 [8.72581115e-17 1.00000000e+00]
 [5.30087640e-10 1.00000000e+00]
 [1.64761130e-19 1.00000000e+00]
 [6.73891506e-28 1.00000000e+00]
 [8.78302053e-09 1.00000000e+00]
 [3.30348031e-23 1.00000000e+00]
 [2.31802262e-13 1.00000000e+00]
 [2.49883381e-09 1.00000000e+00]
 [1.48375946e-07 9.99999881e-01]
 [1.28562980e-20 1.00000000e+00]
 [5.95221614e-08 1.00000000e+00]
 [4.70367390e-22 1.00000000e+00]
 [5.65285241e-08 1.00000000e+00]
 [6.29826775e-03 9.93701696e-01]
 [1.80554798e-06 9.99998212e-01]
 [8.15067597e-11 1.00000000e+00]
 [3.33173045e-10 1.00000000e+00]
 [1.18581287e-03 9.98814225e-01]
 [1.02189091e-25 1.00000000e+00]
 [3.73555058e-35 1.00000000e+00]
 [1.47581449e-22 1.00000000e+00]
 [2.63281735e-25 1.00000000e+00]
 [9.46731790e-28 1.00000000e+00]
 [1.46889794e-14 1.00000000e+00]
 [3.68963120e-23 1.00000000e+00]
 [1.23709513e-20 1.00000000e+00]
 [3.31622081e-15 1.00000000e+00]
 [9.51520605e-08 9.99999881e-01]
 [8.38152191e-05 9.99916196e-01]
 [2.29760322e-09 1.00000000e+00]
 [1.00000000e+00 8.79433408e-13]
 [3.99910550e-06 9.99995947e-01]
 [3.62842894e-14 1.00000000e+00]
 [4.10914344e-17 1.00000000e+00]
 [1.38474088e-09 1.00000000e+00]
 [2.11090810e-05 9.99978900e-01]
 [3.06379079e-05 9.99969363e-01]
 [5.07288544e-21 1.00000000e+00]
 [4.62224490e-14 1.00000000e+00]]
```

(b) Which two images are most similar according to your metric?

In [16]:
```
#TODO: Write your code here.
# While you could also build out a confusion matrix here, because my tester se
t only had 100 photos, it was just as easy to scroll through the results to se
e which two images were the most similar.
# The two Images from my test images folder, are the two that returned [1.0000
0000e+00 8.20685750e-13] and [1.00000000e+00 8.79433408e-13] at index[42] and
 index[91].
# Starting from index[0] and counting over -- that would be images "https://dr
ive.google.com/open?id=1jtAxoskWHTUrxE0ES4WhzPoMFMUm_fth" and "https://drive.g
oogle.com/open?id=11fWEW7z6DSsbfqs8HjqO100V0TGdJ1ZE".
```

(c) Which two images are most different according to your metric?

In [ ]:
```
#TODO: Write your code here.
# Two Images from my test images folder, that are most different are the two t
hat returned [1.61259998e-38 1.00000000e+00] and  [1.00000000e+00       8.7943
3408e-13] at index[57] and index[91].
# Again, starting from index[0] and counting over -- that would be images "htt
ps://drive.google.com/open?id=1HNnFKR0DMUV_hSEzQ9Enw3YkI7Udn-yY" and "https://
drive.google.com/open?id=11fWEW7z6DSsbfqs8HjqO100V0TGdJ1ZE".
# I made sure this test batch would not be shuffeled.
```

(d) What dandelion image is the most representative of the dandelion images?

In [15]:
```
#TODO: Write your code here. Hint: Which image is closest to all the dandelion
images?
numpy.round(predictions)
# In accordance with my model, everything labeled [1. 0.] is predicted as bein
g a dandelion.
# In my test set, the dandelion image most representative of the dandelion ima
ges is the image that returned [1.00000000e+00  8.20685750e-13] at index[42]
# Starting from index[0] and counting over -- that would be image "https://dri
ve.google.com/open?id=1jtAxoskWHTUrxE0ES4WhzPoMFMUm_fth".
```

Out[15]:
```
array([[1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
```

```
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [1., 0.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

(e) Which other image is most representative of the other images?

In [ ]:
```
#TODO: Write your code here.
# In accordance with my model, everything labeled [0. 1.] is predicted as bein
g other.
# In my test set, the image most representative of the other images is the ima
ge that returned [1.61259998e-38 1.00000000e+00] at index[57]
# Starting from index[0] and counting over -- that would be image "https://dri
ve.google.com/open?id=1HNnFKR0DMUV_hSEzQ9Enw3YkI7Udn-yY".
```

(f) What dandelion image and other image are closest to each other?

In [ ]:
```
#TODO: Write your code here.
# As stated above, two Images from my test images folder, that are very simila
r are the two that returned [1.00000000e+00 8.20685750e-13] and [1.00000000e+0
0 8.79433408e-13] at index[42] and index[91].
# One was a dandelion image and other was an other image.
# Starting from index[0] and counting over -- that would be images "https://dr
ive.google.com/open?id=1jtAxoskWHTUrxE0ES4WhzPoMFMUm_fth" and "https://drive.g
oogle.com/open?id=11fWEW7z6DSsbfqs8HjqO100V0TGdJ1ZE".
```

Question 2

Let

$$f(x_1, x_2, x_3, x_4, x_5) = \frac{1}{1 + e^{-2x_1 - 3x_2 - 4x_3 - 5x_4 - 6x_5}}$$

(a) What is

$$\frac{\partial f}{\partial x_i}$$

Calculated this derivative by hand: quotient rule

Answer: ((1+e^(-2x^1-3x^2-4x^3-5x^4-6x^5))(0)-(1)((e^(-2x^1-3x^2-4x^3-5x^4-6x^5))(-2-6x-12x^2-20x^3-30x^4)))/(1+e^(-2x^1-3x^2-4x^3-5x^4-6x^5))^2

The *sympy* calculation can be found below.

In [ ]:
```
import sympy

x = sympy.symbols("x", real=True)
function = 1 / (1 + sympy.exp(-2*(x**1)-3*(x**2)-4*(x**3)-5*(x**4)-6*(x**5)))

print(sympy.diff(function,x))
```

```
-(-30*x**4 - 20*x**3 - 12*x**2 - 6*x - 2)*exp(-6*x**5 - 5*x**4 - 4*x**3 - 3*x
**2 - 2*x)/(exp(-6*x**5 - 5*x**4 - 4*x**3 - 3*x**2 - 2*x) + 1)**2
```

(b) What is

$$\nabla f(1, 0, 0, 0, 0) \cdot ones(5, 1)$$

In [ ]:
```
# To solve for this equation, my setps were as follows:
# Take the derivative and solve f(1,0,0,0,0).
# Place each solution in matrix and multiply by a 5 row matrix filled with one
s.

from math import exp

x = 1
function_1 = -(-30*x**4 - 20*x**3 - 12*x**2 - 6*x - 2)*exp(-6*x**5 - 5*x**4 -
4*x**3 - 3*x**2 - 2*x)/(exp(-6*x**5 - 5*x**4 - 4*x**3 - 3*x**2 - 2*x) + 1)**2

y = 0
function_2 = -(-30*y**4 - 20*y**3 - 12*y**2 - 6*y - 2)*exp(-6*y**5 - 5*y**4 -
4*y**3 - 3*y**2 - 2*y)/(exp(-6*y**5 - 5*y**4 - 4*y**3 - 3*y**2 - 2*y) + 1)**2

matrix = [[function_1],
          [function_2],
          [function_2],
          [function_2],
          [function_2]]

matrix * numpy.ones((5,1))
```

Out[ ]:
```
array([[1.44280753e-07],
       [5.00000000e-01],
       [5.00000000e-01],
       [5.00000000e-01],
       [5.00000000e-01]])
```

Answer

```
array([[1.44280753e-07],
       [5.00000000e-01],
       [5.00000000e-01],
       [5.00000000e-01],
       [5.00000000e-01]])
```

TODO: Write your answers in Markdown/Latex in this code block.

(b) What is

$$\nabla f(0,1,0,0,0) \cdot ones(5,1)$$

In [ ]:
```
# To solve for this equation, my setps were as follows:
# Take the derivative and solve f(0,1,0,0,0).
# Place each solution in matrix and multiply by a 5 row matrix filled with one
s.

from math import exp

x = 1
function_1 = -(-30*x**4 - 20*x**3 - 12*x**2 - 6*x - 2)*exp(-6*x**5 - 5*x**4 -
4*x**3 - 3*x**2 - 2*x)/(exp(-6*x**5 - 5*x**4 - 4*x**3 - 3*x**2 - 2*x) + 1)**2

y = 0
function_2 = -(-30*y**4 - 20*y**3 - 12*y**2 - 6*y - 2)*exp(-6*y**5 - 5*y**4 -
4*y**3 - 3*y**2 - 2*y)/(exp(-6*y**5 - 5*y**4 - 4*y**3 - 3*y**2 - 2*y) + 1)**2

matrix = [[function_2],
          [function_1],
          [function_2],
          [function_2],
          [function_2]]

matrix * numpy.ones((5,1))
```

Out[ ]:
```
array([[5.00000000e-01],
       [1.44280753e-07],
       [5.00000000e-01],
       [5.00000000e-01],
       [5.00000000e-01]])
```

Answer:

```
array([[5.00000000e-01],
       [1.44280753e-07],
       [5.00000000e-01],
       [5.00000000e-01],
       [5.00000000e-01]])
```

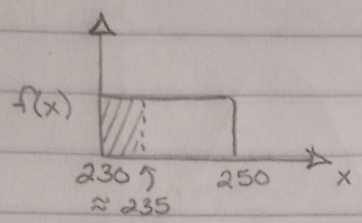
TODO: Write your answers in Markdown/Latex in this code block.

Question 3

Some Tesla batteries have an amp hour rating of 230 but in tests, it has been determined that the actual amp hours of a new battery is a random variable uniformly distributed between 230 and 250 amp hours. (Tesla is conservative in reporting this.)

(a) Tesla service center is shipped 5 randomly selected batteries. What is the probability that exactly 2 of the 5 batteries have less than 235 amp hours?

In [ ]:
```
#TODO: Write your answers in Markdown/Latex or code.
```



Area = 1 = (b - a)(f(x))

$\hookrightarrow \frac{1}{b-a} = \boxed{f(x) = \frac{1}{250-230} = \frac{1}{20}}$

a) What is the prob. 2 of the 5 will have less than 235 amp hours?

Area = b · h

$P(x < 235) = (5)(\frac{1}{20}) = 25\%$ for 1 battery

$(25\%)^2(75\%)^3 \approx \boxed{2.64\%}$

(b) Your Tesla battery is being replaced at that service center. The service center does not measure the amp hours of batteries it receives or installs. What is the probability that the smallest amp hours of the 5 batteries is below 235?

In [ ]:
```
#TODO: Write your answers in Markdown/Latex or code.
```

b) What is the prob. the the smallest amp. hr. of the 5 batteries is less than 235?

$P(x < 235)$ = Area

$b \cdot h = (5)(\frac{1}{20}) = \boxed{25\%}$

(c) What is the probability that your Tesla gets the battery with the lowest amp hours?

In [ ]:
```
#TODO: Write your answers in Markdown/Latex or code.
# The area of a Continuous Uniform Distribution (Rectangle) of base (0) * heig
ht (1/20) is equal to zero.
```

c) What is the prob. that you get a battery equal to 230?

$P(x = 230) = \boxed{0}$