**Brain Anomaly Detection**

(competitie Kaggle)

Bratan Radu-George

Facultatea de Matematică și Informatică

Grupa 242

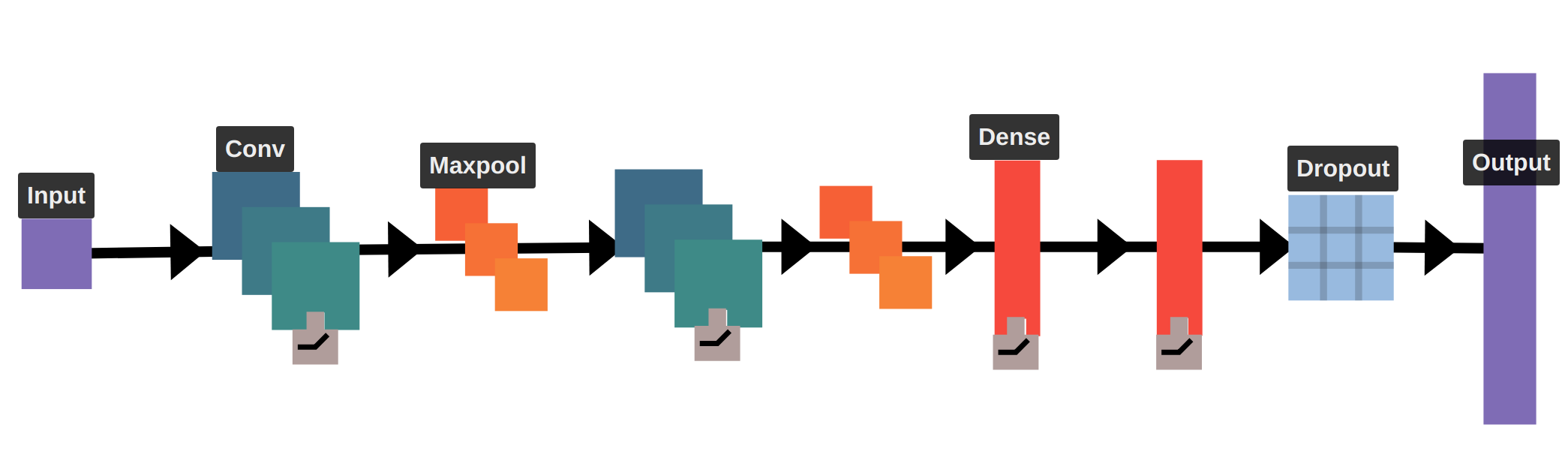
**1. Introducere**

Proiectul de pe platforma Kaggle presupune discriminarea între două clase de tomografii cerebrale, una care contine anomalii (etichetată cu 1) și una normală (eitchetată cu 0). Pentru a efectua clasificarea, am creat două modele antrenate pe 15000 de imagini alb-negru de dimensiuni 224 pe 224 de pixeli. Validarea s-a făcut pe 2000 de imagini, iar testarea pe 5149.

**2. Modelele CNN**

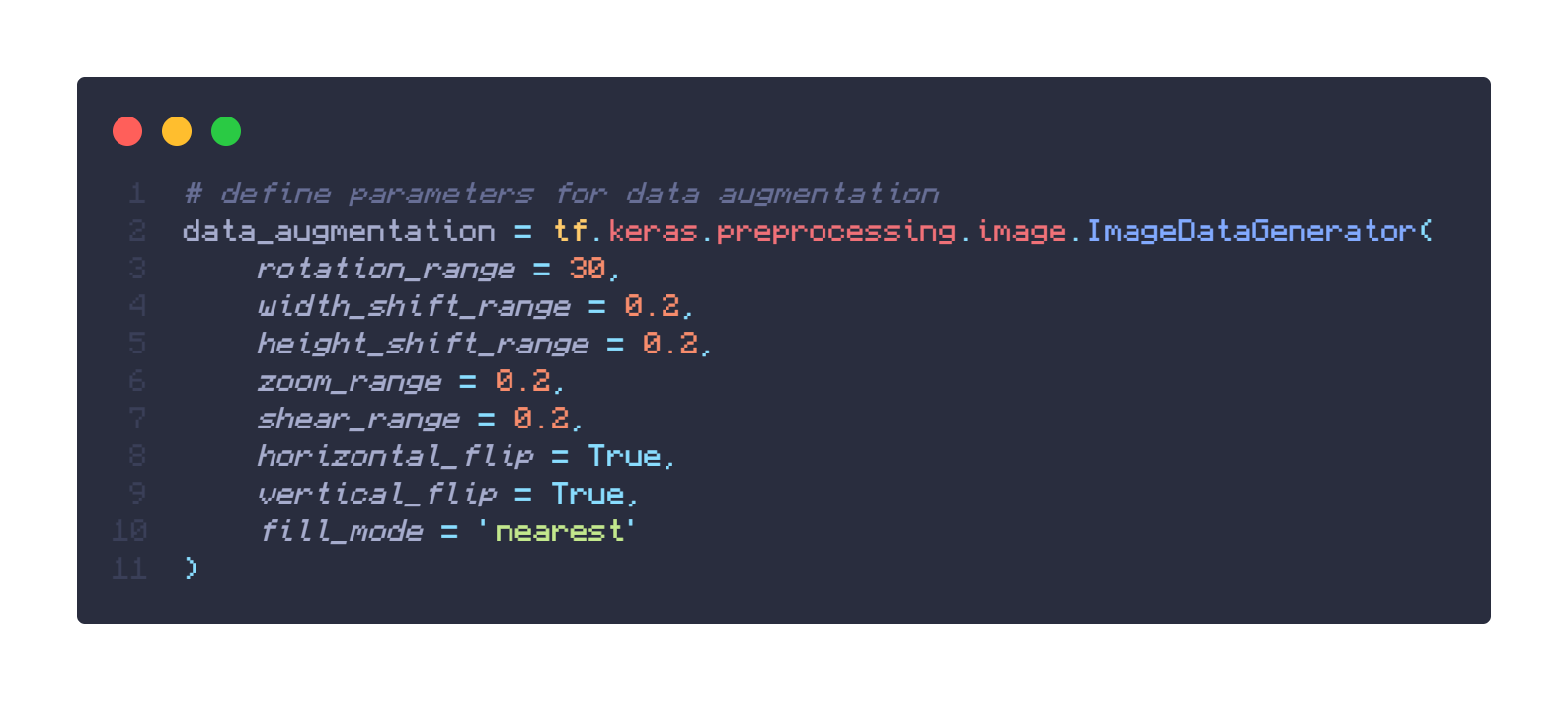
**2.1 Descriere**

O retea neuronală convolutională (CNN) este un tip de retea neuronală care este specializată în procesarea datelor aranjate sub formă de matrice, precum imaginile. Un CNN are de obicei trei tipuri principale de straturi: straturi convolutionale, straturi de grupare (pooling) și straturi complet conectate. Stratul convolutional este elementul de bază al unui CNN, unde au loc majoritatea calculelor.



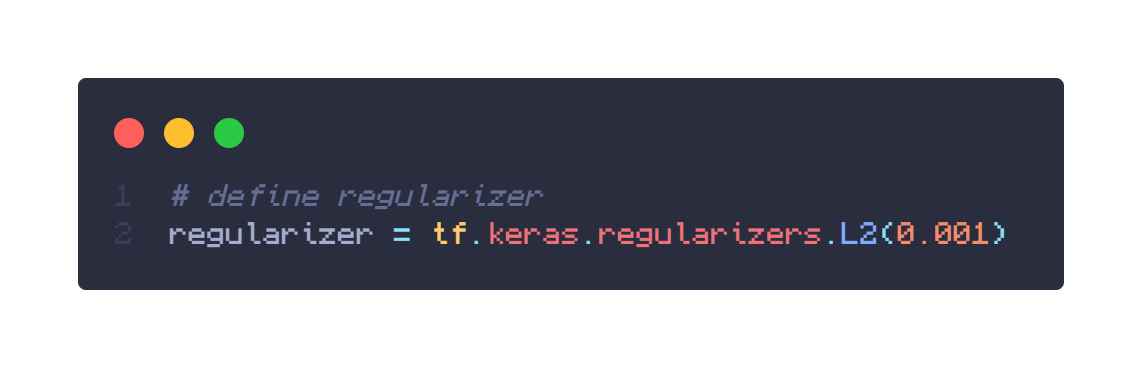
**2.2 Augmentare de date**

Pentru augmentarea datelor, am aplicat diferite transformări imaginilor, precum rotatii, schimbări de lătime și de înăltime, zoom, shear și răsturnare. Parametrul „fill\_mode” este setat la „nearest” astfel încât orice pixel nou creat în urma acestor transformări va fi umplut cu valoarea celui mai apropiat pixel.



**2.3 Regularizare**

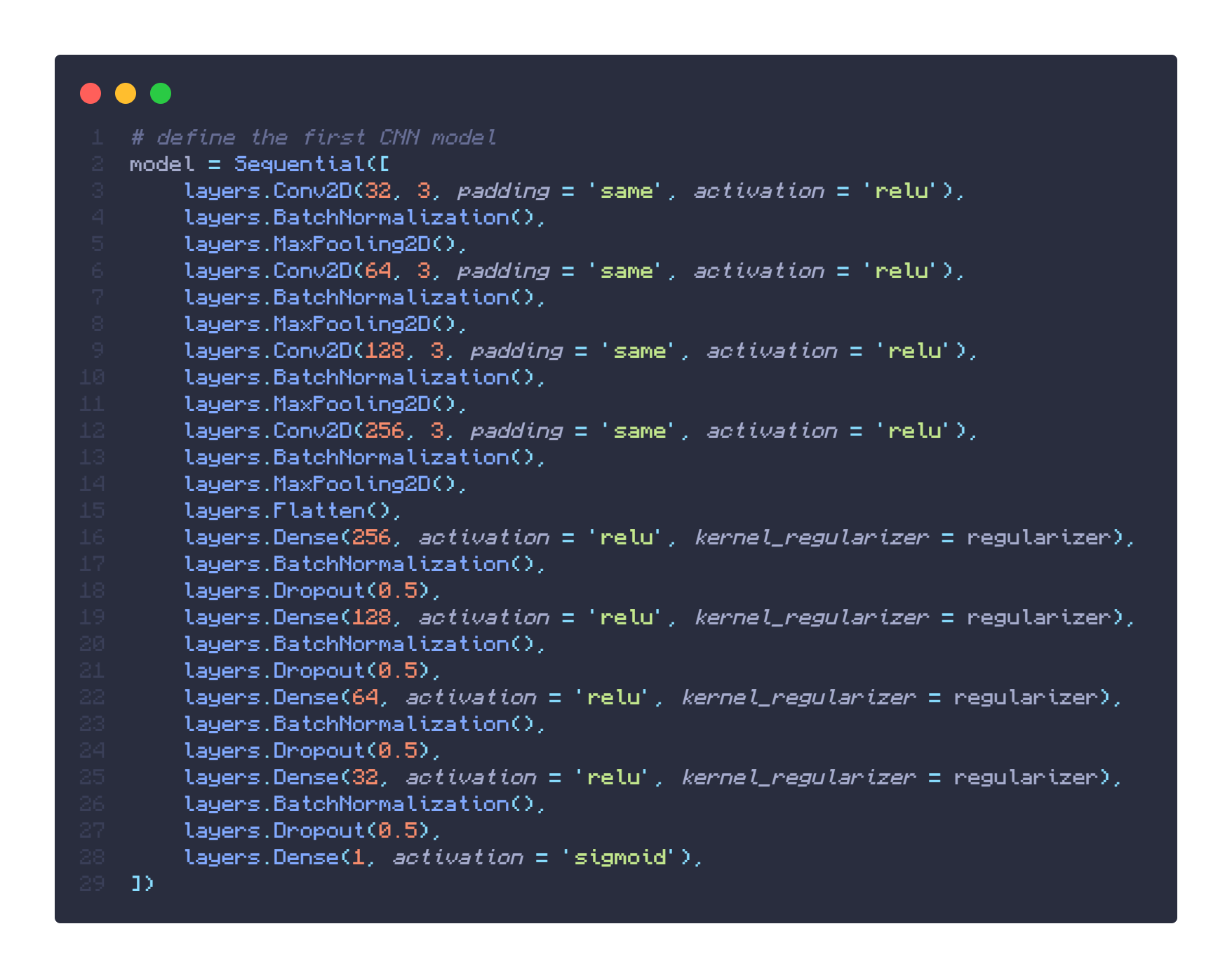
Am definit un regularizer de tipul L2 și un factor de regularizare de 0.001 pentru a fi aplicat weight-urilor unora dintre straturi, prevenind astfel overfitting-ul. Regularizatorul L2 se aplică weight-urilor straturilor complet conectate pentru ambele modele.



**2.4 Primul model CNN**

Pentru primul model, după fiecare strat convolutional (Conv2D) si complet conectat (Dense), am aplicat o functie de activare de tipul Rectified Linear Unit (ReLU), deoarece este simplă și eficientă din punct de vedere computational. Pe ultimul strat Dense am folosit Sigmoid, deoarece aici se potrivește cel mai bine.

Între straturile convolutionale si cele conectate, am adăugat un strat Flatten, care convertește datele într-un array unidimensional. Straturile BatchNormalization au fost folosite pentru a standardiza input-urile iar straturile MaxPooling au fost folosite pentru a reduce numărul de parametrii și, implicit, numărul de calcule efectuate. Straturile Dropout au fost folosite pentru a reduce overfitting-ul.



**2.5 Al doilea model CNN**

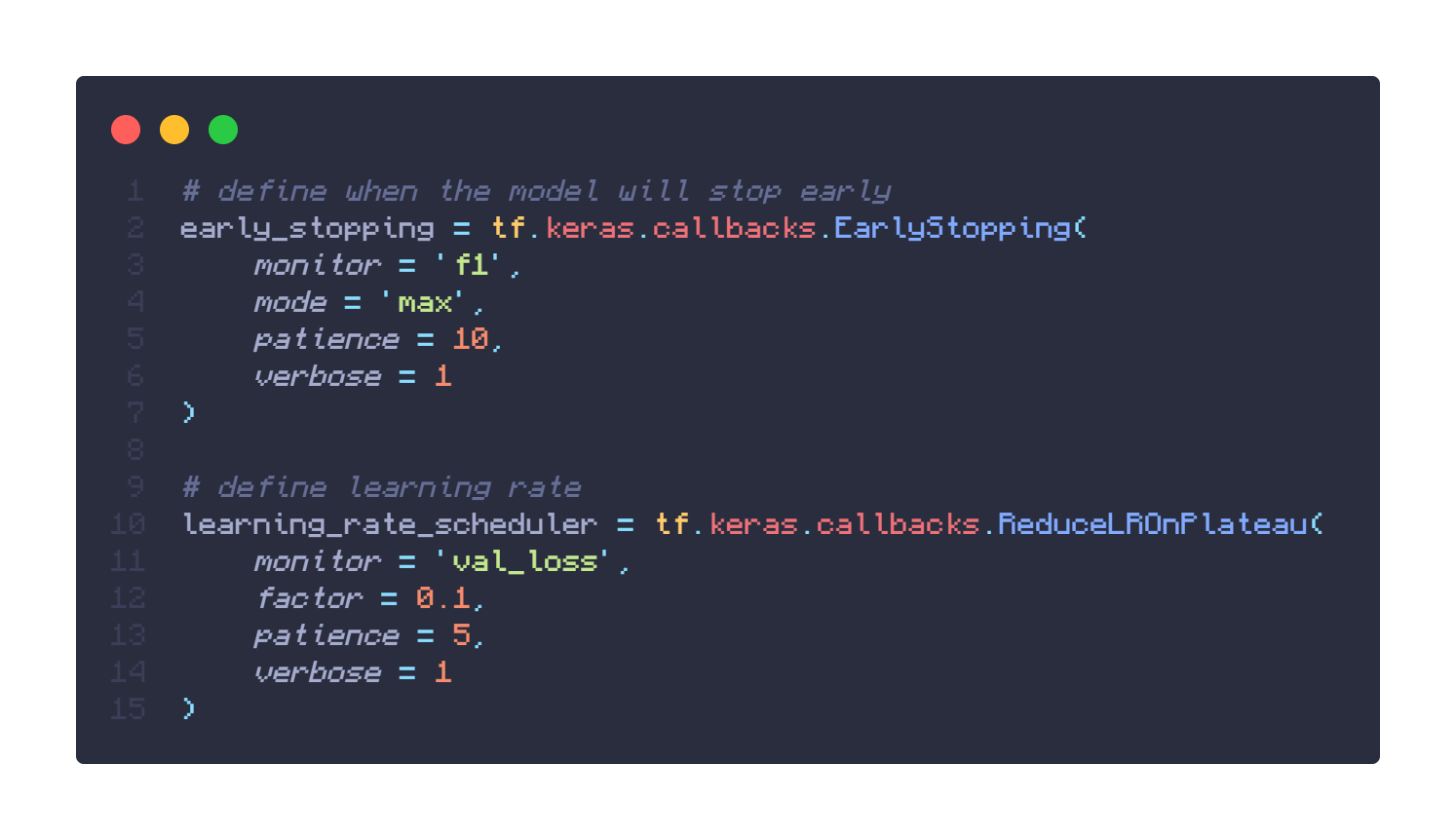
Al doilea model este format din patru straturi convolutionale cu functii de activare ReLU, urmate de straturi de max-pooling. După straturile convolutionale, modelul are un strat Flatten, două straturi Dense (primul cu functie de activare Sigmoid, al doilea cu functie ReLU) urmate de Dropout și un strat final Dense cu functie de activare Sigmoid.



**2.6 Apeluri**

Am definit două apeluri (callbacks) pentru antrenamentul modelului:

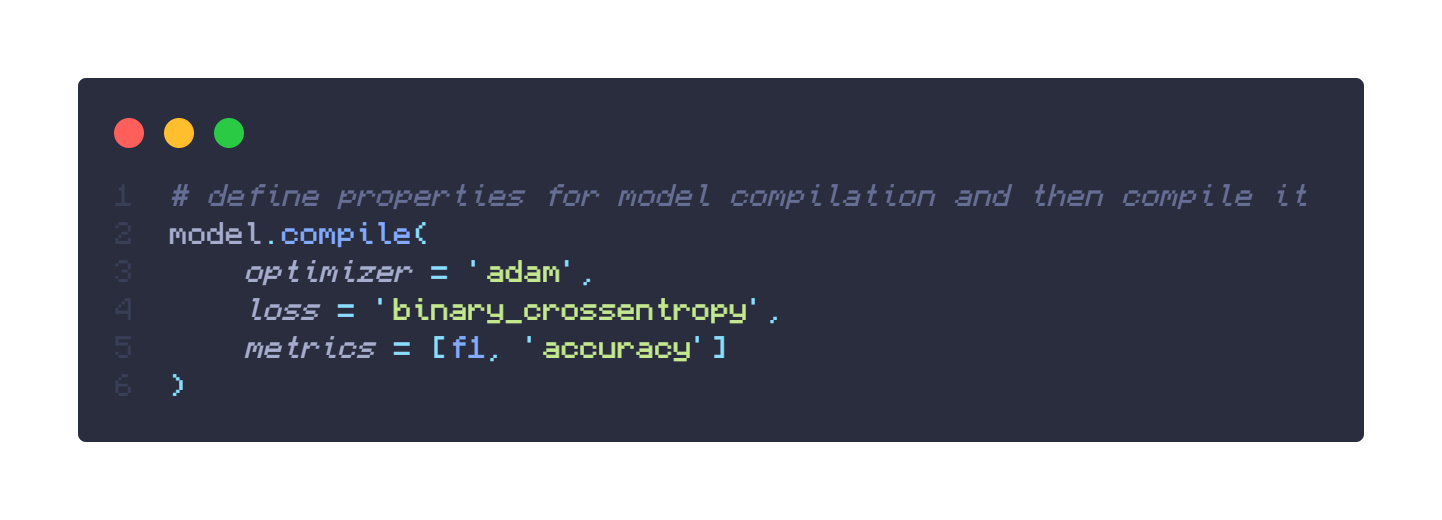
* EarlyStopping: oprește antrenamentul atunci când scorul F1 nu se îmbunătătește timp de 10 epoci consecutive; modul este setat la „max” pentru a maximiza scorul F1
* ReduceLROnPlateau: reduce rata de învătare cu un factor de 0.1 dacă validation loss nu se îmbunătătește timp de 5 epoci consecutive

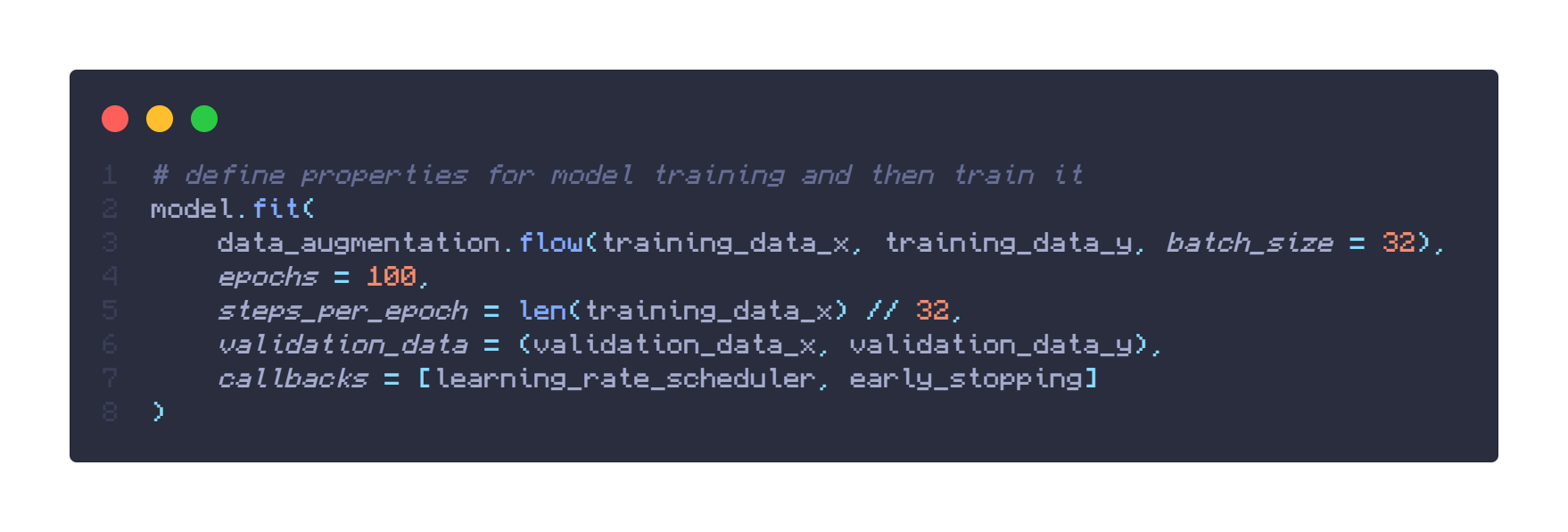


**2.7 Compilare și antrenare**

Modelul este compilat folosind optimizatorul „Adam” și măsurătorile F1 și „accuracy”.

Antrenarea este efectuată cu un batch size de 32 și un număr de 100 de epoci. Parametrul „steps\_per\_epoch” este setat la lungimea vectorului datelor de antrenament împărtită la batch size.





**2.8 Testare**

În timpul competitiei, am testat mai multe modele, dintre care doar acestea două au oferit rezultate favorabile. Pentru antrenare, am folosit atât platforma Google Collaborator, cât și Kaggle. Din testele efectuate de mine, am constatat că platforma Colab este mai lentă decât Kaggle. Cu toate acestea, pentru a reduce timpul de antrenare, am folosit ambele platforme în paralel. Primul model CNN a fost antrenat pe Google Colab, al doilea pe Kaggle.

Restul modelelor care nu au fost incluse au obtinut rezultate intre 0.0000e+00 și 0.2, deoarece fie citeam greșit datele și modelul se antrena practic pe nimic, fie reduceam dimensiunile imaginilor (de la 224x224 la 64x64, 32x32, chiar și 16x16), fie nu foloseam BatchNormalization, fie foloseam prea putini neuroni și prea putine epoci.

**3. Referinte**

* <https://www.datascience.stackexchange.com/questions/12851/how-do-you-visualize-neural-network-architectures>
* <https://www.deeplizard.com/learn/video/ZjM_XQa5s6s>
* <https://www.ibm.com/topics/convolutional-neural-networks>
* <https://www.keras.io/api/layers/core_layers/dense/>
* <https://www.keras.io/api/layers/reshaping_layers/flatten/>
* <https://www.machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/>
* <https://www.nickmccullum.com/python-deep-learning/convolutional-neural-network-tutorial/>
* <https://www.pyimagesearch.com/2021/07/19/pytorch-training-your-first-convolutional-neural-network-cnn/>
* <https://www.tensorflow.org/tutorials/images/cnn>
* <https://www.towardsdatascience.com/batch-normalisation-in-deep-neural-network-ce65dd9e8dbf>
* <https://www.towardsdatascience.com/understanding-cnn-convolutional-neural-network-69fd626ee7d4>