**Image Recognition**

**Group Members:**

➢ Huang Xie, 281685, huang.xie@tuni.fi

➢Liang Fang, 281684, liang.fang@tuni.fi

Training

1. *Load data and split dataset.*

Information of data contains image paths and corresponding labels is loaded as a pack from the dataset " genki4k ".

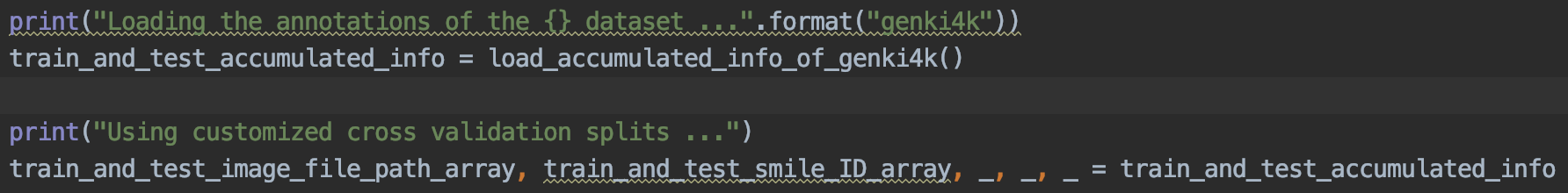


Fig. 1 Code of loading data and unpack data

Then, use **train\_test\_split** to split indices of training and test data randomly with the ratio of 80% / 20% respect to training / test, and split the total data into training set and test set.

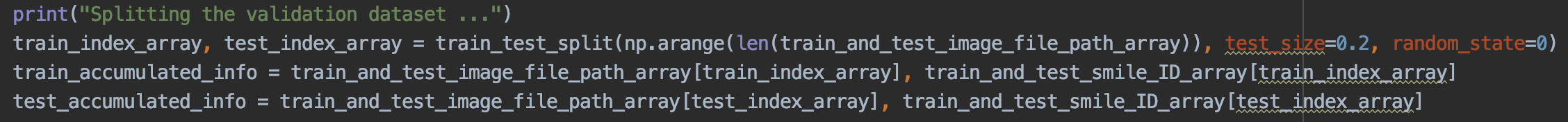


Fig. 2 Code of splitting dataset

1. *Model initialization.*

Considering from the requirement of execution time, we select the lightweight model "MobileNet" as the backbone model with pretrained weights from ImageNet dataset, then add a global average pooling layer after the feature maps generated from the backbone model to transform into a vector, finally a single-output dense layer with sigmoid is added as the last layer for classification. Since it is a binary classification task so that the loss function is "binary\_crossentropy", and the we use Adam optimizer with learning rate of 0.00032.



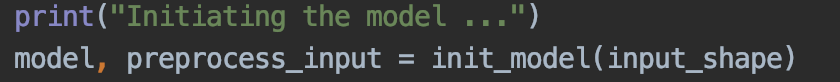
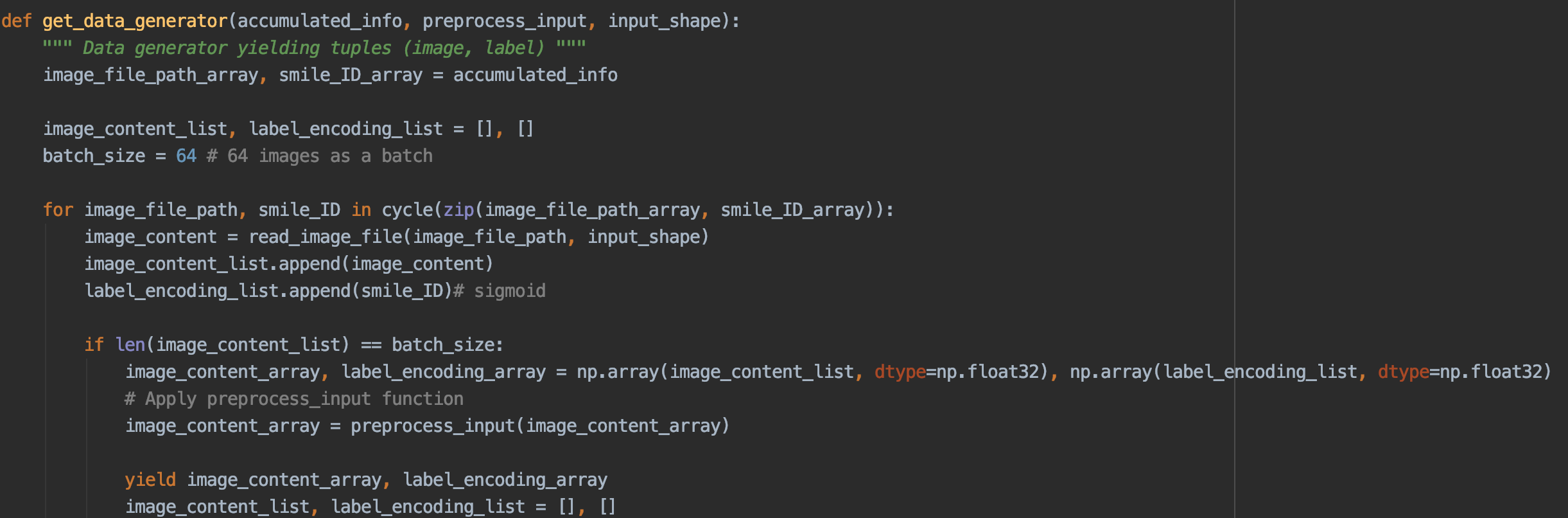


Fig. 3 Data generator

*Data generator.*

We define a cycling generator to yield images and label into the model for training. The output will be yielded with a batch of 64 samples sequentially from training set in an infinite loop,



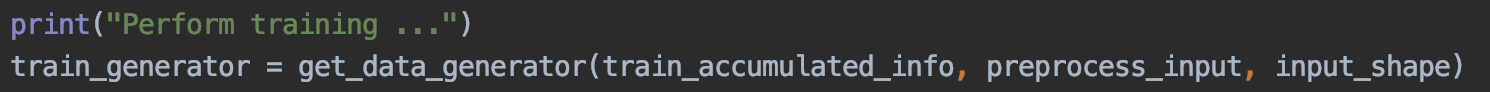


Fig. 4 Data generator

1. *Training and callbacks.*

We define several callback functions for every epoch, in which **Evaluator** is for inference of model in current model and storing accuracy in logs, **ModelCheckpoint** is for saving current model and **historylogger\_callback** is for plots of loss and accuracy with epoch.

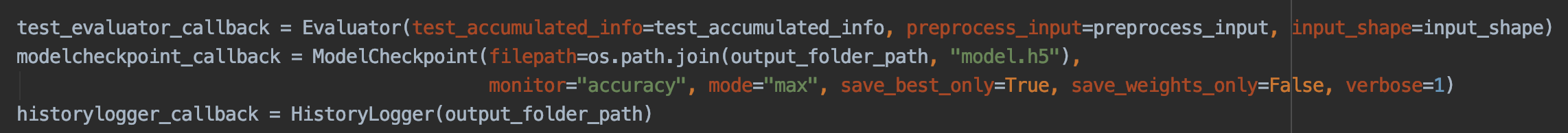


Fig. 5 Callbacks

Since we use **model.fit\_generator** as the feeding method, we set the 50 steps as one epoch so that the model can be fed with the whole 4k dataset with an epoch (50\*64=3200), and we set 50 epochs for training.

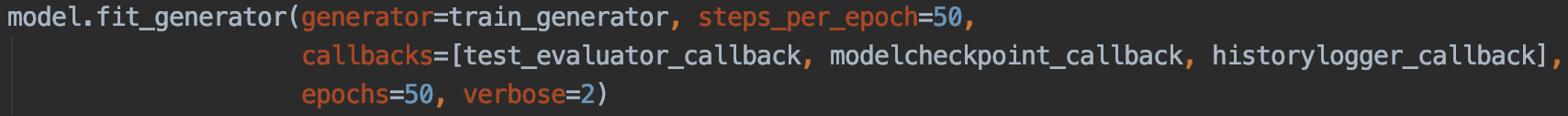
**

Fig. 6 Training

Fig. 7 shows the accuracy curve with each epoch. As we can see, the accuracy is about 90% after training with 50 epochs.

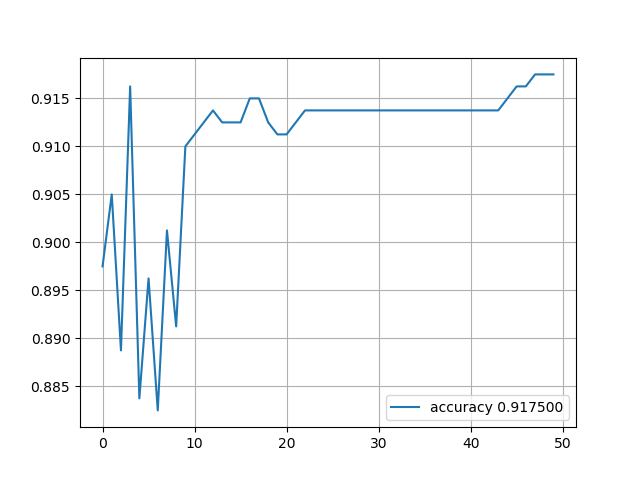
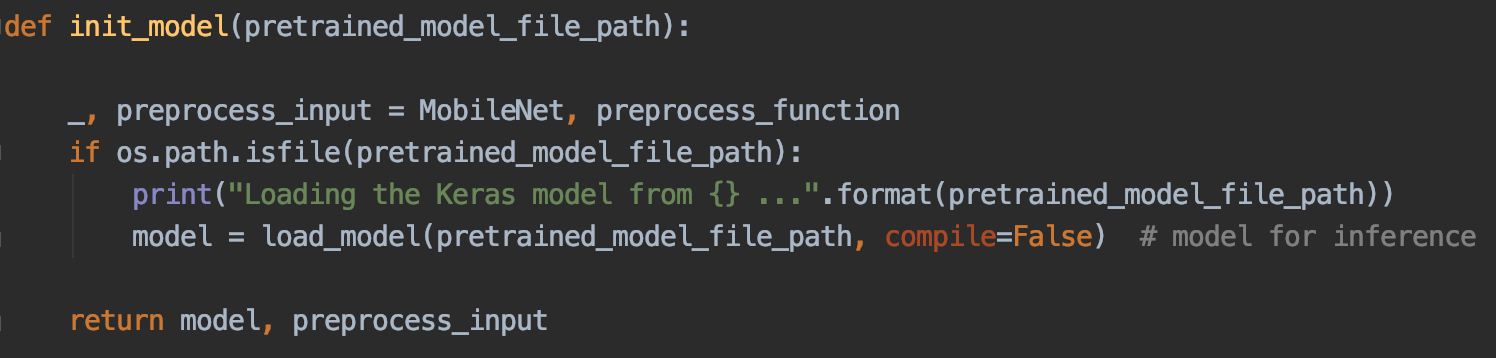


Fig. 7 Accuracy with epochs

Evaluation

1. *Load trained model.*

Trained model is loaded with **load\_model** from the given path. Dataset split is done with the same process as training previously.



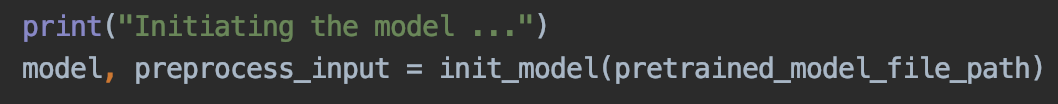


Fig. 8 Load trained model

1. *Accuracy evaluation and confusion matrix.*

Inference is done with the loaded model and obtain the prediction **y\_pred**, so **y\_pred** and **y\_true** are a pair for the construction of confusion matrix.

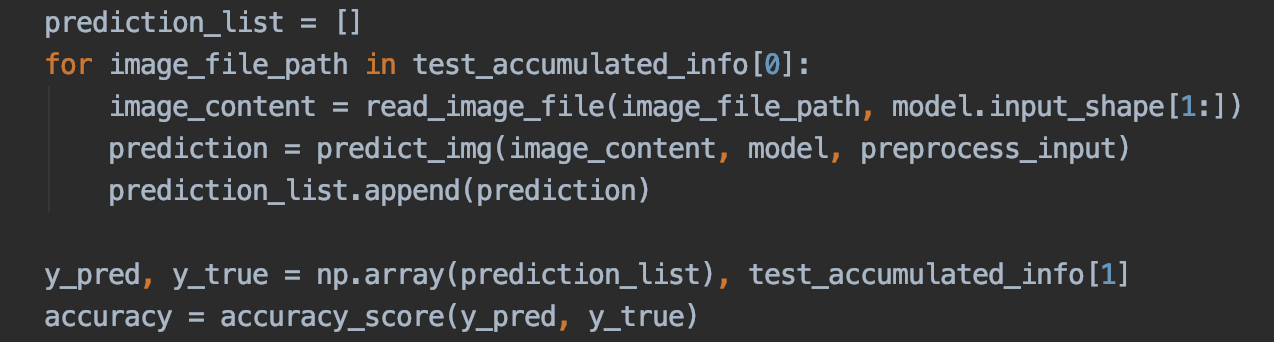


Fig. 9 Code of Convolutional network structure

Confusionmatrix is implemented with labels. As we can see from Fig. 11 right, majority of predictions are correct, that is, which are in the diagonal.

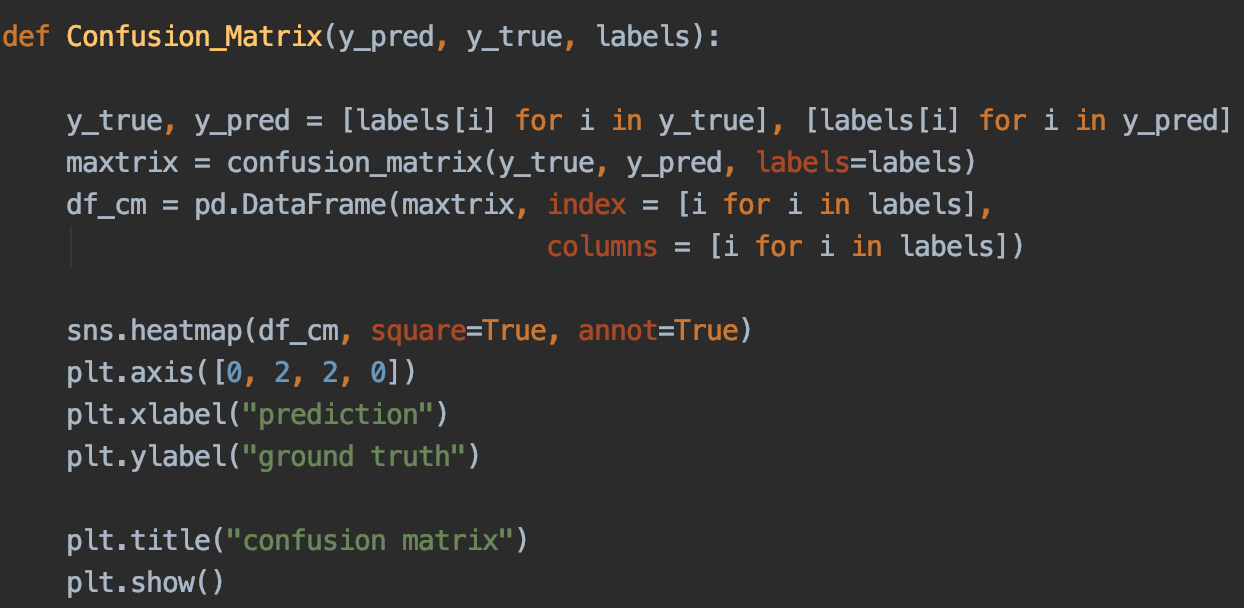


Fig. 10 Confusion Matrix

1. *Real-time evaluation with webcam.*

Real-time evaluation with webcam is based on the frames captured by camera, captured frame is flipped and as the input into the trained model for classification, then label the classification result as text in the captured frame and show it.

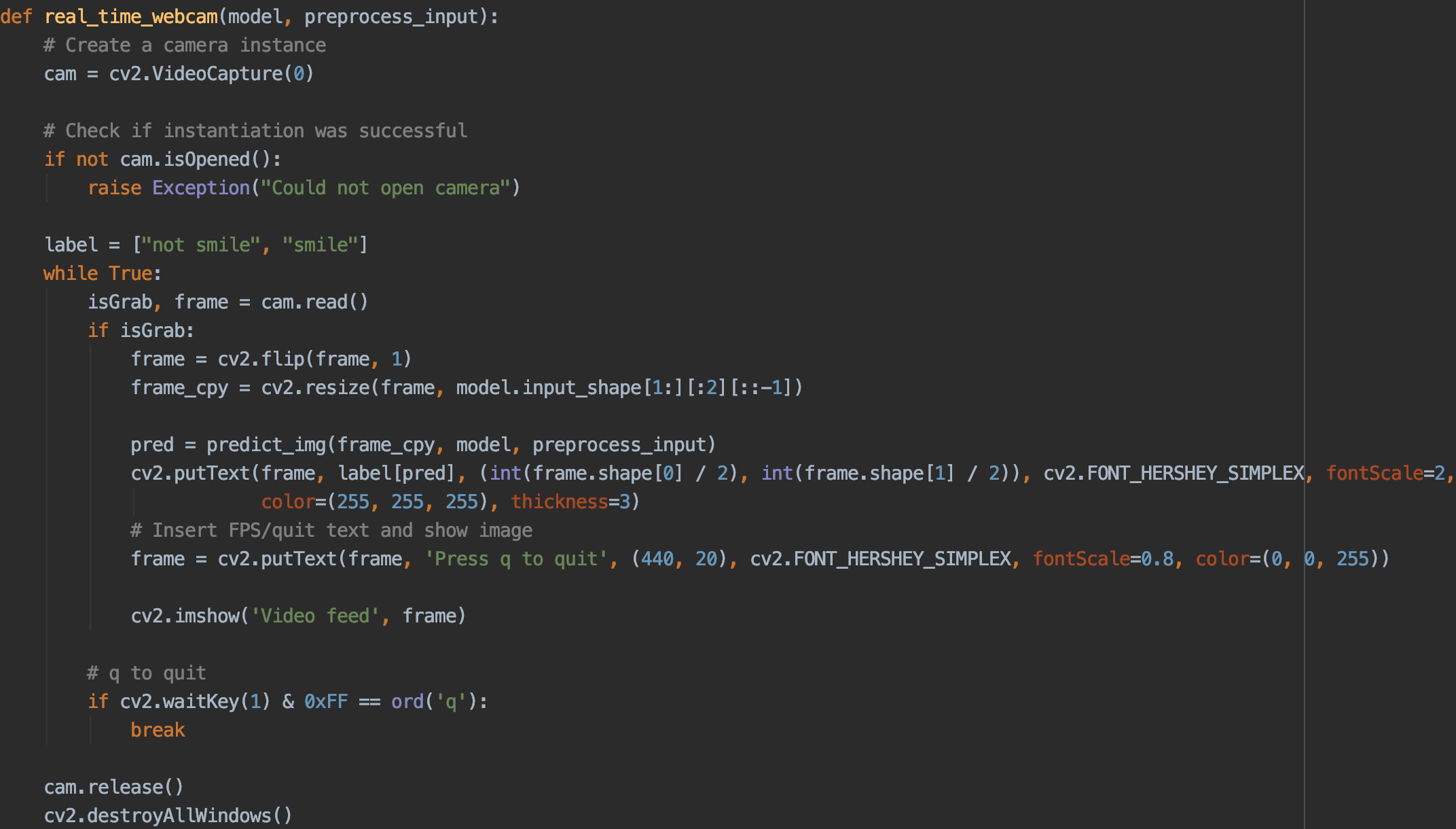


Fig. 11 Real-time evaluation with webcam



Fig. 12 Real-time evaluation demo