

Ζητούμενο 2

Να αναπτύξετε ένα deep learning μοντέλο που θα κατηγοριοποιεί εικόνες γατών και σκυλιών σε 2 κατηγορίες (γάτες, σκύλοι). Χρησιμοποιήστε το Kaggle dataset. Στοχεύστε σε ένα επίπεδο ακρίβειας πάνω από 85%. Αιτιολογήστε όλες τις αποφάσεις σας. Μπορείτε να χρησιμοποιήσετε την βιβλιοθήκη Tensorflow (το dataset το παρέχει έτοιμο αυτή η βιβλιοθήκη) ή Pytorch της Python ή οποιαδήποτε άλλη επιθυμείτε σε οποιαδήποτε γλώσσα προγραμματισμού θέλετε.

Σημείωση: το παρακάτω documentation ακολουθεί τη ροή του κώδικα

Για την εκπόνηση του παραπάνω task αποφάσισα να χρησιμοποιήσω το jupyter notebook (VS Code embedded), καθώς και την βιβλιοθήκη tensorflow, η οποία χρησιμοποιείται για ανάπτυξη deep learning που είναι μια τεχνική του machine learning μοντέλων, μέσω νευρωνικών δικτύων. Μετά την εγκατάσταση των απαραίτητων βιβλιοθηκών: tensorflow, numpy για μαθηματικούς υπολογισμούς, cv2 για την τροποποίηση και διαχείριση εικόνων, matplotlib για στατιστικά διαγράμματα, os για διαχείριση των αρχείων. Ύστερα όρισα το directory στο οποίο βρίσκονται η φωτογραφίες του Kaggle dataset (γάτων και σκύλων) που θα χρησιμοποιηθεί για την εκπαίδευση του μοντέλου. Συνεχίζοντας, περιόρισα τους αποδεκτούς τύπους φωτογραφιών σε jpg, jpeg, bmp, png μέσω ενός array.

Όσον αφορά την φόρτωση του αρχείων χρησιμοποίησα την συνάρτηση `image_dataset_from_directory` του keras η οποία φτιάχνει αυτόματα τις κλάσεις των δεδομένων με βάση τους φακέλους που έχουμε στο αρχείο που ορίσαμε (εδώ έχουμε την test1 και train), καθώς εκτελεί και το απαραίτητο pre-processing των δεδομένων πριν τα φορτώσει (τα δεδομένα δεν φορτώνονται εξ αρχής, αλλά on-the-fly). Αφού φτιάξω το `data_iterator` το οποίο παίρνει τα δεδομένα από τους φακέλους (one batch at a time), ορίζω και το batch ως το σύνολο δεδομένων που λαμβάνεται εκάστη φορά από τον `data_iterator` (κάθε batch εκφράζεται ως ένα numpy array από 0 και 1, όπου 0 εικόνες και 1 labels που είναι arrays αποτελούμενα από 0 και 1 όπου 0 αν ανήκει στη μία κλάση και 1 αν ανήκει στην άλλη κλάση).

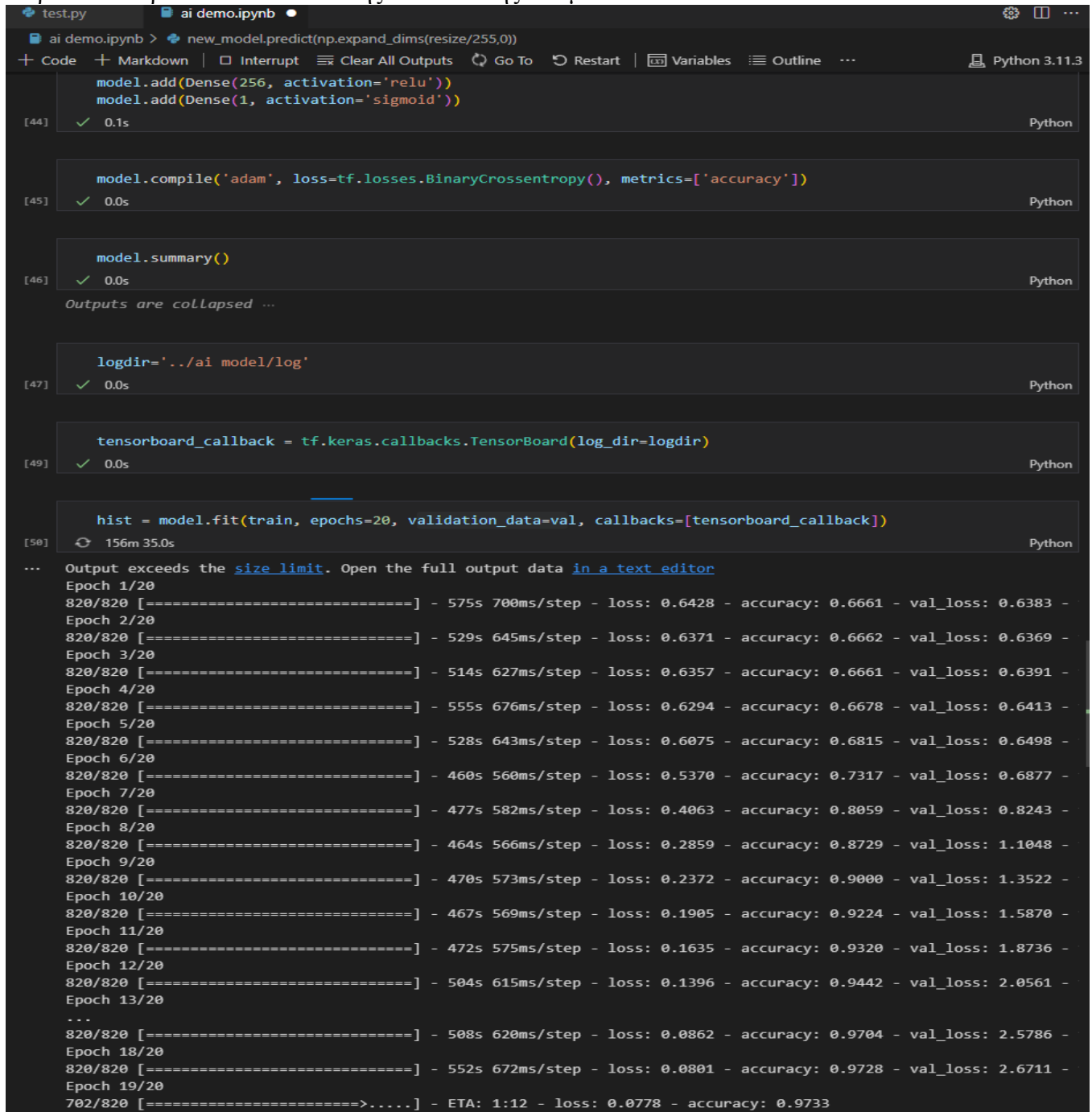
Το επόμενο βήμα αφορά το pre-processing των δεδομένων όπου κάνω ένα scaling των δεδομένων για εξοικονόμηση πόρων. Διαιρούμε εικόνες με το 255 ώστε να δημιουργήσουμε συνταγμένες με εύρος τιμών [0, 1] με ένα lambda function στη συνάρτηση `map` η οποία μας βοηθάει να κάνουμε αυτόν τον υπολογισμό κατά την φόρτωση των δεδομένων.

Συνεχίζοντας στο deep learning μοντέλο χρησιμοποίησα το Sequential API του keras (1 είσοδος και 1 έξοδος και τα layers προσθέτονται ακολουθιακά), καθώς και μερικά layers το Conv2D (spatial convolution over images), MaxPooling2D (condensing images), Dense, Flatten. Μετά όρισα ένα μοντέλο, ενώ ύστερα πρόσθεσα τα layers που θα αναλύσω παρακάτω. Μέσω της συνάρτησης `add` προσθέτουμε τα layers που επιθυμούμε στο μοντέλο. Πρώτα προσθέτουμε ένα convolution με 16 φίλτρα των 3x3 pixel, εξετάζοντας 1 pixel τη φορά με relu activation (διατηρούμε μόνο τις θετικές τιμές θέτωντας 0 τις αρνητικές συμπεριλαμβάνοντας έτσι μη γραμμικά patterns στην έξοδο) και μεγέθους 256x256 pixels σε 3 channels. Στο δεύτερο βήμα, η MaxPooling2D θα πάρει τη μέγιστη τιμή μιας 2x2 περιοχής από την έξοδο που θα επιστρέψει το πρώτο layer. Επαναλαμβάνουμε τη διαδικασία δύο ακόμα φορές πριν κάνουμε Flatten() την έξοδο ώστε να έχουμε 256 τιμές και να έχουμε 1 έξοδο τελικά μέσω της Dense() με sigmoid activation (μετατρέπει οποιαδήποτε έξοδο σε ένα εύρος μεταξύ 0 και 1).

Επιπροσθέτως, θα χρειαστεί να κάνουμε compile το μοντέλο με το adam optimizer, ορίζοντας το loss και το metrics που θα δείξει την ευστοχία του.

Όσον αφορά την εκπαίδευση του μοντέλου δημιούργησα ένα αρχείο log για την παρακολούθηση του μοντέλου κατά την εξέλιξη της εκπαίδευσής του. Συνεχίζοντας μέσω της fit function έθεσα το μοντέλο στη διαδικασία της εκπαίδευσης σε 20 εποχές.

Παρακάτω παραθέτω screenshot της εκπαίδευσης του μοντέλου.



```
test.py | ai demo.ipynb
ai demo.ipynb > new_model.predict(np.expand_dims(resize(255,0))
+ Code + Markdown | Interrupt Clear All Outputs Go To Restart Variables Outline Python 3.11.3

[44] ✓ 0.1s Python
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

[45] ✓ 0.0s Python
model.compile('adam', loss=tf.losses.BinaryCrossentropy(), metrics=['accuracy'])

[46] ✓ 0.0s Python
model.summary()
Outputs are collapsed ...

[47] ✓ 0.0s Python
logdir='../ai model/log'

[49] ✓ 0.0s Python
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)

[50] 156m 35.0s Python
hist = model.fit(train, epochs=20, validation_data=val, callbacks=[tensorboard_callback])

... Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/20
820/820 [=====] - 575s 700ms/step - loss: 0.6428 - accuracy: 0.6661 - val_loss: 0.6383 -
Epoch 2/20
820/820 [=====] - 529s 645ms/step - loss: 0.6371 - accuracy: 0.6662 - val_loss: 0.6369 -
Epoch 3/20
820/820 [=====] - 514s 627ms/step - loss: 0.6357 - accuracy: 0.6661 - val_loss: 0.6391 -
Epoch 4/20
820/820 [=====] - 555s 676ms/step - loss: 0.6294 - accuracy: 0.6678 - val_loss: 0.6413 -
Epoch 5/20
820/820 [=====] - 528s 643ms/step - loss: 0.6075 - accuracy: 0.6815 - val_loss: 0.6498 -
Epoch 6/20
820/820 [=====] - 460s 560ms/step - loss: 0.5370 - accuracy: 0.7317 - val_loss: 0.6877 -
Epoch 7/20
820/820 [=====] - 477s 582ms/step - loss: 0.4063 - accuracy: 0.8059 - val_loss: 0.8243 -
Epoch 8/20
820/820 [=====] - 464s 566ms/step - loss: 0.2859 - accuracy: 0.8729 - val_loss: 1.1048 -
Epoch 9/20
820/820 [=====] - 470s 573ms/step - loss: 0.2372 - accuracy: 0.9000 - val_loss: 1.3522 -
Epoch 10/20
820/820 [=====] - 467s 569ms/step - loss: 0.1905 - accuracy: 0.9224 - val_loss: 1.5870 -
Epoch 11/20
820/820 [=====] - 472s 575ms/step - loss: 0.1635 - accuracy: 0.9320 - val_loss: 1.8736 -
Epoch 12/20
820/820 [=====] - 504s 615ms/step - loss: 0.1396 - accuracy: 0.9442 - val_loss: 2.0561 -
Epoch 13/20
...
820/820 [=====] - 508s 620ms/step - loss: 0.0862 - accuracy: 0.9704 - val_loss: 2.5786 -
Epoch 18/20
820/820 [=====] - 552s 672ms/step - loss: 0.0801 - accuracy: 0.9728 - val_loss: 2.6711 -
Epoch 19/20
702/820 [=====>.....] - ETA: 1:12 - loss: 0.0778 - accuracy: 0.9733
```

*Επειδή είναι η πρώτη μου επαφή με deep learning μοντέλο ως βασική πηγή χρησιμοποίησα το βίντεο: <https://www.youtube.com/watch?v=jztwpsIzEGc>