

Assignment no :1

Name : CHO VATIYA JAY MAHESHBHAI

Student ID : 1161980

Course: COMP5313 Artificial Intelligence

Topic: Multi-Class Connectionist AI

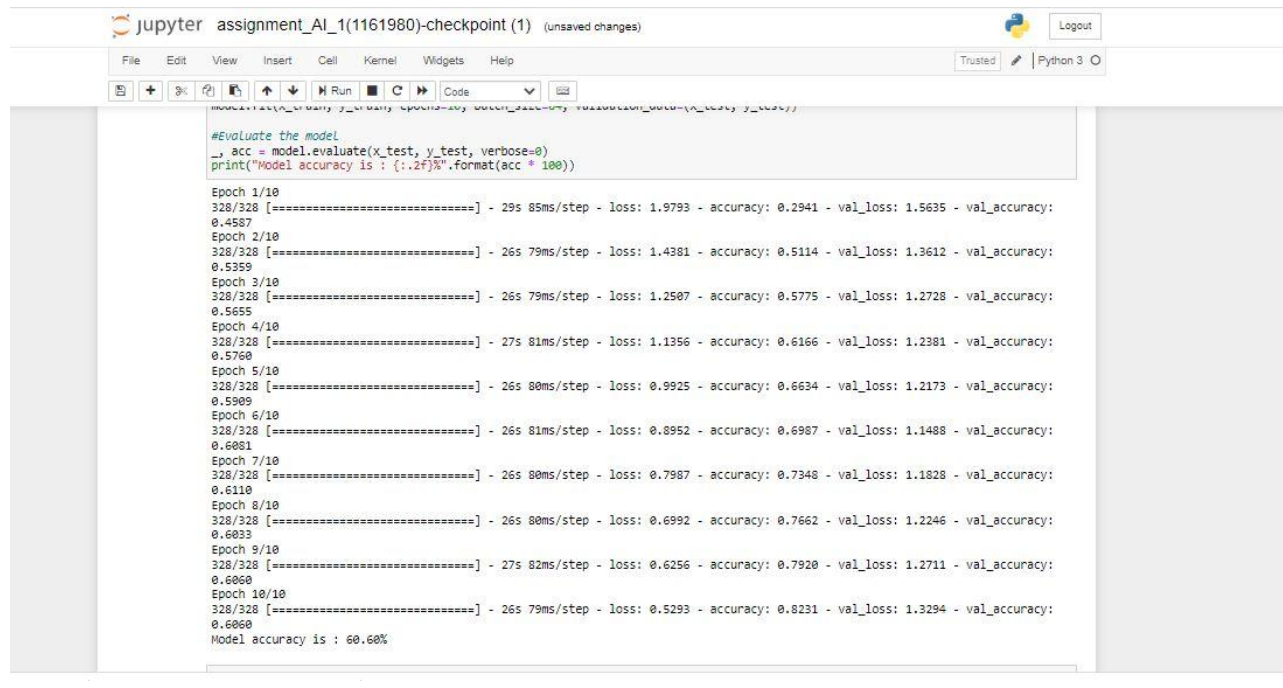
Multi-Class Connectionist AI :

There are several steps which i have followed to complete this task. Discussing about first one in which i have pre-process the dataset of animals. To elaborate, split the data into training and testing sets after pre-processing the photos to shrink and convert them to a uniform size.

In the following stage, i have created the model by defining the number of layers, activation functions, optimizers, and loss functions are all examples of this. With the help of optimizer and loss function specified during the second stage which are used to train the model. This phase will also update the model's weights depending on the training data. In the next stage, i have used the testing data to assess the model's correctness. This can be accomplished by comparing the expected and actual class labels.

Last but not the least, in order to improve the accuracy of our multi-class classifier, i have Fine-tune the model by modifying the architecture or adjusting the hyperparameters. If the accuracy is not adequate, Last two Steps 3 and 4 should be repeated until the accuracy is adequate.

Outputs :1.



The screenshot shows a Jupyter Notebook interface with a code cell containing the following Python code and its output:

```
#Evaluate the model
_, acc = model.evaluate(x_test, y_test, verbose=0)
print("Model accuracy is : {:.2f}%".format(acc * 100))
```

Epoch 1/10
328/328 [=====] - 29s 85ms/step - loss: 1.9793 - accuracy: 0.2941 - val_loss: 1.5635 - val_accuracy: 0.4587
Epoch 2/10
328/328 [=====] - 26s 79ms/step - loss: 1.4381 - accuracy: 0.5114 - val_loss: 1.3612 - val_accuracy: 0.5359
Epoch 3/10
328/328 [=====] - 26s 79ms/step - loss: 1.2507 - accuracy: 0.5775 - val_loss: 1.2728 - val_accuracy: 0.5655
Epoch 4/10
328/328 [=====] - 27s 81ms/step - loss: 1.1356 - accuracy: 0.6166 - val_loss: 1.2381 - val_accuracy: 0.5760
Epoch 5/10
328/328 [=====] - 26s 80ms/step - loss: 0.9925 - accuracy: 0.6634 - val_loss: 1.2173 - val_accuracy: 0.5909
Epoch 6/10
328/328 [=====] - 26s 81ms/step - loss: 0.8952 - accuracy: 0.6987 - val_loss: 1.1488 - val_accuracy: 0.6081
Epoch 7/10
328/328 [=====] - 26s 80ms/step - loss: 0.7987 - accuracy: 0.7348 - val_loss: 1.1828 - val_accuracy: 0.6110
Epoch 8/10
328/328 [=====] - 26s 80ms/step - loss: 0.6992 - accuracy: 0.7662 - val_loss: 1.2246 - val_accuracy: 0.6033
Epoch 9/10
328/328 [=====] - 27s 82ms/step - loss: 0.6256 - accuracy: 0.7920 - val_loss: 1.2711 - val_accuracy: 0.6060
Epoch 10/10
328/328 [=====] - 26s 79ms/step - loss: 0.5293 - accuracy: 0.8231 - val_loss: 1.3294 - val_accuracy: 0.6060
Model accuracy is : 60.60%

Overall, it can be recapitulated that the output stipulates the information regarding the progress of the training process, with the model's loss and accuracy reported after every epochs. Here I have taken 10 epochs to show accuracy in which you can see in the picture that loss is inversely proportional to the value accuracy. Eventually, the model accuracy is presented at the conclusion which is 60.60%, which indicates how well the model generalizes to previously unknown data. The outcome may fluctuate significantly across runs just because of unpredictability of the training process.

Outputs :2 According to the trained model, the predicted class output is elephant.



The screenshot shows a Jupyter Notebook interface with a code cell containing the following Python code and its output:

```
In [14]: # classify image of elephant
image = keras.preprocessing.image.load_img("C:\\Users\\dell\\Desktop\\animal\\raw-img\\elefante\\elephant.jpg", target_size=(32,
image_array = keras.preprocessing.image.img_to_array(image)
predictions = model.predict(image_array[np.newaxis, ...])
predicted_class = label_encoder.inverse_transform(np.array([np.argmax(predictions)]))
print("Your Predicted class is:", predicted_class)
```

Your Predicted class is: ['elefante']

In []: