Using Neural Networks for Image Recognition

For this portfolio, we will make use of the Logo Images Dataset obtained from https://github.com/msn199959/Logo-2k-plus-Dataset, it contains over 150, 000 images, which are categorised as follows:

Root Category	Logos	Images
Food	769	54,507
Clothes	286	20,413
Institution	238	17,103
Accessories	210	14,569
Transportation	203	14,719
Electronic	191	13,972
Necessities	182	13,205
Cosmetic	115	7,929
Leisure	99	7,338
Medical	48	3,385
Total	2,341	167,140

Introduction

A neural network attempts to replicate the structure of the brain, in which neurons pass electrical current form a directed network. Here if enough neurons sense an the input (for example a touch on the skin), a signal is sent via a charge through the network to produce an output.

An artificial neural network imitates this by having an input layer, hidden layers and an output layer as shown below:

\textbf{REVISIT!!!}

As we can see, the internal neurons can have multiple inputs and outputs. The inputs, $x_1, \ldots, x_n$, have weights $w_1, \ldots, w_n$ and this weighted input is passed to an activation function $\phi()$, to get the output of the neuron as:

$$y = \phi\left(\sum_i x_i w_i\right)$$

A simple neural network is made up of an input layer, hidden layers and an output layers. Our aim is to select correct weights on each edge using iterative methods.

Backpropogation

This is a training method, also referred to "the backward propogation of errors". To use this, we first define the following quatities

$$J(y) = (t-y)^2 \text{ the loss function,} \tag{1}$$

$$D_n(y) = \frac{dJ(y)}{dw_n} \text{ the derivative of the loss function} \tag{2}$$

We then perform the following steps for each $(x, t) \in X$

1. Pass x through the neural network and obtain the output y
2. Obtain the new weight for each edge $w_n' = \delta w_n = -RD_n(y)$ for a learning rate R

The Pet Breed Dataset

In [9]:

```
import pathlib

data_dir = pathlib.Path("Pet_Breeds")
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

```
3366
```

In [10]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [11]:

```
import PIL
import PIL.Image

abyssinian = list(data_dir.glob('abyssinian/*'))
PIL.Image.open(str(abyssinian[50]))
```

Out[11]:

In [12]:

```
TF_ENABLE_ONEDNN_OPTS=0
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential

batch_size = 32
img_height = 180
img_width = 180
```

In [13]:

```
train_dataset, test_dataset = tf.keras.utils.image_dataset_from_directory(
      data_dir,
      validation_split=0.2,
      subset="both",
      seed=123,
      image_size= (img_height, img_width),
      batch_size= batch_size
  )
```

```
Found 3535 files belonging to 23 classes.
Using 2828 files for training.
Using 707 files for validation.
```

In [14]:

```
import tensorflow as tf
tf.config.list_physical_devices('GPU')
```

Out[14]:

```
[]
```

In [15]:

```
class_names = train_dataset.class_names
print(class_names)
```

```
['abyssinian', 'american shorthair', 'beagle', 'boxer', 'bulldog', 'chihuah
ua', 'corgi', 'dachshund', 'german shepherd', 'golden retriever', 'husky',
'labrador', 'maine coon', 'mumbai cat', 'persian cat', 'pomeranian', 'pug',
'ragdoll cat', 'rottwiler', 'shiba inu', 'siamese cat', 'sphynx', 'yorkshir
e terrier']
```

In [16]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_dataset.take(5):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

```
2023-04-26 14:40:40.892804: I tensorflow/core/common_runtime/executor.cc:11
97] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not ind
icate an error and you can ignore this message): INVALID_ARGUMENT: You must
feed a value for placeholder tensor 'Placeholder/_4' with dtype int32 and s
hape [2828]
         [[{{node Placeholder/_4}}]]
2023-04-26 14:40:40.893363: I tensorflow/core/common_runtime/executor.cc:11
97] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not ind
icate an error and you can ignore this message): INVALID_ARGUMENT: You must
feed a value for placeholder tensor 'Placeholder/_0' with dtype string and
shape [2828]
         [[{{node Placeholder/_0}}]]
```

Data Processing

In [17]:
```
AUTOTUNE = tf.data.AUTOTUNE

train_dataset = train_dataset.cache().shuffle(1000).prefetch(buffer_size=AUT
test_dataset = test_dataset.cache().prefetch(buffer_size=AUTOTUNE)
```

In [18]:
```
normalization_layer = layers.Rescaling(1./255)
```

In [19]:
```
num_classes = len(class_names)

model = Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

In [20]:
```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])
```

In [21]:
```
epochs=10
history = model.fit(
  train_dataset,
  validation_data=test_dataset,
  epochs=epochs
)
```

```
Epoch 1/10
```

```
2023-04-26 14:40:53.969908: I tensorflow/core/common_runtime/executor.cc:11
97] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not ind
icate an error and you can ignore this message): INVALID_ARGUMENT: You must
feed a value for placeholder tensor 'Placeholder/_4' with dtype int32 and s
hape [2828]
         [[{{node Placeholder/_4}}]]
2023-04-26 14:40:53.970458: I tensorflow/core/common_runtime/executor.cc:11
97] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not ind
icate an error and you can ignore this message): INVALID_ARGUMENT: You must
feed a value for placeholder tensor 'Placeholder/_4' with dtype int32 and s
hape [2828]
         [[{{node Placeholder/_4}}]]
```

```
89/89 [==============================] - ETA: 0s - loss: 3.1427 - accuracy:
0.0591
```

```
2023-04-26 14:41:12.843482: I tensorflow/core/common_runtime/executor.cc:11
97] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not ind
icate an error and you can ignore this message): INVALID_ARGUMENT: You must
feed a value for placeholder tensor 'Placeholder/_4' with dtype int32 and s
hape [707]
         [[{{node Placeholder/_4}}]]
2023-04-26 14:41:12.843680: I tensorflow/core/common_runtime/executor.cc:11
97] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not ind
icate an error and you can ignore this message): INVALID_ARGUMENT: You must
feed a value for placeholder tensor 'Placeholder/_4' with dtype int32 and s
hape [707]
         [[{{node Placeholder/_4}}]]
```

```
89/89 [==============================] - 22s 151ms/step - loss: 3.1427 - ac
curacy: 0.0591 - val_loss: 3.0483 - val_accuracy: 0.0679
Epoch 2/10
89/89 [==============================] - 12s 130ms/step - loss: 2.9214 - ac
curacy: 0.1277 - val_loss: 2.9274 - val_accuracy: 0.1273
Epoch 3/10
89/89 [==============================] - 12s 131ms/step - loss: 2.6216 - ac
curacy: 0.2281 - val_loss: 2.8700 - val_accuracy: 0.1627
Epoch 4/10
89/89 [==============================] - 12s 131ms/step - loss: 2.0710 - ac
curacy: 0.3876 - val_loss: 2.9338 - val_accuracy: 0.1966
Epoch 5/10
89/89 [==============================] - 12s 132ms/step - loss: 1.2289 - ac
curacy: 0.6312 - val_loss: 3.7097 - val_accuracy: 0.1952
Epoch 6/10
89/89 [==============================] - 12s 132ms/step - loss: 0.5505 - ac
curacy: 0.8441 - val_loss: 4.5895 - val_accuracy: 0.1754
Epoch 7/10
89/89 [==============================] - 12s 130ms/step - loss: 0.2273 - ac
curacy: 0.9445 - val_loss: 6.0688 - val_accuracy: 0.2008
Epoch 8/10
89/89 [==============================] - 12s 131ms/step - loss: 0.1319 - ac
curacy: 0.9703 - val_loss: 7.1190 - val_accuracy: 0.1952
Epoch 9/10
89/89 [==============================] - 12s 132ms/step - loss: 0.0743 - ac
curacy: 0.9851 - val_loss: 7.5558 - val_accuracy: 0.1980
Epoch 10/10
89/89 [==============================] - 12s 131ms/step - loss: 0.0634 - ac
curacy: 0.9866 - val_loss: 8.0122 - val_accuracy: 0.1754
```

In [22]:

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
              metrics=['accuracy'])
```

In [23]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_1 (Rescaling)     (None, 180, 180, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 180, 180, 16)      448       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 90, 90, 16)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 90, 90, 32)        4640      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 45, 45, 32)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 45, 45, 64)        18496     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 22, 22, 64)       0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 30976)             0         
                                                                 
 dense (Dense)               (None, 128)               3965056   
                                                                 
 dense_1 (Dense)             (None, 23)                2967      
                                                                 
=================================================================
Total params: 3,991,607
Trainable params: 3,991,607
Non-trainable params: 0
_________________________________________________________________
```

In [24]:

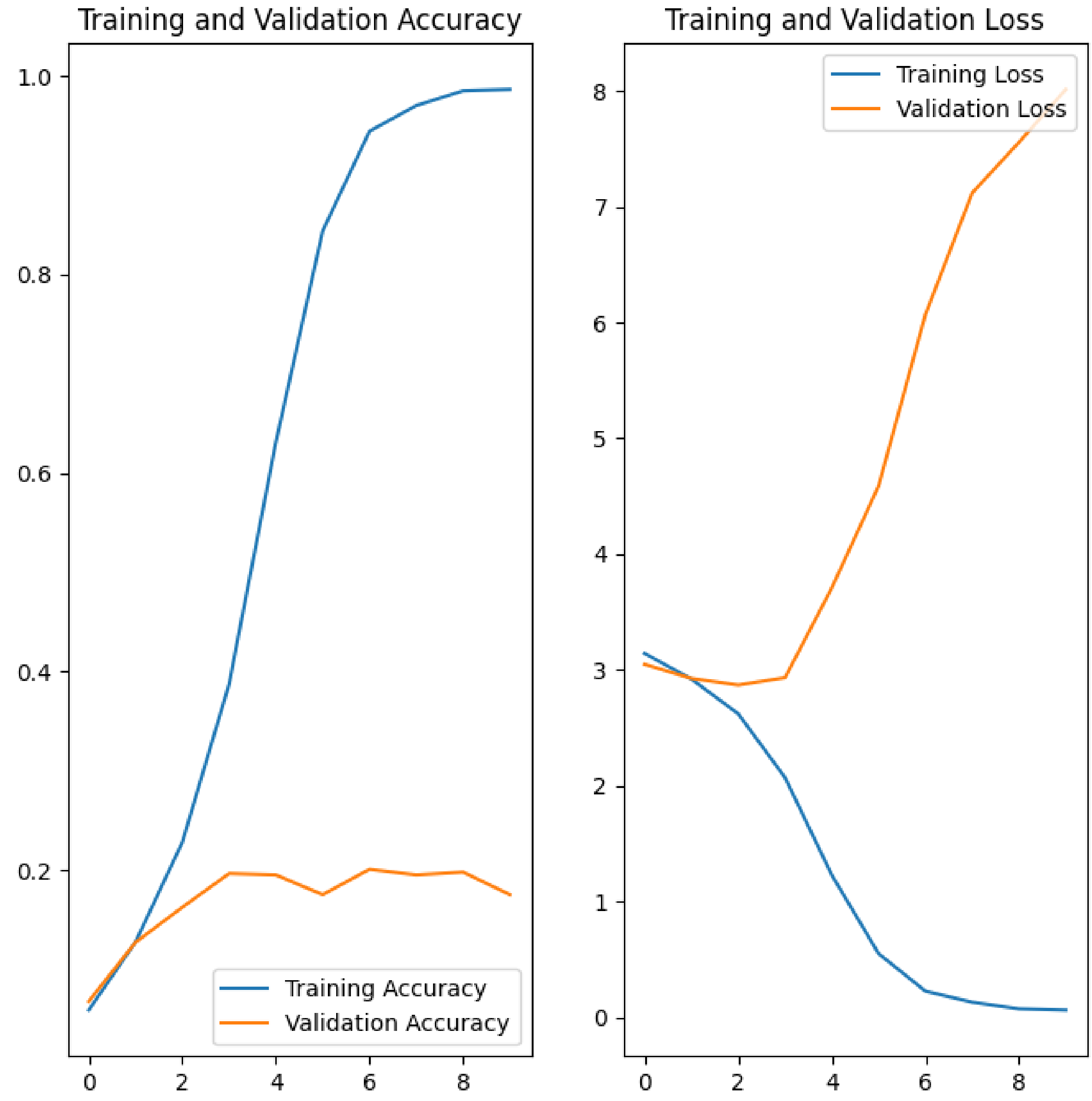
```python
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```


Training and Validation Accuracy
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
Training Accuracy
Validation Accuracy
Training and Validation Loss
8
7
6
5
4
3
2
1
0
0
2
4
6
8
Training Loss
Validation Loss