ML / DL Engineer – Test Task

Train a neural-network based model (Train.py)

To accomplish this task, python 3.7 language and Tensorflow 2.0 are used as following:

1- Retrieve the names of images from obs: obs1=np.load(path+"obs.npy")
2- load the images and store it in a list in the same order of obs
3- Store the corresponding actions and rewards
4- Split the dataset into training and testing.
5- Use the actions as labels
6- Build and fit the convolutional Neural network
7- Use the testing set to measure the accuracy
8- Save the model

Evaluate / demonstrate how well the model works (Test.py)

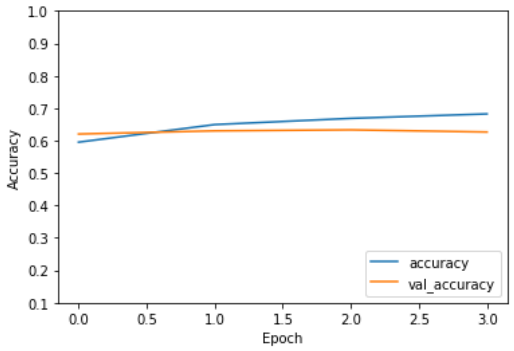
The library gym[Atari] is used to simulate and evaluate the new model as following:

1- Load the model
2- Get the new observations using reset and step in the breakout environment
3- Convert the observation to grayscale
4- Resize the observation to 84*84
5- Use queue of length 8 to enqueue the new observations
6- Dequeue it and max each pair to get four 2D arrays
7- Store the results in 3D array, where each 2D array is stored in different channel
8- Predict the action using the trained model where the input is the 3D array
9- The model is evaluated based on the average of the rewords

Experiments and Results

Several experiments are performed to find the best setting as summarized in the following table:

Dataset	# Episodes	Accuracy	Avg. Rewards	Max Reward
Random	5	25.0	1.2	3
5k	5	51.0	4.2	6
100K	5	60.0	21.8	29
200K	5	62.0	22.2	53
300K	5	62.8	29.8	56
400K	5	63.2	32.0	46
410K	5	**68.1**	**42.6**	**72**
500k	5	63.1	30.0	40



Sample output

Conclusions:

The results could be improved more by applying one or more of the following suggestions:

1- Using bigger dataset
2- Fine tuning the parameters and the architecture
3- Embedding the rewards during the training as a weights in the fit method: model.fit(train_images, train_labels, sample_weight=R, …..)
4- Using Deep Q-Network (DQN)

Bonus Task

We can Import this model into TensorFlow.js and can be used by a JavaScript or by Node.js as following

1- Convert it using tensorflowjs_converter

```
tensorflowjs_converter --input_format keras\  my_model.h5\  tfjs_target_dir
```

2- Load the model

```
import * as tf from '@tensorflow/tfjs;'
const model = await tf.loadLayersModel('model.json');
```

3- Predict the actions for the new observations

```
const cv = require('opencv;('
cv.readImage('./img/myImage.jpg', function (err, observation} (
  if (err} (
    throw err;
  }

const action = model.predict(preprocess(observation));
```