Deep Learning Image Classification using Neural Network

by Matan-Ben Nagar & Yaara Kresner-Barak

Importing our libraries for the project

```
import matplotlib.pyplot as plt
import numpy as np

import tensorflow as tf
import tensorflow_datasets as tfds

from tensorflow import keras
from keras.layers import Dense, Flatten,Conv2D,MaxPool2D,Dropout,AveragePooling2D
tfds.disable_progress_bar()
```

Load rock_paper_scissors dataset

The dataset contains 2892 images of hands playing rock, paper, scissor game. Its have two features- image (300, 300, 3) and lable.

```
##import the dataset from tensorflow_datasets library
builder = tfds.builder('rock_paper_scissors')
```

Split Rock, Paper, Scissors data The train set contains 2520 images, and the test set contains 372 images.

```
#load the train and test sets from the DB
ds_train = tfds.load(name="rock_paper_scissors", split="train")
ds_test = tfds.load(name="rock_paper_scissors", split="test")
```

```
Downloading and preparing dataset rock_paper_scissors/3.0.0 (download: 219.53 MiB, generated: Unknown size, total:
Shuffling and writing examples to /root/tensorflow_datasets/rock_paper_scissors/3.0.0.incompleteGQU6ZK/rock_paper_
```

```
Shuffling and writing examples to /root/tensorflow_datasets/rock_paper_scissors/3.0.0.incompleteGQU6ZK/rock_paper_
Dataset rock_paper_scissors downloaded and prepared to /root/tensorflow_datasets/rock_paper_scissors/3.0.0. Subseq
```

Converting the tensorflow dataset format into numpy format,

Create numpy arrays that contains the images and the labels separately,

And change the images three color channels RGB format to one color channel (to reduce the unimportant data)

```
train_images = np.array([example['image'].numpy()[:,:,0] for example in ds_train])
train_labels = np.array([example['label'].numpy() for example in ds_train])

test_images = np.array([example['image'].numpy()[:,:,0] for example in ds_test])
test_labels = np.array([example['label'].numpy() for example in ds_test])
```

Reshaping the images to 300 x 300 x 1 (add color feature- grayscale images).

```
train_images = train_images.reshape(2520, 300, 300, 1)
test_images = test_images.reshape(372, 300, 300, 1)
```

getting us ready to be able to convert it from a scale of 0 to 1 instead of 0 to 255

```
train_images = train_images.astype('float32')
test_images = test_images.astype('float32')
```

Normalize the Data

Train images dividing equal by 255, So the max value we can have is 255 because RGB values are between 0 and 255 so by doing this we're scaling every value to be between 0 & 1 and this is just a good common practice that helps you classify it. It helps the basically network learn better than if you use the 0 to 255 values you could leave it 0 to 255 but it's just ultimately it's gonna probably decrease your performance a bit, so it's a common step to normalize between 0 & 1.

```
train_images /= 255
test_images /= 255
```

Logistic regression

```
# Output layer.
model_lr = keras.Sequential([
   keras.layers.Flatten(),
   keras.layers.Dense(3, activation='softmax')
])


# adam_optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
rmsprop_optimizer = tf.keras.optimizers.RMSprop(learning_rate=0.001)

model_lr.compile(
    optimizer=rmsprop_optimizer,
    loss=tf.keras.losses.sparse_categorical_crossentropy,
    metrics=['accuracy']
)
model_lr.fit(train_images, train_labels, epochs=5, batch_size=32)
```

```
Epoch 1/5
79/79 [==============================] - 3s 9ms/step - loss: 54.6780 - accuracy: 0.3742
Epoch 2/5
79/79 [==============================] - 1s 9ms/step - loss: 38.5769 - accuracy: 0.4119
Epoch 3/5
79/79 [==============================] - 1s 9ms/step - loss: 33.9021 - accuracy: 0.4591
Epoch 4/5
79/79 [==============================] - 1s 10ms/step - loss: 29.8731 - accuracy: 0.5032
Epoch 5/5
79/79 [==============================] - 1s 13ms/step - loss: 26.4786 - accuracy: 0.5417
<keras.callbacks.History at 0x7fea59393150>
```

```
model_lr.evaluate(test_images, test_labels)
```

```
12/12 [==============================] - 0s 8ms/step - loss: 39.3082 - accuracy: 0.4032
[39.30821990966797, 0.4032258093357086]
```

MLP

This model network layer transform the 300 by 300 image into single colum, After that we have two layers of activation relu function- because the constant gradient of ReLUs results in faster learning. Finally ,the output layer going to be the same size as the number of labels we trying to classify- we use softmax because it efficient in classification problems.

```
model = keras.Sequential([
  Flatten(),
  Dense(512, activation='relu'),
  Dense(256, activation='relu'),
  Dense(3, activation='softmax')
])

#setup loss function
model.compile(optimizer='adam',
              loss=keras.losses.SparseCategoricalCrossentropy(),
              metrics=['accuracy'])

#fit our data to the model
model.fit(train_images, train_labels, epochs=5, batch_size=32)
```

```
Epoch 1/5
79/79 [==============================] - 4s 28ms/step - loss: 19.8999 - accuracy: 0.4389
Epoch 2/5
79/79 [==============================] - 2s 27ms/step - loss: 3.3100 - accuracy: 0.6329
Epoch 3/5
79/79 [==============================] - 2s 26ms/step - loss: 0.6363 - accuracy: 0.8171
Epoch 4/5
```

```
79/79 [==============================] - 2s 27ms/step - loss: 1.3719 - accuracy: 0.6770
Epoch 5/5
79/79 [==============================] - 2s 27ms/step - loss: 0.3837 - accuracy: 0.8611
<keras.callbacks.History at 0x7fe9de92aad0>
```

```
model.evaluate(test_images, test_labels)
```

```
12/12 [==============================] - 0s 11ms/step - loss: 2.3324 - accuracy: 0.4462
[2.332437753677368, 0.4462365508079529]
```

In this case we overfitting to our data - the model not learning the train data. we can see it by the results- the accuracy in the train data is 0.89 and the accuracy in the test data is 0.44.

Convolutional Neural Network

This time the first layer will be Conv2D() because the dataset consist of 2D images. The first variable inserted in the function is basically how many times a smaller gird is passing over the image
this is how big or smaller grid is so if I said three and I didn't pass in three to start off and we'll leave the rides at one two one that just means they'll move one every time so it's gonna be a sliding window of three by three

```
model = keras.Sequential([
   AveragePooling2D(6,3, input_shape=(300,300,1)),
   Conv2D(64, 3, activation='relu'),
   Conv2D(32, 3, activation='relu'),
   MaxPool2D(2,2),
   Dropout(0.5),
   Flatten(),
   Dense(128, activation='relu'),
   Dense(3, activation='softmax')
])

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

```
model.fit(train_images, train_labels, epochs=5, batch_size=32)
```

```
Epoch 1/5
79/79 [==============================] - 12s 55ms/step - loss: 0.9457 - accuracy: 0.6841
Epoch 2/5
79/79 [==============================] - 4s 49ms/step - loss: 0.1389 - accuracy: 0.9683
Epoch 3/5
79/79 [==============================] - 4s 49ms/step - loss: 0.0446 - accuracy: 0.9893
Epoch 4/5
79/79 [==============================] - 4s 49ms/step - loss: 0.0144 - accuracy: 0.9964
Epoch 5/5
79/79 [==============================] - 4s 50ms/step - loss: 0.0093 - accuracy: 0.9988
<keras.callbacks.History at 0x7fe9df04af50>
```

```
model.evaluate(test_images,test_labels)
```

```
12/12 [==============================] - 0s 22ms/step - loss: 0.9401 - accuracy: 0.6909
[0.9401404857635498, 0.6908602118492126]
```

✓ 0s completed at 5:12 PM