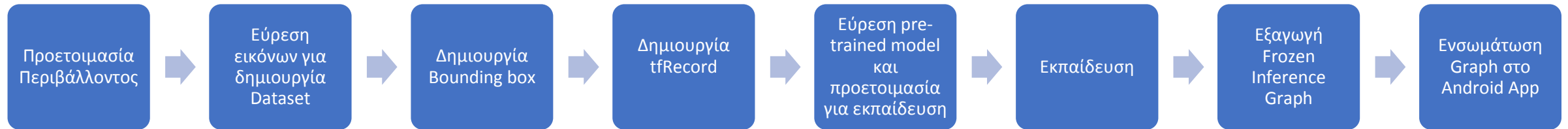




ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS

Πτυχιακή Εργασία Object-Detection

Βήματα για τη δημιουργία Object-detection App



Προετοιμασία Περιβάλλοντος

1. Εγκατάσταση Anaconda
2. Εγκατάσταση Tensorflow-GPU
3. Εγκατάσταση βιβλιοθηκών όπως pillow, lxml, matplotlib κλπ
4. Εγκατάσταση Protobuf (για διαμόρφωση μοντέλου και εκπαίδευση παραμέτρων)
5. Ρύθμιση Μεταβλητών Περιβάλλοντος(Environment variables)
6. Κλωνοποίηση του Tensorflow-Models repository για την αξιοποίηση του Object Detection μοντέλου

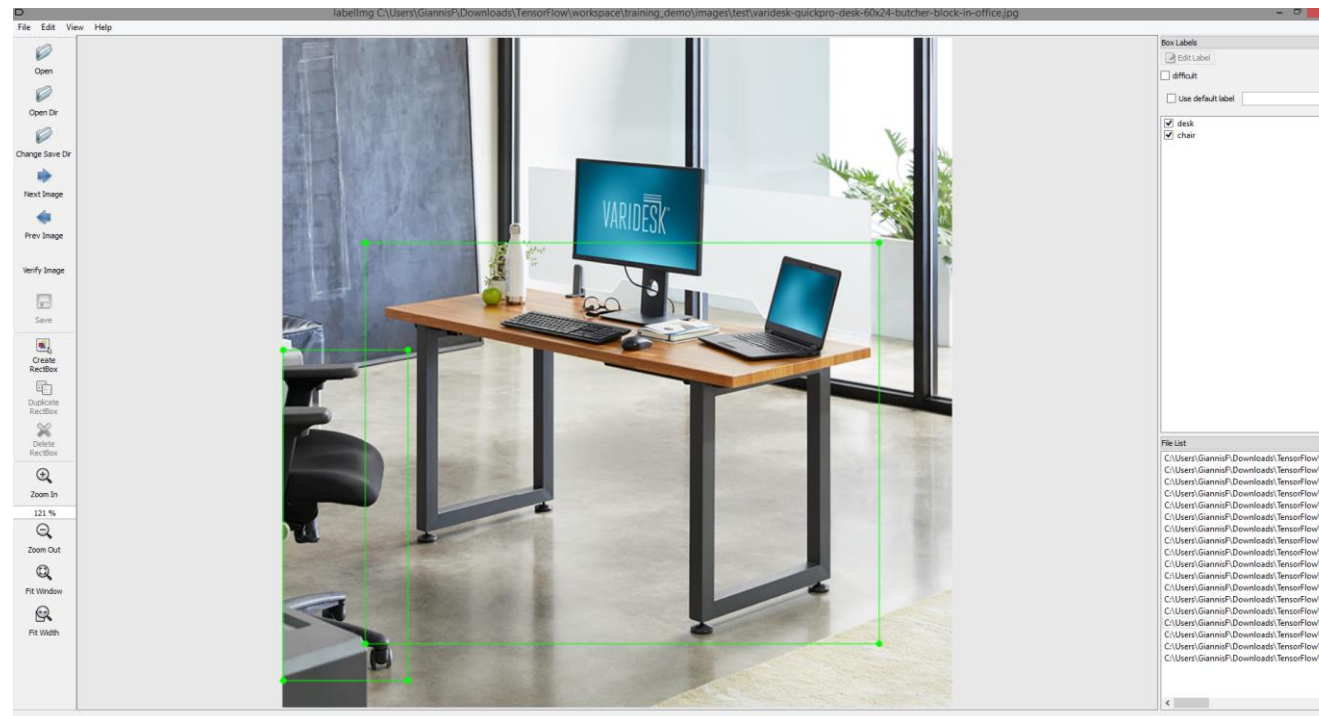
Εύρεση Εικόνων Για Δημιουργία Dataset

- Εύρεση εικόνων που περιέχουν το/τα αντικείμενα που θέλουμε για αναγνώριση. Οι εικόνες είναι κατεβασμένες μια προς μια είτε από το Google Images είτε από το PixaBay. Για κάθε αντικείμενο που θέλω να αναγνωρίσω συλλέγω περίπου 200 φωτογραφίες, οι οποίες περιλαμβάνουν διάφορες θεάσεις αυτού του αντικειμένου και διάφορες φωτεινότητες.



Δημιουργία Bounding box

- Κατέβασμα του LabelImg, πρόγραμμα το οποίο χρησιμεύει στη χειροκίνητη δημιουργία Bounding Box για κάθε αντικείμενο στις διάφορες εικόνες και δημιουργία XML αρχείων.



Δημιουργία tfRecord

- Για την εκπαίδευση του μοντέλου χρειάζεται τα δεδομένα εκπαίδευσης να είναι σε μια συγκεκριμένη μορφή, η οποία ονομάζεται tfRecords. Αυτό επιτυγχάνεται αν τα XML τα μετατρέψω σε CSV και στην συνέχεια τα CSV τα μετατρέψω σε tfRecords.

Εύρεση Pre-trained Model Και Προετοιμασία Για Εκπαίδευση

1. Εύρεση κατάλληλου pre-trained μοντέλου για εκπαίδευση. Εμάς μας ενδιαφέρει περισσότερο η ταχύτητα από ότι το accuracy αρά το ssd model είναι το καταλληλότερο. Το μοντέλο που αρχικά επιλέχθηκε ήταν το ssd_mobilenet_v1_coco
2. Τροποποίηση του .config file για το συγκεκριμένο μοντέλο.(Training Pipeline)

```
model {  
  (... Add model config here...)  
}  
  
train_config : {  
  (... Add train_config here...)  
}  
  
train_input_reader: {  
  (... Add train_input configuration here...)  
}  
  
eval_config: {  
}  
  
eval_input_reader: {  
  (... Add eval_input configuration here...)  
}
```

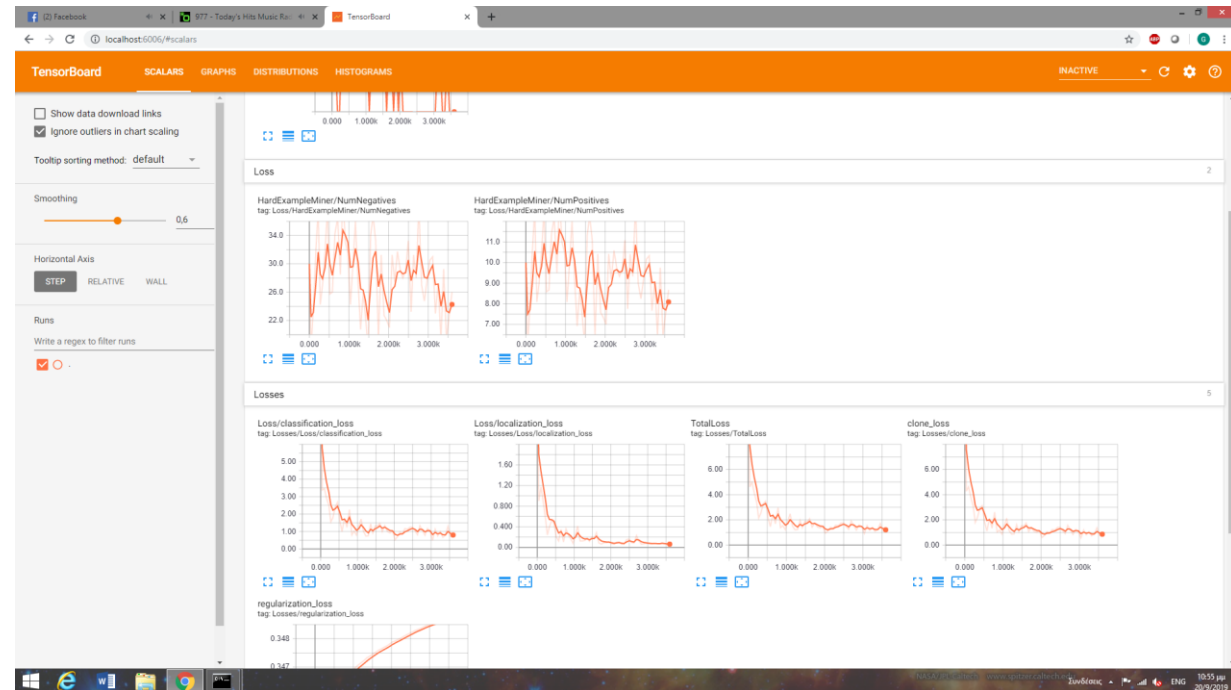
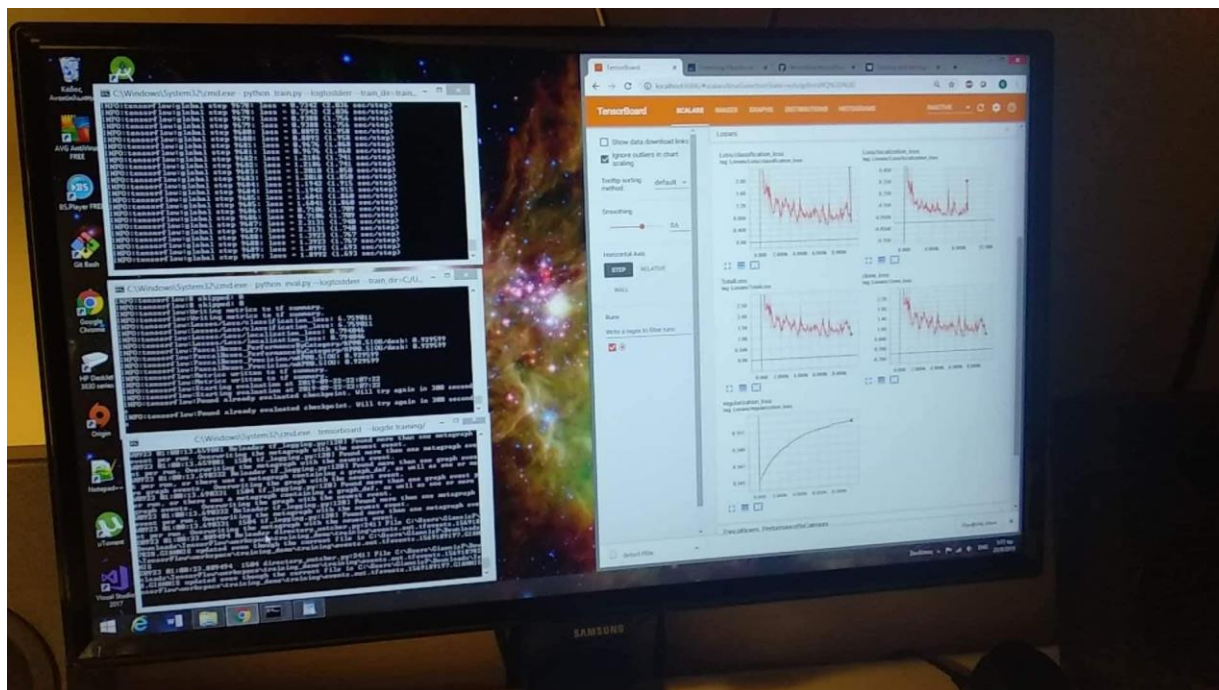
Pre-trained Models

COCO-trained models

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes
faster_rcnn_nas_lowproposals_coco	540		Boxes
mask_rcnn_inception_resnet_v2_atrous_coco	771	36	Masks
mask_rcnn_inception_v2_coco	79	25	Masks

Εκπαίδευση

Έναρξη εκπαίδευσης μοντέλου και επισκόπηση προόδου.



Εξαγωγή Frozen Inference Graph

Αφού τελειώσει η εκπαίδευση χρησιμοποιώ ένα scriptaki που έχει το object detection model για την δημιουργία του graph ο οποίος θα χρησιμοποιηθεί στο Android-App. Ως frozen graph ορίζεται ένας serialized graph, ο οποίος δεν μπορεί να εκπαιδευτεί περισσότερο και έχει 'κλειδώσει' όλες τις τελικές τιμές του όπως βάρη, biases, κλπ που χρησιμοποιούνται για να κάνουν inference σε μια εικόνα (Αποτελεί δηλαδή μια μορφή παράστασης γνώσης, η οποία χρησιμοποιείται για εξαγωγή συμπερασμάτων).

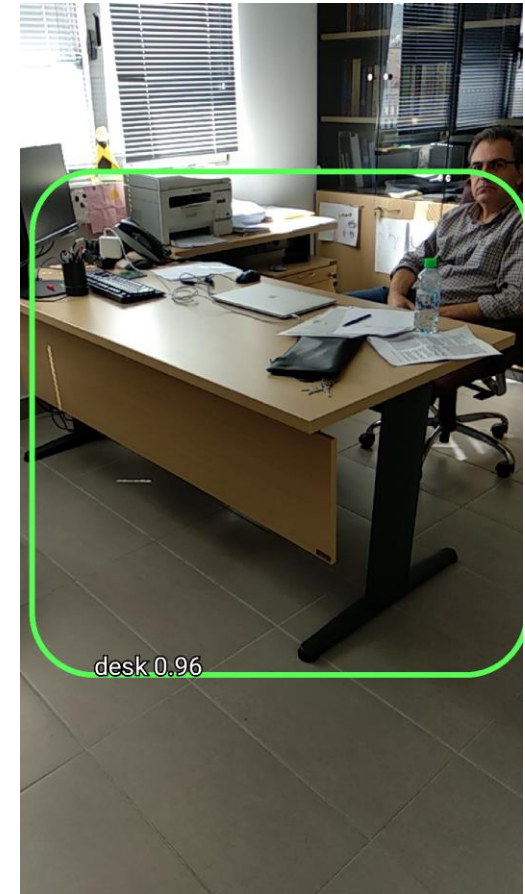
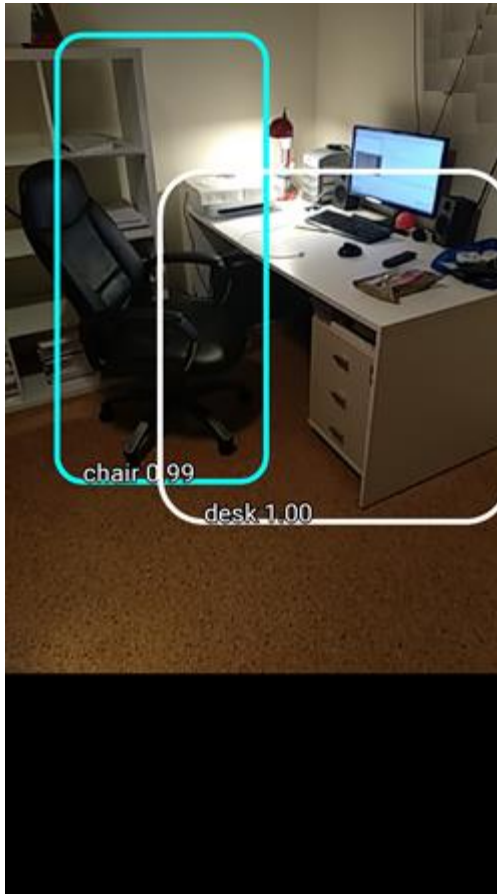
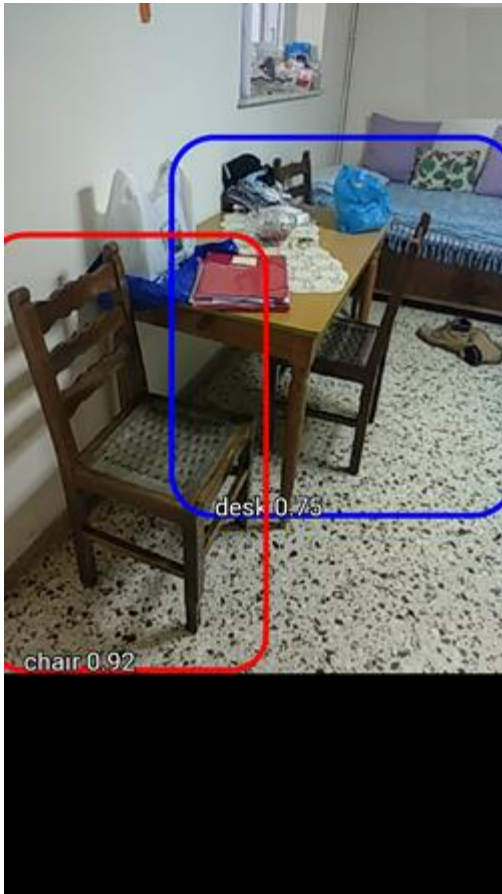
Δομή Εντολής

```
python object_detection/export_tflite_ssd_graph \  
  --pipeline_config_path path/to/ssd_mobilenet.config \  
  --trained_checkpoint_prefix path/to/model.ckpt \  
  --output_directory path/to/exported_model_directory
```

Ενσωμάτωση Graph στο Android App

Κλωνοποίηση του Tensorflow-repository και διαμόρφωση του android object-detection example ώστε να αναγνωρίζει τα αντικείμενα που θέλω. Συγκεκριμένα γραφείο και καρέκλες γραφείου.

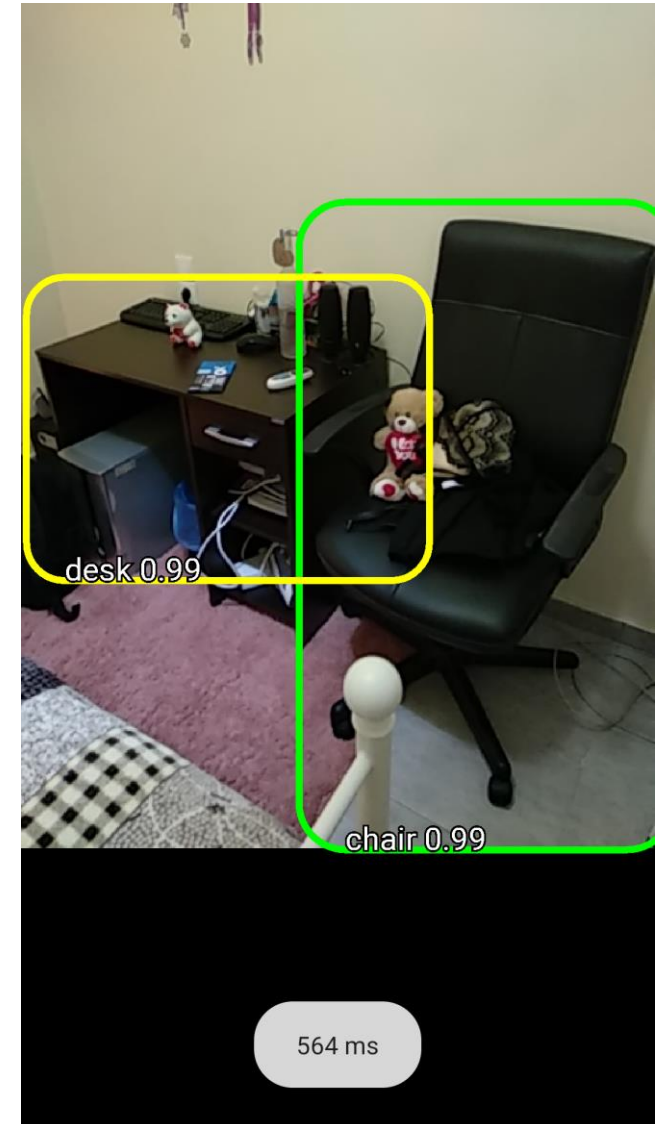
Τελικά αποτελέσματα Android App



Android App

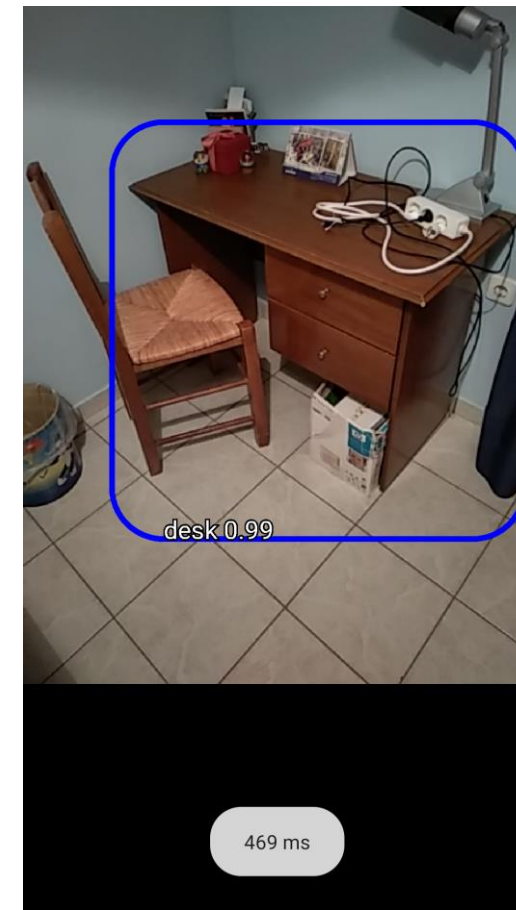
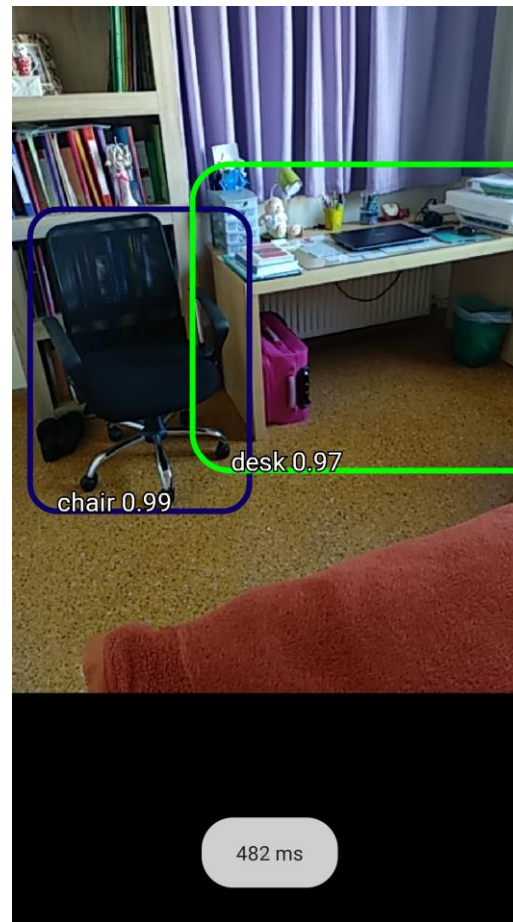
Παρέμβαση στο DetectorActivity.java ώστε όταν βρίσκει ένα αντικείμενο, να εμφανίζεται το inference time.

```
//if a Result is found make a toast for  
detection time  
Context context = getApplicationContext();  
CharSequence text =  
String.valueOf(lastProcessingTimeMs)+ " ms";  
int duration = Toast.LENGTH_SHORT;  
Toast.makeText(context, text,  
duration).show();
```



Τελικά Αποτελέσματα Android App Μετά Από Βελτίωση

Από ~700 ms που ήταν το inference time στο tensorflow mobile app (train με float model(ssd_mobilenet_v1_coco)), κατάφερε να φτάσει το ~450 ms (train με quantized model(ssd_mobilenet_v2_quantized_coco)).



Δημιουργία Tensorflow Lite App

Κλωνοποίηση του Tensorflow-lite repository και διαμόρφωση του android object-detection example ώστε να αναγνωρίζει τα αντικείμενα που θέλω αξιοποιώντας τις δυνατότητες της Lite έκδοσης. Το tensorflow lite είναι σχεδιασμένο για να διευκολύνει το machine learning on device.

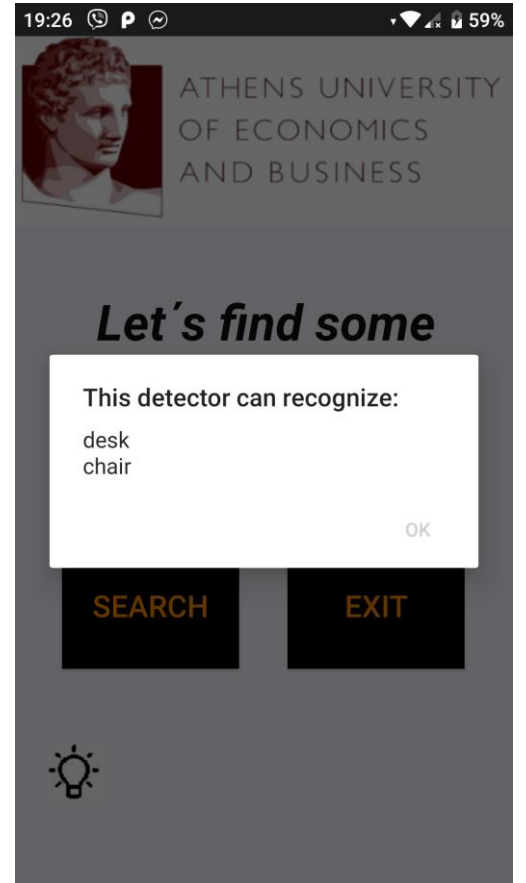
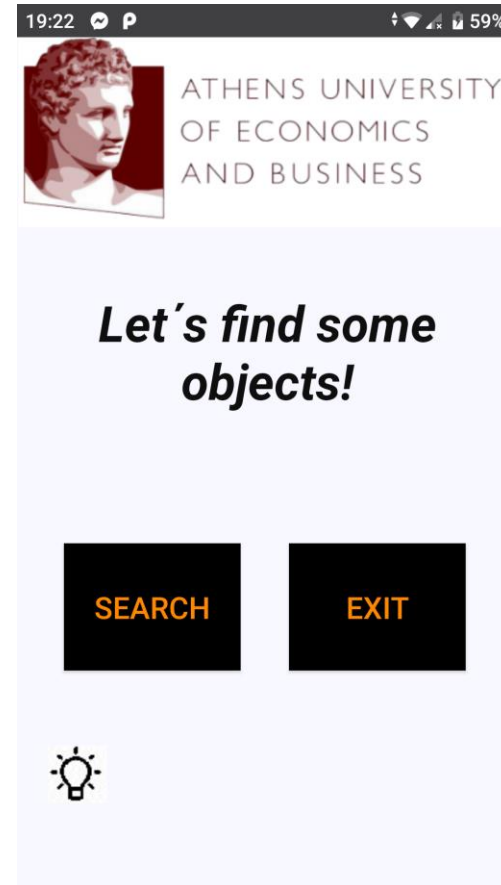
Βασικά Χαρακτηριστικά που μειώνουν το Inference Time στο Tensorflow Lite

1. Επιτάχυνση Interpreter με τη χρήση ενός set of cores οι οποίοι είναι βελτιστοποιημένοι για τις συσκευές
2. Υψηλό Performance με τη αξιοποίηση του hardware, με device-optimized kernel και με pre-fused activations.
3. Εργαλεία βελτιστοποίησης μοντέλων (όπως quantization) τα οποία μειώνουν το μέγεθος και αυξάνουν την απόδοση, χωρίς να επιδρούν στην ακρίβεια.
4. Αποτελεσματική μορφή μοντέλων, με χρήση ενός FlatBuffer ο οποίος είναι βελτιστοποιημένος για μικρό μέγεθος και φορητότητα.

<https://www.tensorflow.org/lite/guide>

Δημιουργία Tensorflow Lite App

1) Δημιουργία αρχικής οθόνης.



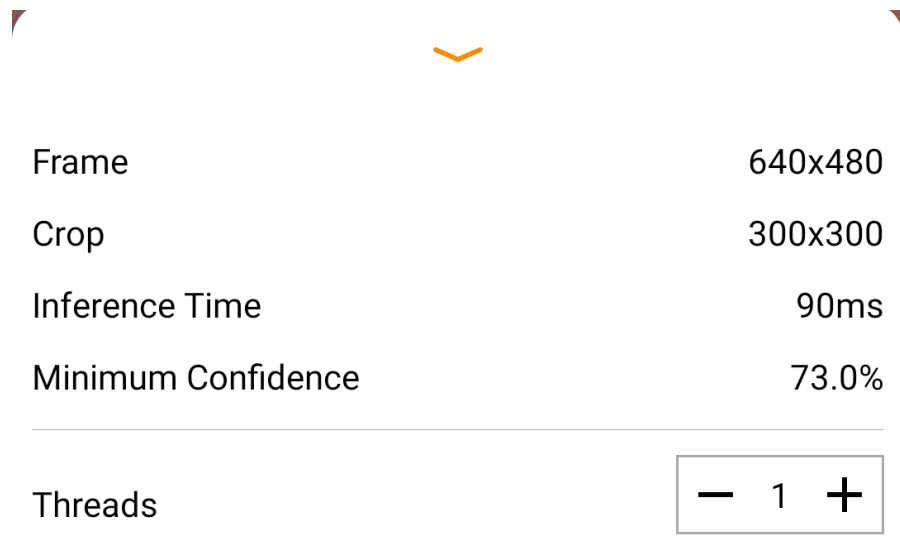
Δημιουργία Tensorflow Lite App

2) Παρέμβαση στο MultiBoxTracker.java καθώς όταν δεν υπήρχε η εντολή clear, σε περίπτωση μη εύρεσης αντικειμένου freezare το κουτάκι από την τελευταία εύρεση.

```
if (rectsToTrack.isEmpty()) {  
    logger.v("Nothing to track, aborting.");  
    //if nothing has to show then clear  
    trackedObjects.clear();  
    return;  
}
```

Δημιουργία Tensorflow Lite App

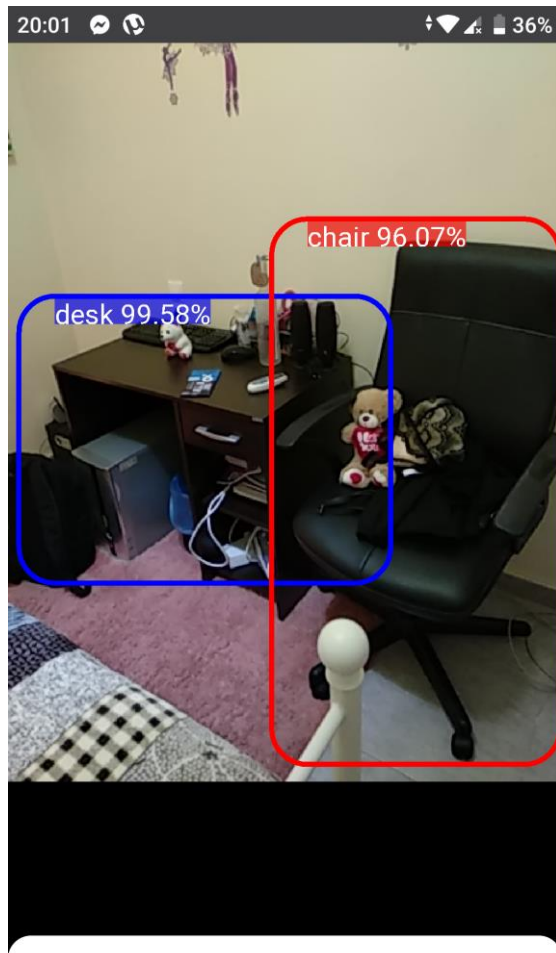
3) Παρέμβαση στο layout_bottom_sheet.xml ώστε να φαίνεται και το minimum confidence percentage.



The screenshot shows a bottom sheet with a white background and a light gray border. At the top, there is a small orange chevron icon pointing downwards. Below this, there is a table with four rows of configuration parameters. The first three rows are separated by a horizontal line, and the last row is also separated by a horizontal line. The 'Threads' row has a control box with minus, plus, and number 1 buttons.

Frame	640x480
Crop	300x300
Inference Time	90ms
Minimum Confidence	73.0%
Threads	<div>— 1 +</div>

Αποτελέσματα Tensorflow Lite App



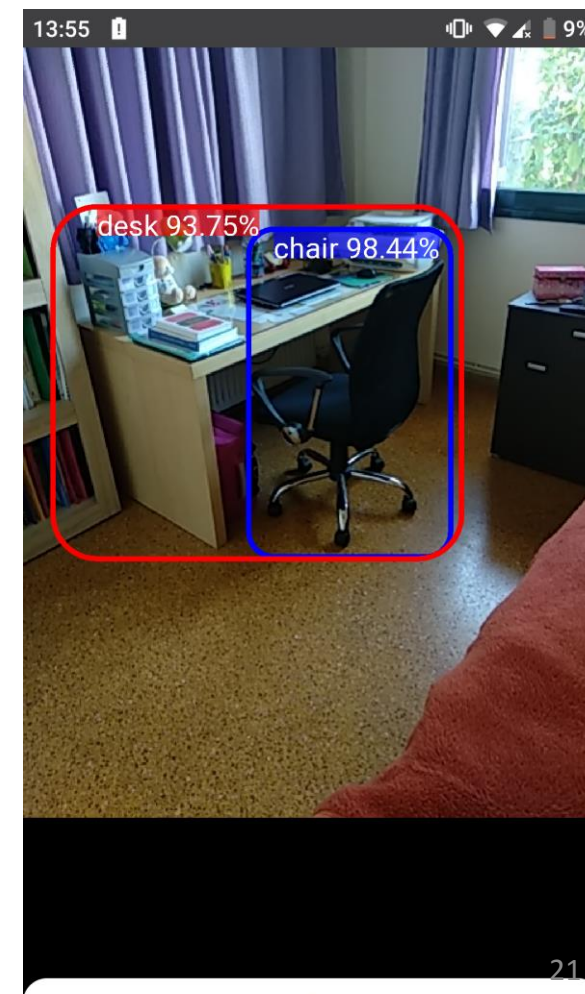
Βελτίωση Tensorflow Lite App

- 1) Αρχικά εκπαιδεύσα με το μοντέλο πάνω στο ίδιο dataset.
- 2) Στη συνέχεια διαμόρφωσα τον Interpreter(TFLiteObjectDetectionAPIModel.java) ώστε να κάνει inference στη Gpu του κινητού.

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v2_quantized_coco	29	22	Boxes

```
// this is for gpu
GpuDelegate delegate = new GpuDelegate();
Interpreter.Options options = (new Interpreter.Options()).addDelegate(delegate);
try {
    //d.tflite = new Interpreter(loadModelFile(assetManager, modelName));
    //without gpu uncomment this remove import org.tensorflow.lite.gpu.GpuDelegate;
    and implement lite-gpu from build gradle
    d.tflite = new Interpreter(loadModelFile(assetManager, modelName), options);
} catch (Exception e) {
    throw new RuntimeException(e);
}
```

Τελικά αποτελέσματα Tensorflow Lite App (1)



Τελικά αποτελέσματα Tensorflow Lite App (2)



Frame 640x480
Crop 300x300
Inference Time 96ms
Minimum Confidence 73.0%

Threads

- 1 +



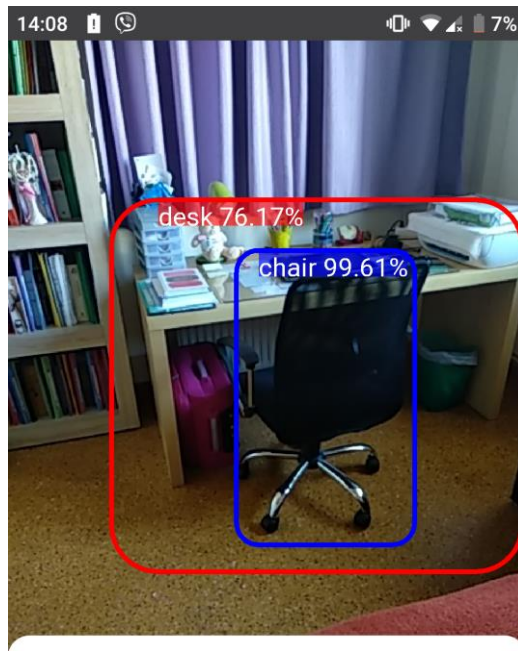
Frame 640x480
Crop 300x300
Inference Time 94ms
Minimum Confidence 73.0%

Threads

- 1 +

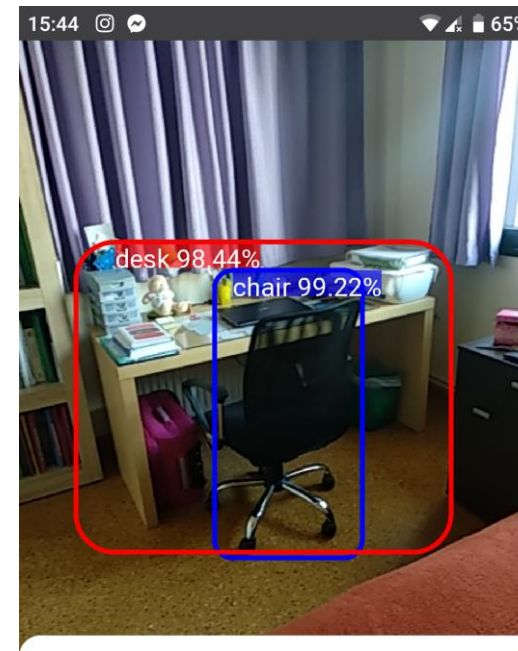
Σύγκριση Inference Time Tensorflow Lite App (Πριν Και Μετά Τη Βελτίωση)

Πριν



Frame 640x480
Crop 300x300
Inference Time 175ms
Threads - 1 +

Μετά



Frame 640x480
Crop 300x300
Inference Time 93ms
Threads - 1 +

Συμπεράσματα

Από ~700ms που ήταν το inference time στο tensorflow mobile app (train με float model), κατάφερε να φτάσει το ~450ms (train με quant model).

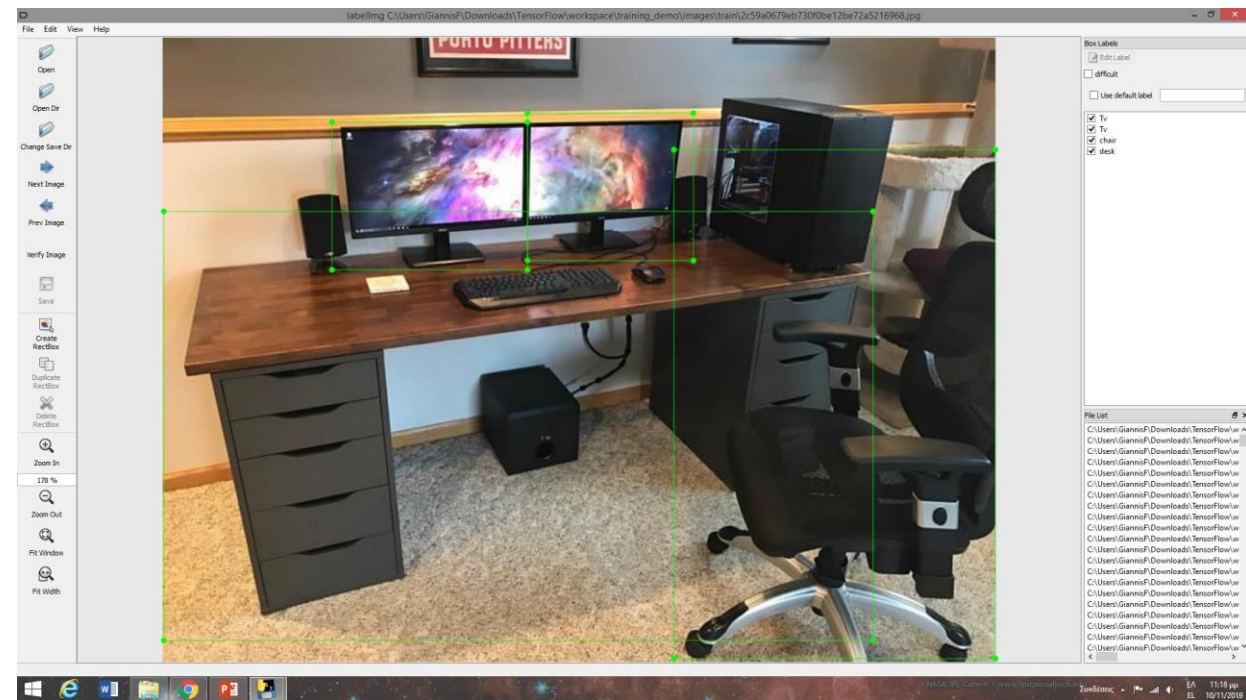
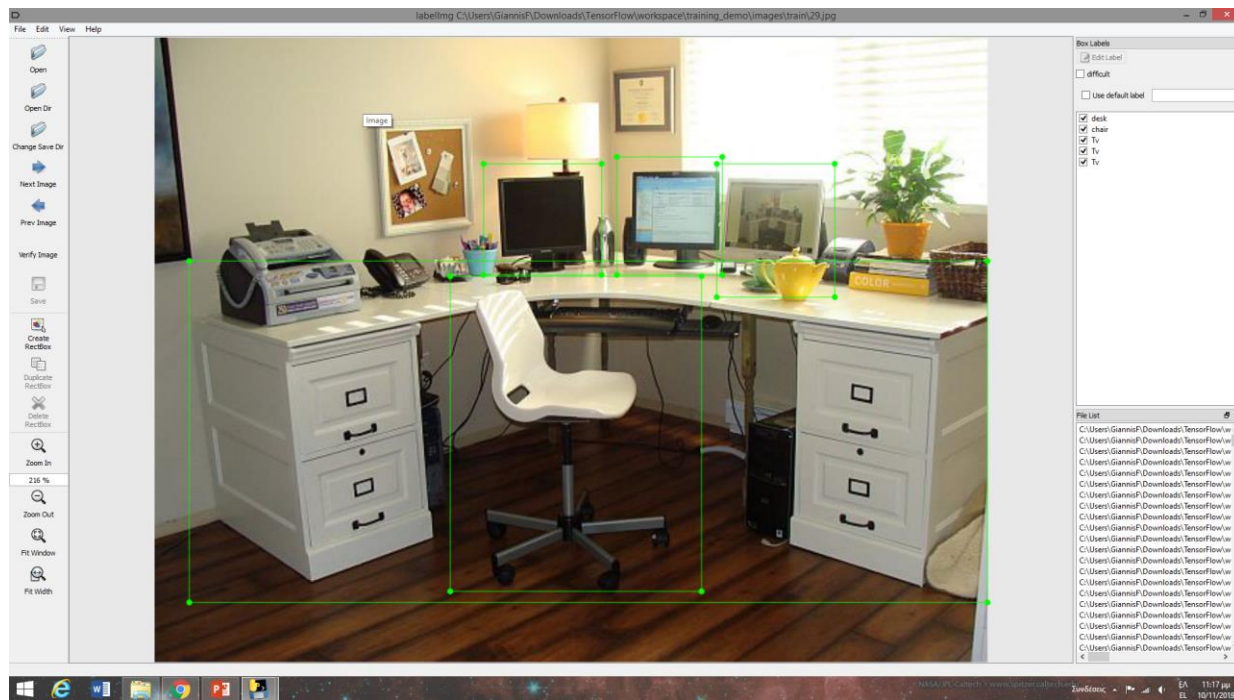
Απο ~450ms που ήταν το inference time στο tensorflow mobile app (train με quant model), κατάφερε να φτάσει το ~200ms (train με float model) με τη δημιουργία του tensorflow lite app.

Με το χτίσιμο του tensorflow lite app το inference time από ~200ms (train με float model), κατάφερε να φτάσει το ~140ms (train με quant model) και με τη βοήθεια της GPU έφτασε τελικά το ~80ms (inference στη GPU + train με quant model).

Μελλοντική Δουλειά

- Δημιουργία dataset και train για αναγνώριση desk, chair, Tv.
- Ερευνά για το πως και αν γίνεται το inference να γίνει στη μνήμη cache.

Δημιουργία Νέου Dataset Προκειμένου να Αναγνωρίζει Και Tv



Αποτελέσματα Tensorflow Lite App Για Το Καινούργιο Dataset

Εκπαίδευση του καινούργιου dataset, το οποίο περιέχει και το αντικείμενο Tv (πέρα από desk, chair) με χρήση `ssd_mobilenet_v2_quantized_coco`.

* Παρατηρούμε πως το inference time δεν αλλάζει ακόμα και αν προσθέσουμε και άλλο αντικείμενο.



Frame	640x480
Crop	300x300
Inference Time	103ms
Minimum Confidence	73.0%
Threads	- 1 +



Frame	640x480
Crop	300x300
Inference Time	87ms
Minimum Confidence	73.0%
Threads	- 1 +



Frame	640x480
Crop	300x300
Inference Time	87ms
Minimum Confidence	73.0%
Threads	- 1 +



Frame	640x480
Crop	300x300
Inference Time	92ms
Minimum Confidence	73.0%
Threads	- 1 +



Frame	640x480
Crop	300x300
Inference Time	103ms
Minimum Confidence	73.0%
Threads	- 1 +

Αποτελέσματα Έρευνας Για Χρήση Cache

Μετά το διάβασμα των paper, και έρευνα στο διαδίκτυο σχετικά με τη χρήση της cache για επιτάχυνση του inference Time κατέληξα στα εξής συμπεράσματα.

- 1) Το paper αναφέρει τη δημιουργία και χρήση της βιβλιοθήκης (NCNN) η οποία έχει όμως χρήση μονό στο TensorFlow Mobile App (ncnn is a high-performance neural network inference computing framework optimized for mobile platforms)(Μείωση inference time από 600 ms σε 400 ms. Το tensorflow lite app που έχουμε κατασκευάσει εμείς κάνει inference time σε 80ms).
- 2) Το android studio σου επιτρέπει να χρησιμοποιήσεις ένα κομμάτι της cache το οποίο δεν μπορεί να ξεπερνά το 1 MB.(You should always maintain the cache files yourself and stay within a reasonable limit of space consumed, such as 1MB). Το Tensorflow Lite App χρησιμοποιεί για inference ένα συγκεκριμένο τύπο αρχείου(.tflite) το οποίο είναι >1 MB. Δεν μπορώ να χρησιμοποιήσω απλά μια εικόνα με το label της για να κάνω inference. Χρειάζομαι αναγκαστικά αυτό το αρχείο.
<https://developer.android.com/guide/topics/data/data-storage>
- 3) Περαιτέρω optimization του inference στο tensorflow lite app (πέρα από την εκπαίδευση με quant μοντέλο και τη χρήση GPU για inference) βρήκα ότι μπορώ να κάνω με χρήση του Neural Network Api (NNAPI) που χρησιμοποιεί διαφορές λειτουργίες του hardware, το οποίο όμως χρησιμοποιείται ήδη μέσα στην εφαρμογή.

Δεν υπάρχει κάποιος άλλος τρόπος για περαιτέρω optimization του inference, που να αναφέρεται στο διαδίκτυο, πέρα από αυτά που ήδη έχουμε κάνει.